EDUARDO TADASHI ASATO
THIAGO YUKIO NAGATOMO RODRIGUES DE ASSIS
WILLIAM HIDEKI MIYANO

APLICATIVO PARA COMPENSAÇÃO DE DEFICIÊNCIAS AUDITIVAS

São Paulo
2022

EDUARDO TADASHI ASATO
THIAGO YUKIO NAGATOMO RODRIGUES DE ASSIS
WILLIAM HIDEKI MIYANO

APLICATIVO PARA COMPENSAÇÃO DE DEFICIÊNCIAS AUDITIVAS

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro Eletricista com ênfase em Engenharia de Telecomunicações.

São Paulo
2022

EDUARDO TADASHI ASATO
THIAGO YUKIO NAGATOMO RODRIGUES DE ASSIS
WILLIAM HIDEKI MIYANO

APLICATIVO PARA COMPENSAÇÃO DE DEFICIÊNCIAS AUDITIVAS

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro Eletricista com ênfase em Engenharia de Telecomunicações.

Área de Concentração:
Filtragem de áudio em tempo real

Orientador:
Phillip Burt

Co-orientador:
Ricardo Luis de Azevedo da Rocha

São Paulo
2022

RESUMO

Temas relacionados à inclusão social têm entrado em voga nos últimos tempos, entre os quais está a acessibilidade, inclusive de pessoas com perda auditiva. Além disto, outra tendência é a do uso de aparelhos celular, que têm se tornado mais comum na vida cotidiana, permitindo um alto nível de conectividade. No entanto, para aqueles que sofrem de perda auditiva, o acesso ao uso do celular pode ser dificultado, pois a interface sonora é a uma das principais do aparelho.

A presbiacusia, ou perda de audição relacionada à idade, tem se tornado cada vez mais comum devido ao aumento da expectativa de vida mundial, o que torna a necessidade de ferramentas de amplificação do som ainda mais urgente.

O aplicativa de compensação de deficiências auditivas se apresenta como uma solução viável para esse problema, pois ele permite que usuários com perda auditiva ajustem a intensidade das frequências do som de acordo com as suas necessidades. Isso pode ser uma alternativa mais simples e acessível do que próteses auditivas, que costumam ser caras e incômodas para uso constante. Isso pode melhorar significativamente a qualidade de vida dessas pessoas e contribuir para sua inclusão social.

Palavras-Chave – Software, Acessibilidade, Som, Áudio, Aplicativo, Móvel, Tempo Real, Processamento.

ABSTRACT

Issues relating to social inclusion have become incresingly popular in recent times, one of which is accessibility, including for people with hearing loss. In addition, another trend is the use of cell phones, which have become more common in everyday life, allowing for a high degree of connectivity. However, for those who suffer from hearing loss, the experience of using a cell phone may be difficult, as the audio interface is one of the main ones of the device.

Presbycusis, or age-related hearing loss, has become increasingly common due to the increase in global life expectancy, making the need for sound amplification tools even more urgent.

The hearing loss compensation application presents itself as a viable solution to this problem, as it allows users with hearing loss to adjust the intensity of sound frequencies according to their needs. This is a more simple and acessible solution in comparison to conventional hearing aids, which are often expensive and uncomfortable for constant use. This can significantly improve the quality of life of these people and contribute to their social inclusion.

Keywords – Software, Accessibility, Sound, Audio, Application, Realtime, Mobile, Processing.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação

1.1.1 Próteses auditivas

As próteses auditivas são dispositivos elétricos que consistem em três partes principais: um microfone, que faz a captação do áudio externo; um circuito elétrico, que faz uma transformação sobre esse sinal sonoro, geralmente o amplificando; e um transdutor, que retransmite o sinal transformado para a orelha interna, via conduto externo ou transmissão óssea. [1]

Existem vários tipos de próteses auditivas no mercado: as convencionais, que foram as primeiras a surgir; as retroauriculares e intra-retroauriculares, que surgiram depois, que trouxeram a inovação de não necessitarem que o usuário carregue uma caixa no bolso com o circuito amplificador, o mesmo ficava todo na orelha do usuário; e os de canal, que são ainda mais miniaturizados, ficam totalmente dentro do canal auditivo. [1][2]

As próteses também podem classificadas segundo a tecnologia do circuito elétrico, que pode ser analógica, que vêm sendo utilizadas há mais tempo, elas têm a vantagem de serem mais baratas, mas têm a desvantagem de serem menos flexíveis, já que sua programação é limitada, possuindo apenas um ajuste de calibração; e as digitais, que são mais caras, mas têm muito mais flexibilidade do processamento do sinal do microfone, podendo adaptar a transformação para cada situação, e alterações no perfil de audição do usuário. [2]

Um problema que afeta todas as próteses auditivas é o seu elevado preço, sendo acima de R$ 3500 reais, incluindo o dispositivo em si, mas também a audiometria, as consultas médicas e a fabricação do molde, este preço, claro, aumenta rapidamente quanto mais avançada for a tecnologia do dispositivo. [3] Além disto, eles podem sofrer interferência eletromagnética de celulares, impedindo o uso em conjunto dos dois. [2]

Os custos recorrentes a se levar em conta são a manutenção do dispositivo, que pode

incluir mais consultas, e a da troca das baterias. Todos estes custos levam muitos a desistir da reabilitação auditiva, e apenas continuam com uma qualidade de vida reduzida. [3]

1.1.2 Telefone com compensação auditiva

Para o caso de uso de telefone, uma alternativa ao aparelho auditivo é o de um circuito que intercepta os sinais vindos da linha telefônica, e aplica uma transformação nos mesmos, de forma a compensar a perda auditiva do usuário. Foi exatamente esta proposta que foi apresentada na referência [4], na qual um equipamento que implementa processamento digital de sinais foi proposto, que iria fazer a filtragem de sinal com frequências de corte continuamente variáveis, compressão de faixa dinâmica, replicação espectral e translação espectral, funcionalidades que não estão presentes nos aparelhos auditivos mais simples.

Esta proposta tem a vantagem de trazer um ganho de qualidade de vida muito maior que um simples amplificador não-seletivo, mas não foi feita para usar com um aparelho celular, meio pelo qual muitas pessoas se comunicam atualmente.

1.1.3 Aplicativo para compensação auditiva

Os custos associados aos aparelhos auditivos, e a proposta de telefone com compensação auditiva, levaram à concepção do projeto do aplicativo para celular, aqui descrito.

1.2 Justificativa

Segundo a referência [5]: “A contratação de pessoas com deficiência (PCDs) aumentou 147 % até agosto deste ano [2021] em relação ao mesmo período de 2020. Os dados são da Page PCD, consultoria especializada no recrutamento de pessoas com deficiência [...]”, ela é apenas um dos exemplos da tendência de crescimento no interesse em questões de inclusão de pessoas com deficiência nas discussões no meio do desenvolvimento. Busca-se permitir que a maior parte possível de usuários que desejem e necessitem de serviços disponibilizados sejam capazes de utilizá-los.

Uma das deficiências com maior notoriedade é a auditiva, pois, segundo a referência [6], em 2019 ela afetava mais de dez milhões de brasileiros, no mundo essa número chegou a 1,5 bilhões, cerca de 20 % da população mundial.

Outra tendência que vem aumentando nos últimos anos é o uso do celular: de acordo

com um levantamento pela consultoria da plