|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | «Информатика и системы управления» |
| КАФЕДРА | «Теоретическая информатика и компьютерные технологии» |

**КУРСОВАЯ РАБОТА**

***НА ТЕМУ:***

|  |
| --- |
| ***Использование методов фильтрации*** |
| ***аудиосигналов в мобильном приложении*** |
|  |
|  |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ9-72Б |  |  |  | Караник А.А. |
|  | (группа) |  | (подпись, дата) |  | (И.О. Фамилия) |
| Руководитель курсового проекта |  |  |  |  | Каганов Ю.Т. |
|  |  |  | (подпись, дата) |  | (И.О. Фамилия) |
|  |  |  |  |  |  |
| Консультант |  |  |  |  |  |
|  |  |  | (подпись, дата) |  | (И.О. Фамилия) |

*2024 г.*

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 1](#_Toc190097577)

[ВВЕДЕНИЕ 2](#_Toc190097578)

[1. Цель и основные задачи 3](#_Toc190097579)

[2. Описание методов 4](#_Toc190097580)

[2.1. Низкочастотные и высокочастотные фильтры 4](#_Toc190097581)

[2.2. Полосовой фильтр 6](#_Toc190097582)

[2.3. Метод Калмана 6](#_Toc190097583)

[2.4. Фильтр Гаусса и медианный фильтр 8](#_Toc190097584)

[2.5. Метод шумоподавления с использованием спектрального вычитания 10](#_Toc190097585)

[3. Реализация 13](#_Toc190097586)

[3.1. Выбор стека технологий 13](#_Toc190097587)

[3.2. Реализация методов и интерфейса приложения 14](#_Toc190097588)

[4. Тестирование 25](#_Toc190097589)

[ЗАКЛЮЧЕНИЕ 31](#_Toc190097590)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 32](#_Toc190097591)

ВВЕДЕНИЕ

Развитие мобильных технологий и рост вычислительных мощностей смартфонов открыли новые возможности для обработки аудиосигналов в реальном времени. Современные устройства оснащены мощными процессорами и специализированными цифровыми сигнальными процессорами, которые позволяют эффективно применять сложные численные методы фильтрации без значительных задержек. Это особенно важно для интерактивных приложений, таких как голосовые помощники, системы видеозвонков и аудиоредакторы, где качество звука напрямую влияет на удобство использования.

Важность методов фильтрации аудиосигналов также связана с развитием технологий искусственного интеллекта и машинного обучения. Современные алгоритмы обработки речи и звука включают элементы адаптивной фильтрации, позволяя устройствам подстраиваться под изменяющиеся условия окружающей среды. Например, в шумных местах смартфоны могут автоматически усиливать голос пользователя и подавлять посторонние звуки, что значительно улучшает восприятие речи. Такие методы находят применение в системах автоматического перевода, голосового управления и биометрической идентификации по голосу.

Кроме того, методы фильтрации активно применяются в аудиостриминговых сервисах для повышения качества воспроизведения музыки. Использование алгоритмов эквализации, динамической компрессии и шумоподавления позволяет улучшать звук в зависимости от типа наушников или акустической среды. Многие современные музыкальные приложения внедряют персонализированные алгоритмы обработки, адаптирующие звук под предпочтения пользователя.

Использование методов фильтрации аудиосигналов в мобильных приложениях является не только важной технологической задачей, но и направлением, способствующим развитию новых форм взаимодействия человека с цифровыми устройствами. Совершенствование этих методов позволяет не только повысить качество звука, но и создать более интеллектуальные системы, способные адаптироваться к потребностям пользователей и обеспечивать комфортное аудиообщение в любых условиях.

1. Цель и основные задачи

Целью данной курсовой работы является изучение и анализ методов фильтрации аудиосигналов, а также их практическое применение в мобильных приложениях для улучшения качества звука, подавления шума и повышения разборчивости речи.

Основные задачи:

1. Изучение методов фильтрации аудиосигналов:
   1. Анализ основных методов обработки аудиосигналов.
   2. Рассмотрение их преимуществ, недостатков и областей применения в мобильных приложениях.
2. Реализация методов фильтрации аудиосигналов:
   1. Выбор наиболее подходящих алгоритмов для цифровой обработки звука.
   2. Разработка и программная реализация фильтров с использованием языка программирования Kotlin.
3. Написание интерфейса для применения данных методов в мобильном приложении:
   1. Создание пользовательского интерфейса для взаимодействия с системой фильтрации.
   2. Интеграция фильтрации аудиосигналов в мобильное приложение с учетом производительности и удобства использования.
4. Тестирование и оценка результатов:
   1. Проведение тестирования работы фильтров на различных аудиосигналах.
   2. Оценка качества фильтрации, анализа производительности и выявление возможных улучшений.
5. Описание методов
   1. Низкочастотные и высокочастотные фильтры

Фильтрация сигналов играет ключевую роль в цифровой обработке аудио, позволяя выделять полезные частотные компоненты и подавлять шумы. Одними из наиболее распространенных типов фильтров являются низкочастотные (Low-Pass Filter, LPF) и высокочастотные (High-Pass Filter, HPF).

Низкочастотный фильтр (НЧ-фильтр) предназначен для подавления частот выше заданной частоты среза и сохранения более низких частот. Такие фильтры применяются в системах шумоподавления, улучшения качества звука, выделения низкочастотных компонентов, например, в голосовых сигналах.

Одним из простейших вариантов является фильтр первого порядка, описываемый дифференциальным уравнением:

(1)

где:

* – входной сигнал.
* – выходной сигнал.
* – постоянная времени фильтра, вычисляемая по формуле:

(2)

В цифровой обработке сигналов применяется разностное уравнение первого порядка:

(3)

где:

* – входной сигнал в дискретном случае.
* – выходной сигнал в дискретном случае.
* – коэффициент фильтрации, вычисляемый по формуле:

(4)

где – шаг дискретизации, а – частота дискретизации.

Этот фильтр выполняет экспоненциальное сглаживание входного сигнала и используется, например, для подавления высокочастотного шума в голосовых приложениях и музыкальных редакторах.

Высокочастотный фильтр (ВЧ-фильтр) выполняет противоположную задачу – подавляет низкие частоты и пропускает высокие. Он используется для удаления низкочастотного гула, подавления вибраций и выделения высокочастотных деталей в аудиосигнале.

Один из простейших реализуемых ВЧ-фильтров имеет дифференциальное уравнение:

(5)

В отличие от НЧ-фильтра, этот фильтр подавляет низкие частоты и усиливает высокие.

В цифровом виде высокочастотный фильтр можно описать разностным уравнением:

(6)

где в свою очередь вычисляется по следующей формуле:

(7)

Этот фильтр удаляет медленно изменяющиеся компоненты сигнала, такие как низкочастотные шумы, оставляя быстрые колебания.

В мобильных приложениях сочетание этих фильтров помогает адаптировать аудиосигналы под различные условия. Например, в VoIP-связи HPF устраняет низкочастотный фоновый шум, а LPF сглаживает резкие артефакты. В музыкальных редакторах LPF усиливает басы, а HPF повышает четкость звучания инструментов.

НЧ и ВЧ фильтры являются важнейшими инструментами цифровой обработки сигналов и находят широкое применение в мобильных технологиях.

* 1. Полосовой фильтр

Полосовой фильтр (Band-Pass Filter, BPF) – это разновидность частотного фильтра, предназначенного для пропускания сигналов в определенном диапазоне частот и ослабления всех остальных. Такой фильтр сочетает в себе свойства низкочастотного (Low-Pass Filter, LPF) и высокочастотного (High-Pass Filter, HPF) фильтров, работая по принципу удаления частот, расположенных ниже нижней границы полосы пропускания и выше ее верхней границы. Полосовой фильтр можно реализовать как в аналоговом, так и в цифровом виде, что делает его важным инструментом в обработке сигналов, в том числе в мобильных приложениях для работы с аудиосигналами.

Формула для полосового фильтра в дискретном представлении выглядит следующим образом:

(8)

где – выходной сигнал низкочастотного фильтра, а – выходной сигнал высокочастотного фильтра.

Этот метод фильтрации эффективен для выделения определенных частотных диапазонов в аудиосигналах, например, для обработки речи или музыкальных инструментов. В мобильных приложениях такие фильтры применяются для шумоподавления, улучшения разборчивости речи и обработки звуковых эффектов. Выбор частотных границ полосового фильтра зависит от конкретных задач, например, в радиосистемах полосовой фильтр может использоваться для выделения сигнала определенной частоты, а в аудиотехнике – для подавления нежелательных шумов.

* 1. Метод Калмана

Фильтр Калмана – это рекурсивный алгоритм оценки состояния системы, подверженной случайным шумам. Он позволяет предсказывать и корректировать значения наблюдаемых параметров, минимизируя влияние шума и неопределенности. Метод находит широкое применение в обработке аудиосигналов, особенно при подавлении шума и улучшении качества звука, так как позволяет динамически адаптироваться к изменяющимся характеристикам сигнала.

Математическая основа фильтра Калмана базируется на линейной модели системы, которая описывает эволюцию состояния во времени. Пусть – вектор состояния системы в момент времени , тогда динамика системы описывается уравнением:

(9)

где:

* – матрица перехода состояния, описывающая изменение состояния во времени.
* – матрица управления.
* – управляющий вход.
* – шум процесса, который считается случайным гауссовым шумом с нулевым средним и ковариационной матрицей :

(10)

Система производит измерения, которые связаны с истинным состоянием следующим уравнением:

(11)

где:

* – наблюдаемый вектор измерений.
* – матрица наблюдений, связывающая состояние системы с измеряемыми параметрами.
* – шум измерений, также предполагаемый гауссовым с нулевым средним и ковариационной матрицей :

(12)

Фильтр Калмана выполняет два ключевых шага: предсказание и обновление. На этапе предсказания рассчитывается априорная оценка состояния и его ковариации:

(13)

(14)

где – это предсказанное состояние, а – ковариационная матрица ошибки предсказания.

После получения нового измерения производится корректировка состояния. Для этого вычисляется матрица Калмана, определяющая, насколько следует доверять измерениям по сравнению с модельным предсказанием:

(15)

Затем обновляется оценка состояния:

(16)

И пересчитывается ковариационная матрица ошибки:

(17)

Фильтр Калмана на каждом шаге динамически корректирует свою оценку состояния на основе новых измерений. В обработке аудиосигналов этот метод позволяет эффективно подавлять шум, поскольку он использует априорную информацию о модели сигнала и адаптируется к изменениям в данных. Например, если шум обладает известной статистикой, фильтр Калмана может выделять полезный сигнал, минимизируя влияние случайных возмущений. В частности, он применяется в шумоподавлении речи, удалении фона в аудиозаписях и повышении разборчивости голоса в зашумленных условиях.

* 1. Фильтр Гаусса и медианный фильтр

Методы фильтрации, такие как гауссов и медианный фильтры, широко используются в обработке аудиосигналов для подавления шума. Они относятся к категории линейных и нелинейных фильтров, соответственно, и применяются для улучшения качества звука в различных условиях.

Фильтр Гаусса является линейным фильтром, основанным на применении ядра с гауссовым распределением для сглаживания сигналов. Ядро Гаусса имеет вид функции Гаусса:

(18)

где – расстояние от центра ядра (от текущего элемента в аудиосигнале), а – стандартное отклонение, которое контролирует ширину гауссовой кривой.

Суть работы гауссова фильтра заключается в том, чтобы применить это ядро к сигналу, заменяя каждое значение на взвешенное среднее значений из окрестности текущего элемента с учетом гауссовых весов. Таким образом, каждый элемент сигнала будет более похож на соседние значения, что снижает влияние случайных шумов.

Процесс применения гауссова фильтра в случае одномерного аудиосигнала можно описать следующим образом. Пусть – это входной сигнал, а – выходной сигнал после фильтрации. Тогда выходной сигнал для каждого значения вычисляется как свертка входного сигнала с ядром Гаусса:

(19)

где – значение ядра Гаусса для сдвига , а – значения аудиосигнала на соответствующих временных позициях.

Медианный фильтр, в отличие от гауссова, является нелинейным фильтром, который работает, заменяя каждое значение сигнала медианой значений в окрестности текущего элемента. Такой фильтр эффективен для удаления импульсного шума, который выражается в виде резких выбросов. Окно медианного фильтра имеет фиксированный размер, например, , и медианная фильтрация для каждого элемента сигнала рассчитывается как медиана всех значений в окне вокруг текущего элемента.

Формально медианная фильтрация для одномерного сигнала может быть записана следующим образом. Пусть – это входной сигнал, а – его фильтрованный вариант. Тогда выходной сигнал на позиции определяется как медиана значений во временном окне вокруг :

(20)

где – размер окна, которое определяется пользователем в зависимости от характеристик шума. Для каждого значения в окне берутся все соседние элементы сигнала, затем эти значения сортируются, и медианное значение (то есть центральное значение после сортировки) становится результатом фильтрации.

Главное отличие медианного фильтра от фильтра Гаусса заключается в том, что медианный фильтр не применяет взвешенные средние значения, а использует медиану, что позволяет эффективно удалять импульсные шумы, не размывая сигнал. Например, если в окне фильтрации присутствуют аномально большие или маленькие значения, такие как выбросы, медианный фильтр будет игнорировать их и заменять на центральное значение, которое лучше отражает нормальные данные.

Оба этих фильтра используются для улучшения качества аудиосигнала, но их применение зависит от типа шума, с которым нужно работать. Гауссов фильтр часто применяется для устранения гауссовского шума, который представляет собой плавные и случайные колебания сигнала, в то время как медианный фильтр более эффективен при наличии резких выбросов или импульсного шума.

* 1. Метод шумоподавления с использованием спектрального вычитания

Метод спектрального вычитания является одним из эффективных подходов к подавлению шума в аудиосигналах. Этот метод основывается на вычислении спектра шума и вычитании его из спектра сигнала, что позволяет существенно снизить уровень шума при сохранении основных характеристик сигнала. Рассмотрим этот процесс более подробно.

Для начала необходимо преобразовать сигнал из временной области в частотную с помощью преобразования Фурье. Это можно сделать с использованием дискретного преобразования Фурье (DFT) или быстрого преобразования Фурье (FFT). Суть преобразования заключается в разложении сигнала на синусоидальные составляющие с различными частотами.

Будем считать, что сигнал представлен как сумма синусоидальных волн:

(21)

где – спектр сигнала, а – частота.

Преобразуя сигнал в частотную область, мы получаем его амплитудный спектр, который характеризует распределение энергии сигнала по частотам.

Для применения метода спектрального вычитания нужно сначала оценить спектр шума. На практике, для этого выбирается участок аудиофайла, который не содержит полезной информации (например, тишины или фона), и из этого участка вычисляется спектр шума. Таким образом, получаем спектр шума, который в дальнейшем будем использовать для вычитания.

Спектр шума рассчитывается аналогично спектру сигнала:

(22)

где – это шум, а – его спектр.

Для получения более точной оценки спектра шума может использоваться несколько оконных фрагментов записи шума, которые затем усредняются. Усредненный спектр шума можно вычислить как:

(23)

где – это количество окон для расчета усредненного спектра шума, а – усредненный спектр шума, который используется для вычитания из спектра сигнала.

После того как спектр шума получен, можно переходить к основному этапу – вычитанию спектра шума из спектра сигнала. Пусть амплитуда спектра сигнала для каждой частоты равна , а амплитуда спектра шума – . Тогда для каждой частоты вычитание происходит следующим образом:

(24)

где – это коэффициент усиления, который определяет, насколько сильно будет подавляться шум. Чем больше , тем шум подавляется сильнее, в противном случае его остается больше.

Важно, чтобы результат вычитания был всегда положительным, потому что амплитуды в спектре не могут быть отрицательными. Поэтому используется следующая модификация:

(25)

Это предотвращает возникновение отрицательных значений амплитуды и сохраняет физически допустимые значения.

После того как из спектра сигнала был вычтен спектр шума, следующий шаг заключается в восстановлении сигнала во временной области. Для этого применяется обратное преобразование Фурье (IFFT), которое позволяет из частотного спектра восстановить сигнал в исходную временную область:

(26)

В дискретном случае используется обратное быстрое преобразование Фурье (IFFT), которое восстанавливает сигнал из его спектра. На выходе мы получаем аудиофайл с удаленным шумом.

После применения спектрального вычитания может возникать несколько нежелательных эффектов, таких как искажения и потеря качества. Для минимизации этих артефактов можно применить различные методы постобработки. Например, можно применить сглаживание амплитуды в частотной области или использовать методы коррекции фазы для минимизации искажений звука. Эти методы могут улучшить качество сигнала и уменьшить побочные эффекты.

Метод спектрального вычитания является мощным инструментом для подавления шума в аудиосигналах. Он позволяет эффективно удалить шумы в частотной области, что приводит к значительному улучшению качества звука. Однако для достижения хороших результатов важно правильно выбрать коэффициент усиления, а также корректно оценить спектр шума.

1. Реализация
   1. Выбор стека технологий

Разработка мобильного приложения для фильтрации аудиосигналов требует выбора стека технологий, который обеспечит оптимальную производительность, удобство разработки и поддержку современных архитектурных решений. В данной работе в качестве основного языка программирования выбран Kotlin, так как он является официально поддерживаемым языком для разработки под Android и предоставляет множество возможностей для удобной и безопасной работы с данными, многопоточностью и функциональным программированием.

В качестве системы сборки используется Gradle, обеспечивающий гибкую и удобную настройку зависимостей, автоматизацию сборки и управление проектом. Благодаря Gradle упрощается интеграция сторонних библиотек, управление модулями и настройка различных сборочных конфигураций, что делает разработку более эффективной. В проекте используется Kotlin DSL для конфигурации Gradle, что улучшает читаемость и поддерживаемость кода.

Для создания пользовательского интерфейса используется Jetpack Compose, современный декларативный фреймворк от Google, который позволяет гибко и удобно описывать UI, избегая сложностей традиционного XML-разметки. Compose обеспечивает высокую скорость рендеринга интерфейса и тесную интеграцию с ViewModel, позволяя эффективно управлять состоянием приложения.

Архитектура приложения построена на принципах MVVM (Model-View-ViewModel), что позволяет отделить логику работы с данными от представления. В качестве управления состоянием выбрана ViewModel, которая обеспечивает сохранение данных при изменении конфигурации экрана и позволяет удобно работать с потоками данных через StateFlow или LiveData. Это особенно важно при обработке аудиосигналов, так как процесс фильтрации может занимать значительное время и требовать асинхронного выполнения.

Работа с аудиофайлами требует доступа к файловой системе, поэтому в проекте используется MediaStore API, который позволяет безопасно получать доступ к файлам пользователя на устройстве без необходимости запрашивать прямые разрешения.

Для многопоточной обработки в Kotlin используется Coroutines, так как они позволяют эффективно работать с фоновыми задачами без создания большого количества потоков. Это критически важно при фильтрации аудиосигнала, поскольку операции преобразования Фурье и обработки данных могут быть ресурсоемкими.

В качестве среды разработки выбрана Android Studio, которая обеспечивает максимальную поддержку Kotlin и Compose, а также содержит мощные инструменты профилирования, дебага и тестирования производительности.

Стек технологий был выбран с учетом производительности, удобства разработки и поддержки актуальных архитектурных подходов. Использование Kotlin, Gradle, Jetpack Compose, ViewModel, Coroutines позволяет создать современное, удобное и эффективное мобильное приложение для фильтрации аудиосигналов.

* 1. Реализация методов и интерфейса приложения

Разработка мобильного приложения для фильтрации аудиосигналов на Android начинается с настройки базовой структуры проекта. Основным экраном приложения является MainActivity, которая выступает в качестве точки входа и содержит основную логику навигации и отображения интерфейса.

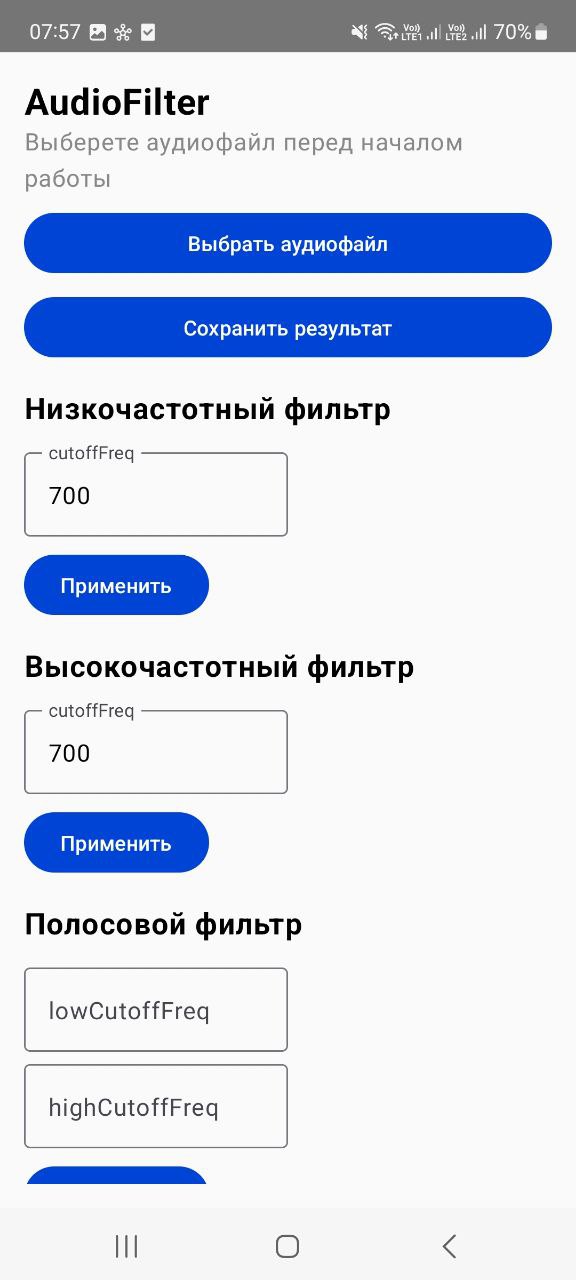


Рисунок 1 – основной экран приложения

Для стилизации интерфейса используется Jetpack Compose, а все визуальные настройки, включая цвета, типографику и формы компонентов, определяются в теме приложения. В файле Theme.kt задаются основные параметры оформления, такие как цветовая палитра (MaterialTheme.colors), шрифты (Typography) и формы элементов (Shapes). Цветовая схема включает в себя основные и дополнительные оттенки, которые применяются ко всем компонентам интерфейса.

В MainActivity вызывается корневой Composable-функция, в которой применяется тема, а затем отображается основной экран. Внутри экрана размещается кнопка выбора аудиофайла, список доступных методов обработки и кнопка сохранения обработанного файла. Все элементы пользовательского интерфейса приложения взаимодействуют с ViewModel, которая играет ключевую роль в управлении состоянием и обработке данных. Эта модель управляет состоянием приложения, обеспечивая функциональность загрузки, обработки и сохранения аудиофайлов. Она служит связующим звеном между пользовательским интерфейсом и логикой работы с аудиосигналами.

Когда пользователь выбирает аудиофайл для обработки, через ViewModel загружается файл и извлекаются данные, такие как сам аудиофайл в виде массива сэмплов и его частота дискретизации. Эти данные сохраняются в переменные внутри ViewModel, что позволяет сохранять текущее состояние приложения, а также передавать эти данные в различные элементы интерфейса, чтобы пользователь мог видеть и взаимодействовать с ними.

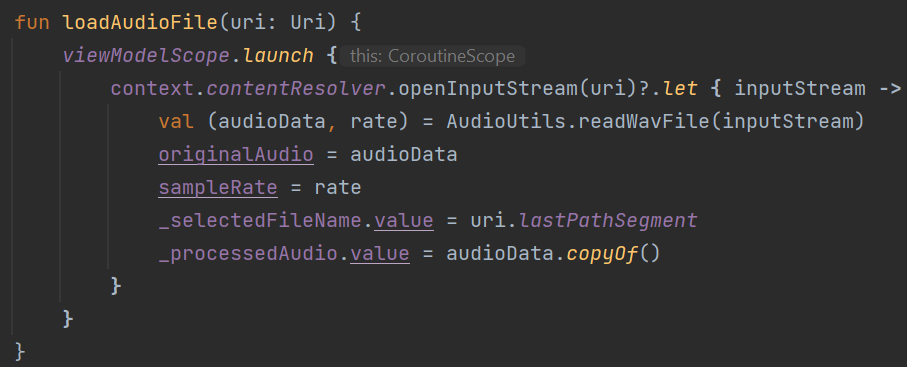


Рисунок 2 – функция загрузки аудиофайла

Одной из основных функций ViewModel является выполнение различных фильтраций и обработок аудиофайла. После применения всех необходимых фильтров и обработки, пользователь может сохранить результат. Для этого в ViewModel реализована функция сохранения обработанного аудиофайла, которая позволяет записать результат в стандартное место для аудиофайлов на устройстве.

Основным принципом работы ViewModel является изоляция логики обработки данных от пользовательского интерфейса, что способствует улучшению структуры приложения и упрощает тестирование. Вся обработка данных происходит в фоновом потоке с использованием корутин, что исключает зависания интерфейса и позволяет пользователю продолжать взаимодействовать с приложением, не прерывая работу.

Для корректной работы мобильного приложения с аудиофайлами необходимо настроить доступ к памяти устройства, а также подключить все необходимые зависимости. В операционной системе Android работа с файлами требует явного указания разрешений, что осуществляется в файле AndroidManifest.xml. В данном проекте требуется доступ к чтению аудиофайлов, поэтому в манифесте были добавлены разрешения READ\_EXTERNAL\_STORAGE и WRITE\_EXTERNAL\_STORAGE. Однако, начиная с Android 10, доступ к памяти стал ограничен, поэтому для более новых версий системы используется Scoped Storage, а для записи файлов применяется MediaStore API или Storage Access Framework. В манифесте также была объявлена основная активность приложения, которая будет запускаться при старте, и указана тема оформления.

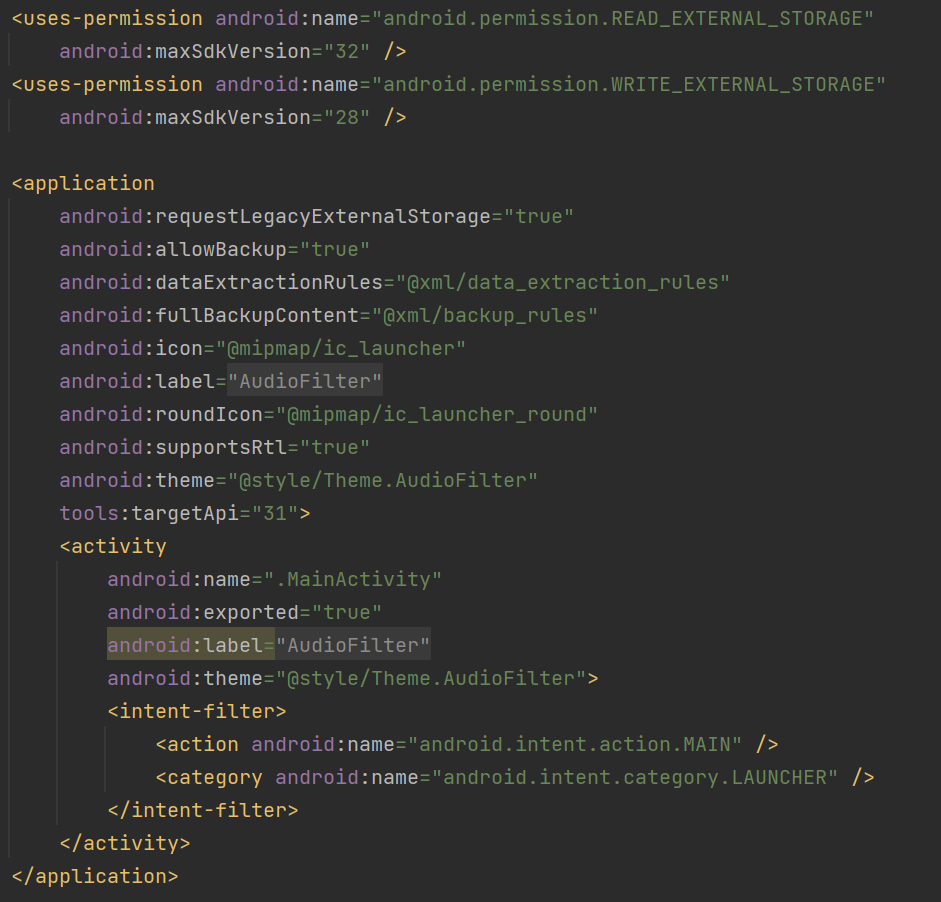


Рисунок 3 – фрагмент файла AndroidManifest.xml

Для сборки проекта и управления зависимостями используется Gradle. В файле build.gradle на уровне проекта были добавлены основные репозитории, такие как Google и Maven Central, обеспечивающие доступ к библиотекам. В файле build.gradle на уровне модуля были заданы версии compileSdk, minSdk и targetSdk, определяющие, с какими версиями Android совместимо приложение. Минимальная поддерживаемая версия была выбрана API 28, так как более ранние версии уже практически не используются.



Рисунок 4 –конфигурация в файле build.gradle

В ходе разработки приложения работа с аудиофайлами велась в формате WAV. Этот формат был выбран благодаря его простоте, отсутствию сжатия и широкому распространению. WAV-файл представляет собой контейнер, содержащий заголовок и несжатые аудиоданные в виде последовательности 16-битных целых чисел. Это позволяет легко обрабатывать звук, не теряя качество из-за компрессии.

Чтение WAV-файла начинается с извлечения заголовка, который содержит информацию о частоте дискретизации, количестве каналов и размере данных. Для корректного парсинга файла считываются первые 44 байта, где расположены метаданные, а затем извлекаются непосредственно аудиоданные. Они интерпретируются как массив 16-битных значений, представляющих амплитуду сигнала во времени.

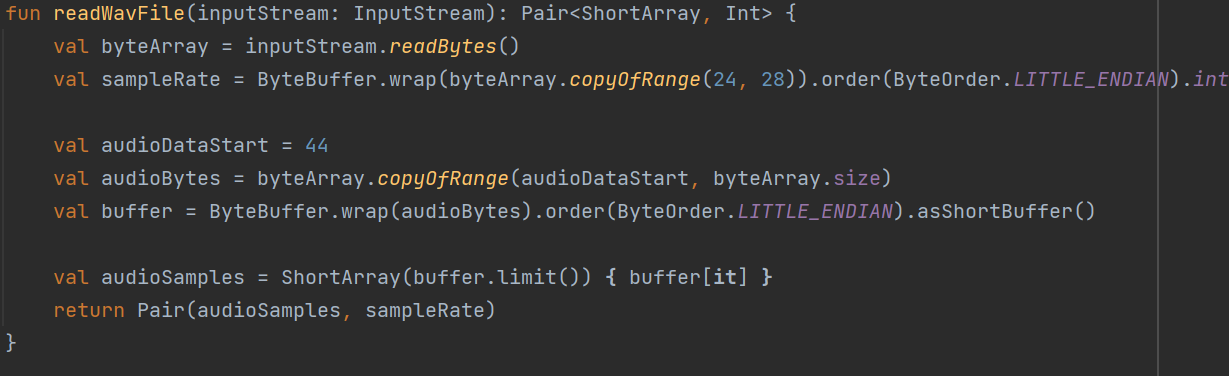


Рисунок 5 – функция чтения WAV-файла

Обработанные аудиоданные сохраняются обратно в формате WAV. Для этого создается новый файл, в который записывается заголовок с актуальными параметрами, такими как длина данных и частота дискретизации. После заголовка записываются обработанные сэмплы в виде 16-битных значений. Запись выполняется в поток, открытый через MediaStore, что обеспечивает доступ к файлу в файловой системе устройства.



Рисунок 6 – функция создания WAV-заголовка

Дополнительно файлы, сохраненные в MediaStore, сканируются системой для отображения в медиагалерее устройства. Это позволяет пользователю легко находить обработанные файлы и воспроизводить их в стандартных аудиоплеерах. WAV-файлы сохраняются в директорию Music, а их имена формируются динамически, чтобы избежать конфликтов с уже существующими файлами.

В основе каждого метода фильтрации лежит итеративная обработка массива выборок аудиосигнала, представленного в виде ShortArray, что соответствует стандартному 16-битному PCM-формату.

Метод низкочастотного фильтра использует простейший рекурсивный фильтр первого порядка. В начале вычисляется постоянная времени, зависящая от частоты среза, а затем рассчитывается коэффициент сглаживания. Далее каждый новый отсчет сигнала формируется как линейная комбинация текущего входного значения и предыдущего выходного, что позволяет постепенно подавлять высокочастотные колебания.

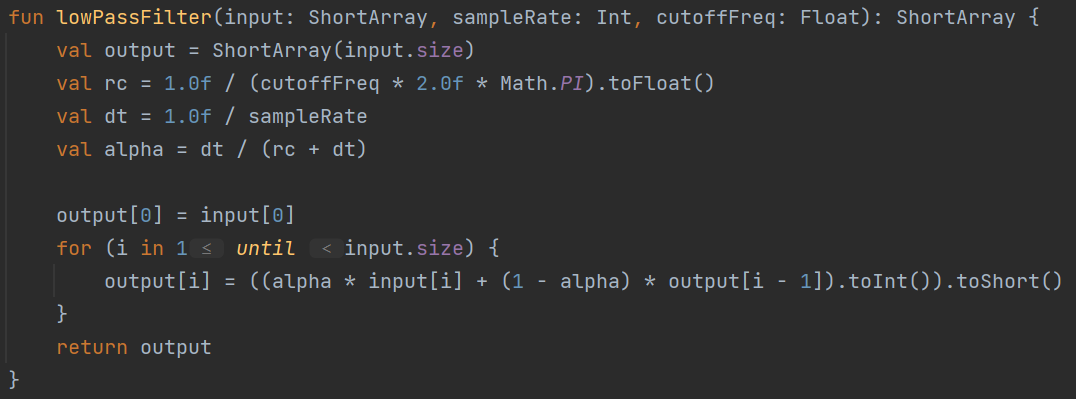


Рисунок 7 – функция низкочастотного фильтра

Высокочастотный фильтр реализуется аналогичным способом, но с другой формулой расчета выходного значения. Здесь используется разность между текущим и предыдущим значением входного сигнала, что позволяет подавлять низкие частоты, оставляя в спектре только высокочастотные составляющие.

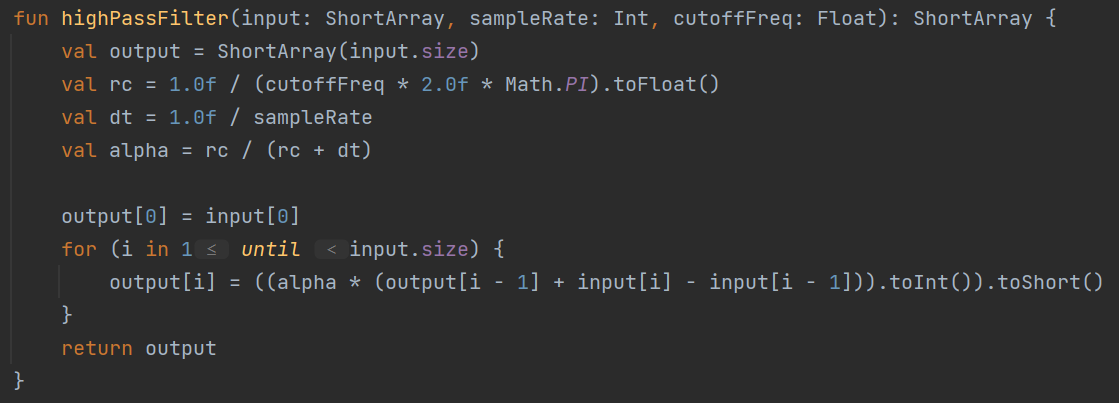


Рисунок 8 – функция высокочастотного фильтра

Полосовой фильтр реализуется комбинацией низкочастотного и высокочастотного фильтров. Вначале сигнал проходит через низкочастотный фильтр, отсекая высокочастотные компоненты, а затем через высокочастотный, удаляющий низкочастотные составляющие. Таким образом, остаются только частоты, находящиеся в заданном диапазоне.

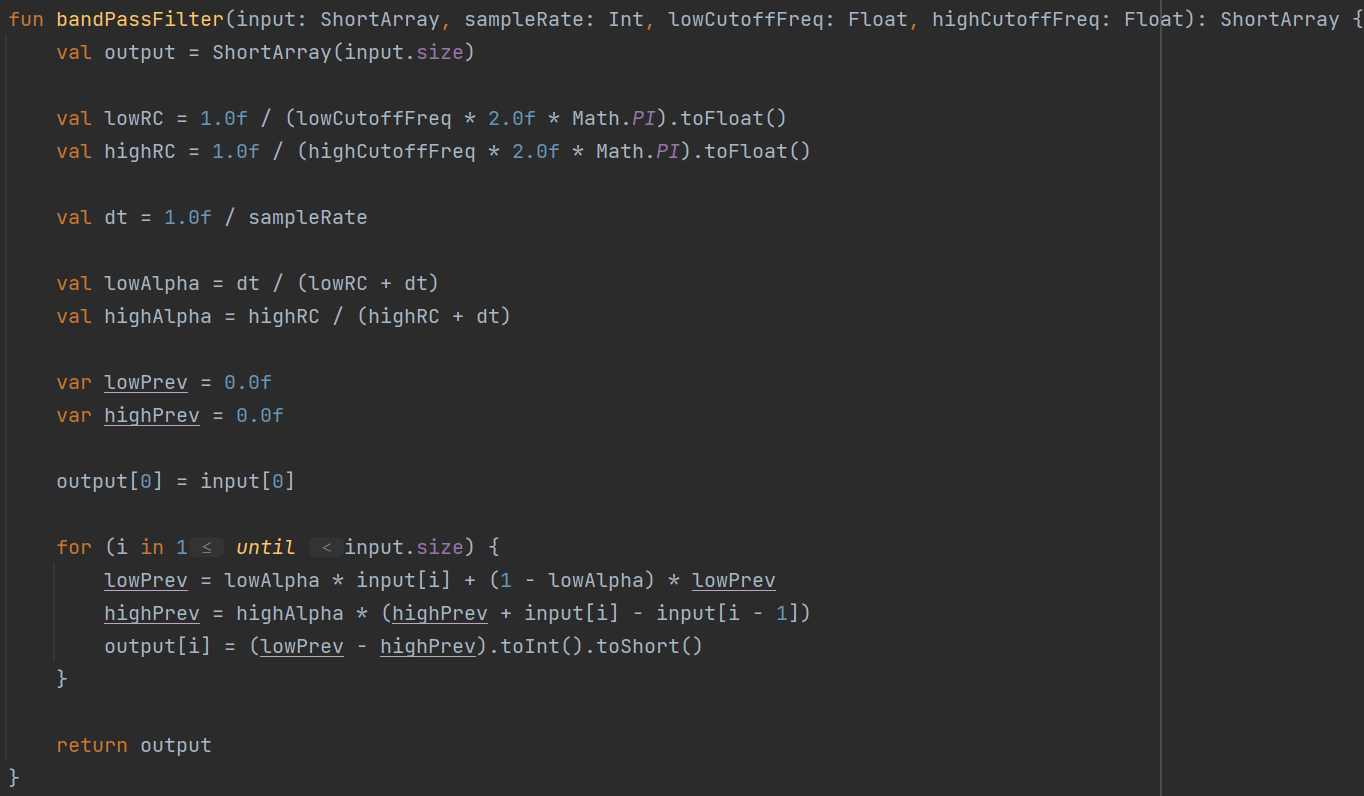


Рисунок 9 – функция полосового фильтра

Метод Калмана используется для сглаживания сигнала и удаления случайных шумов за счет рекурсивной оценки состояния системы. В начале работы задаются параметры ковариации шума процесса и измерений, которые определяют, насколько сильно фильтр будет доверять прогнозу или фактическим данным. Далее на каждом шаге производится предсказание нового значения сигнала на основе предыдущего состояния, а затем корректировка с учетом измеренного значения. Коэффициент Калмана определяет, насколько сильно будет учитываться разница между прогнозируемым и фактическим значением.

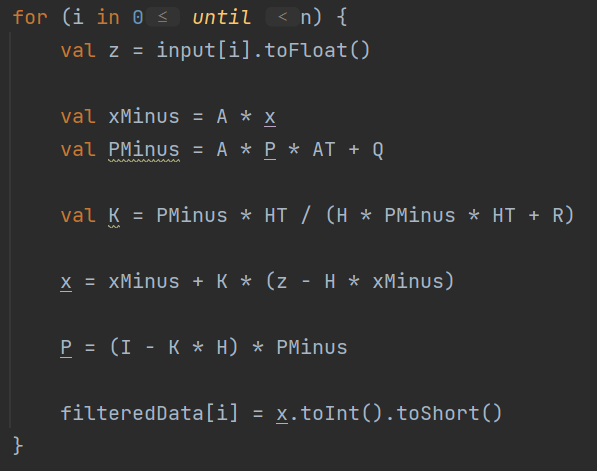


Рисунок 10 – фрагмент функции метода Калмана

Фильтр Гаусса использует свертку сигнала с ядром Гаусса. Вначале формируется одномерный массив коэффициентов, соответствующий функции Гаусса, нормализованный для сохранения амплитуды сигнала. Затем для каждого отсчета аудиоданных применяется скользящее окно, в котором взвешенные значения соседних выборок усредняются, что приводит к плавному сглаживанию сигнала.

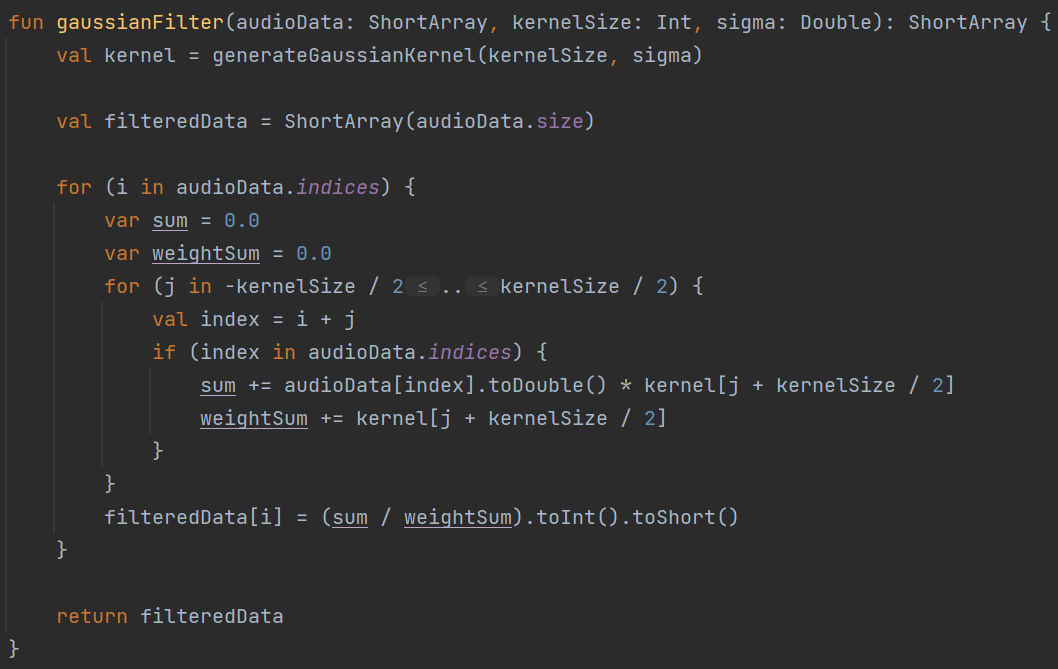


Рисунок 11 – функция фильтра Гаусса

Медианный фильтр применяет окно скользящей медианы, в котором каждый отсчет заменяется на медианное значение из определенного числа соседних отсчетов. Это делает его устойчивым к импульсным шумам и выбросам, так как медиана не подвержена влиянию отдельных аномальных значений. Однако метод требует сортировки значений в окне на каждом шаге, что увеличивает вычислительную сложность, особенно при больших размерах окна.

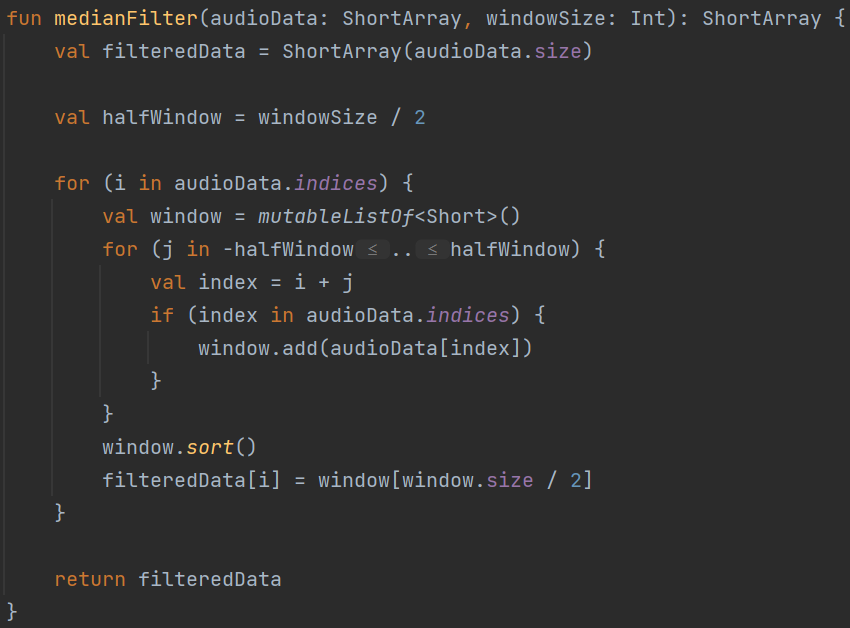


Рисунок 12 – функция медианного фильтра

Метод шумоподавления с использованием спектрального вычитания применяется для удаления фонового шума, анализируя частотный спектр сигнала. Вначале из входного аудиофайла выделяется шумовой профиль – фрагмент, содержащий только шум. Этот профиль разбивается на небольшие окна, для которых вычисляется амплитудный спектр с помощью быстрого преобразования Фурье. Затем для каждого окна сигнала вычисляется его спектр, и из него вычитается средний спектр шума, умноженный на коэффициент подавления. После этого выполняется обратное преобразование Фурье, восстанавливающее сигнал в исходной временной области, но уже с уменьшенным уровнем шума.

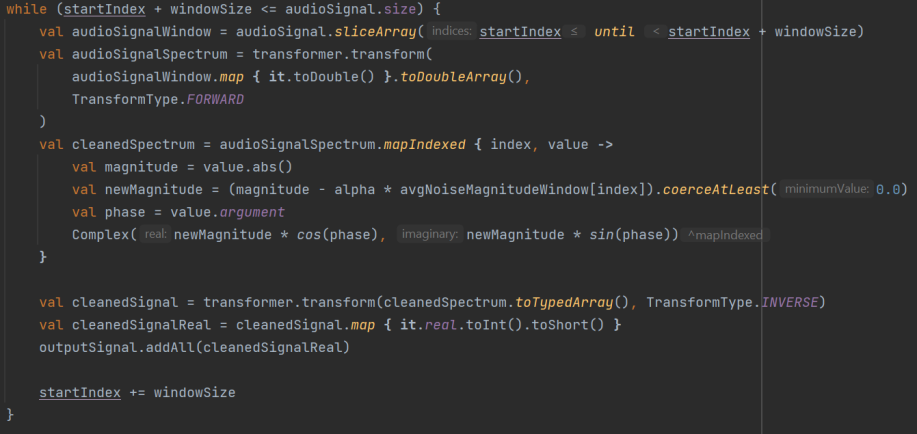


Рисунок 13 – фрагмент функции метода спектрального вычитания

Все методы реализованы в виде отдельных функций, принимающих массив отсчетов сигнала, частоту дискретизации и дополнительные параметры фильтрации. Функции возвращают новый массив с обработанными значениями, сохраняя исходные данные неизменными. Такое разделение позволяет легко комбинировать различные методы и интегрировать их в общую систему обработки аудиофайлов.

Для визуализации аудиосигнала в приложении используется AudioWaveformView, реализованная с помощью Jetpack Compose. Данная Composable-функция принимает массив данных аудиосигнала в формате ShortArray, частоту дискретизации и объект ScrollState, отвечающий за горизонтальную прокрутку. Важным элементом интерфейса является возможность масштабирования отображаемой волновой формы, что реализуется с помощью переменной состояния scale, которая изменяется при взаимодействии со Slider. Пользователь может регулировать масштаб, а текущий коэффициент масштаба отображается в текстовых элементах.

После выбора масштаба происходит расчет ширины Canvas, в котором будет отрисовываться волновая форма. Внутри Canvas происходит отрисовка волновой формы, используя цикл, проходящий по всем точкам ширины Canvas. Для каждой точки выбирается соответствующее значение из массива аудиоданных, которое затем нормализуется в заданный диапазон амплитуд. Это позволяет корректно масштабировать сигнал, чтобы он визуально соответствовал изменяемому пользователем коэффициенту увеличения. Для отображения сигнала используется метод drawLine, который отрисовывает вертикальные линии, соответствующие амплитуде звуковой волны в определенный момент времени.

Прокрутка представлена с помощью Row, в который помещен Canvas, что позволяет пользователю перемещаться по длинному аудиофайлу, если он выходит за пределы экрана.

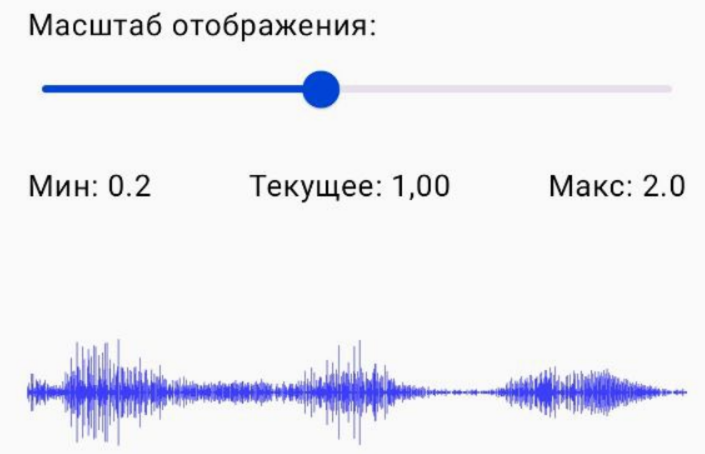


Рисунок 14 – визуализация аудиосигнала



Рисунок 15 – фрагмент функции AudioWaveformView

После запуска приложения пользователю откроется главный экран, на котором будет представлена кнопка для выбора аудиофайла из памяти устройства. После выбора файла появится список доступных методов фильтрации. Пользователь сможет выбрать один или несколько методов и применить их к загруженному аудиофайлу. После обработки для сохранения измененного файла необходимо будет нажать на соответствующую кнопку, после чего файл будет сохранен в память устройства.

1. Тестирование

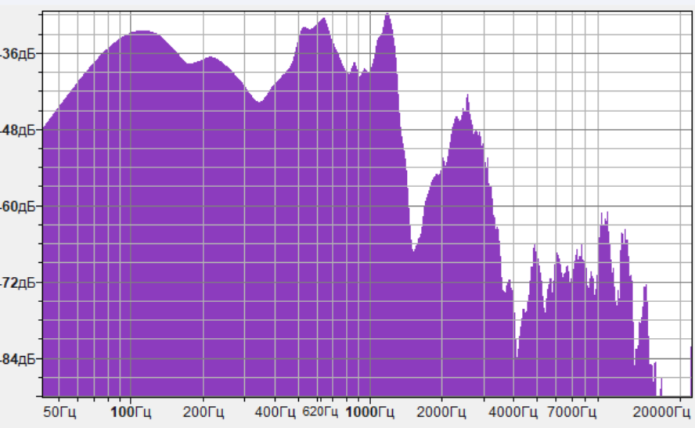
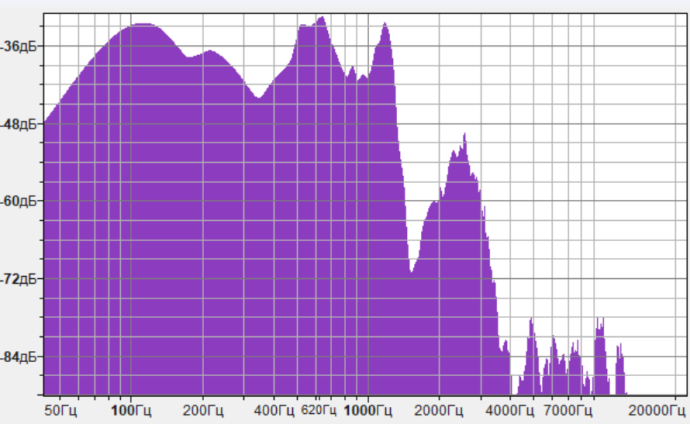
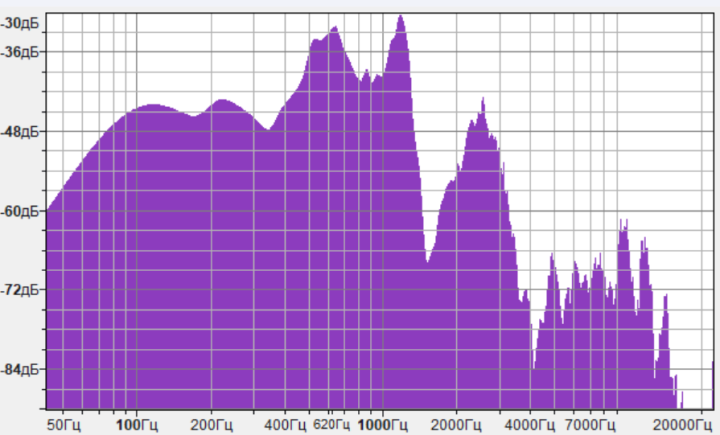


Рисунок 1 – спектрограмма аудиосигнала до применения LPF, HPF и BPF



LPF



HPF

Гауссов фильтр и медианный фильтр используются для подавления шумов, однако работают по-разному и имеют разные области применения.

Для тестирования использован аудиосигнал, содержащий речевой сигнал с добавленным шумом. Шум включает в себя высокочастотные компоненты и импульсные выбросы, что затрудняет разборчивость речи.

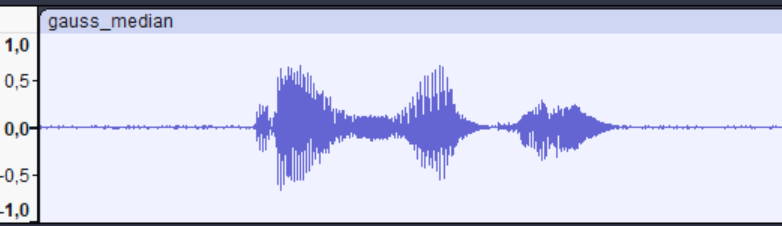


Рисунок 1 – аудиосигнал до применения фильтра Гаусса и медианного фильтра

Фильтр Гаусса выполняет свертку сигнала с гауссовым ядром, что позволяет сгладить шум за счет усреднения значений в скользящем окне. В данном случае применим его со стандартным отклонением, равным 2, и размером скользящего окна, равным 50.

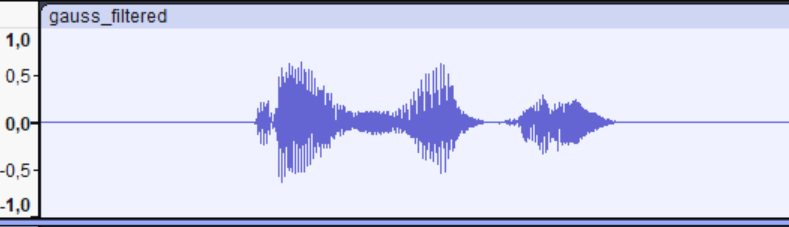


Рисунок 1 – аудиосигнал после применения фильтра Гаусса

Метод хорошо справляется с равномерным шумом, но размывает детали, что может ухудшить четкость речи.

Медианный фильтр заменяет каждый отсчет сигнала медианным значением из скользящего окна. В отличие от Гауссова фильтра, он не размывает резкие переходы.



Рисунок 1 – аудиосигнал после применения медианного фильтра

Медианный фильтр устранил импульсные выбросы, сохранив при этом структуру речевого сигнала.

Последним из реализованных методов фильтрации аудиосигналов является метод шумоподавления с использованием спектрального вычитания.

Для тестирования метода был выбран аудиосигнал, содержащий речевой сигнал с добавленным фоновым шумом. В качестве шума использован белый шум, что создает значительное искажение исходного сигнала.

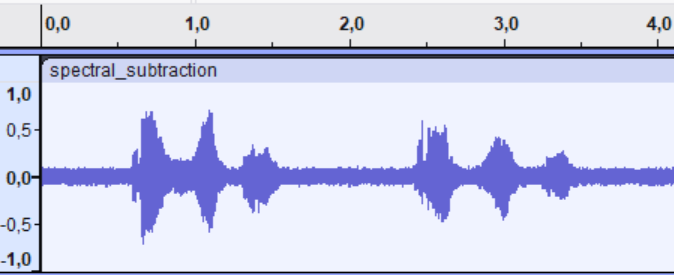


Рисунок 1 – аудиосигнал до применения спектрального вычитания

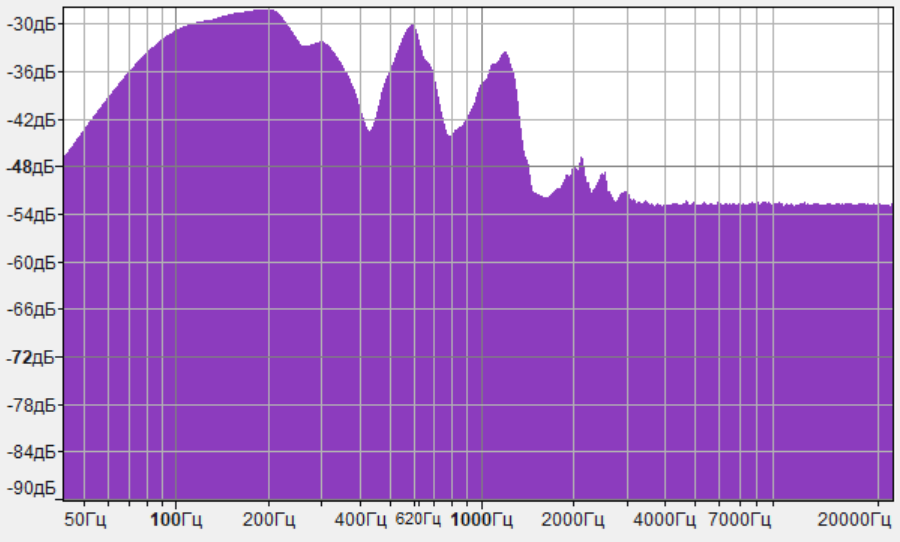


Рисунок 2 – спектрограмма аудиосигнала до применения спектрального вычитания

Для подавления шума был выделен профиль шума в начальном участке сигнала (от 200 до 500 мс), где отсутствует полезная речь. Затем этот профиль использовался для вычисления спектра шума, который впоследствии вычитался из спектра всего сигнала.

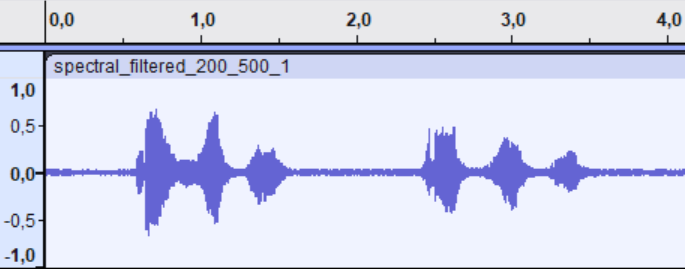


Рисунок 2 – аудиосигнал после применения спектрального вычитания

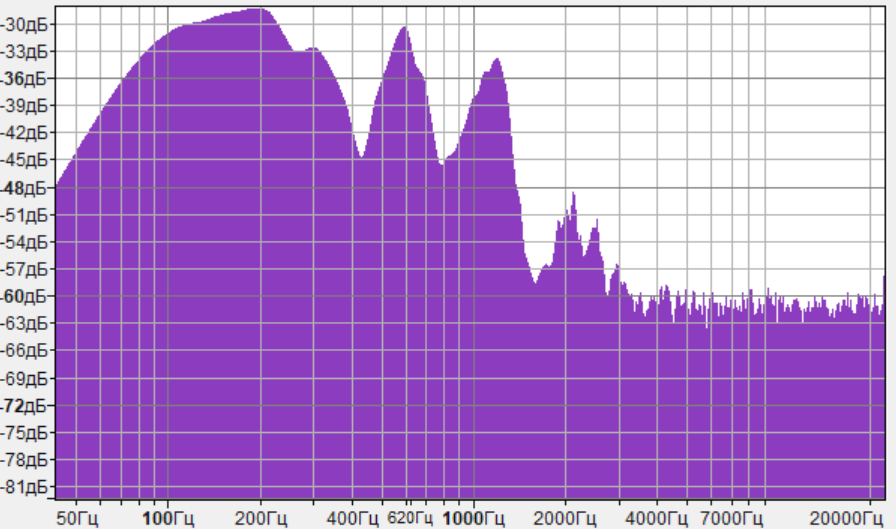


Рисунок 2 – спектрограмма аудиосигнала после применения спектрального вычитания

Как видно, уровень шума значительно снизился, особенно в паузах между словами.

Тестирование показало, что метод спектрального вычитания эффективно подавляет фоновый шум, улучшая разборчивость речи. Однако качество фильтрации зависит от точности оценки шума и выбранных параметров. В будущем можно улучшить метод за счет адаптивной оценки шума или использования нейросетевых алгоритмов.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы были рассмотрены и проанализированы различные методы фильтрации аудиосигналов, их преимущества, недостатки и области применения в мобильных приложениях. Было выявлено, что современные алгоритмы цифровой обработки позволяют значительно улучшить качество звука, снижать уровень шума и повышать разборчивость речи, что особенно важно для голосовых помощников, видеозвонков и музыкальных приложений.

Практическая часть работы включала разработку и реализацию фильтров с использованием языка программирования Kotlin. В результате были созданы эффективные алгоритмы обработки звука, интегрированные в мобильное приложение с удобным пользовательским интерфейсом. Проведенное тестирование подтвердило эффективность выбранных методов, продемонстрировав улучшение качества аудиосигналов в различных условиях.

Таким образом, поставленные цели и задачи были успешно выполнены. Разработка методов фильтрации аудиосигналов для мобильных приложений является перспективным направлением, открывающим новые возможности для повышения удобства и качества взаимодействия пользователей с цифровыми устройствами. Дальнейшее совершенствование алгоритмов, использование машинного обучения и адаптивных систем обработки звука позволят создавать еще более интеллектуальные и эффективные решения для работы с аудиосигналами.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Advanced Digital Signal Processing and Noise Reduction, Second Edition, Saeed V. Vaseghi, 2000
2. Айфичер Э., Джервис Б. Цифровая обработка сигналов. Практический подход, Изд. дом «Вильямс», 2004
3. Бендатт Дж., Пирсон А. Применения корреляционного и спектрального анализа, Мир, 1983
4. Интернет ресурс: https://developer.android.com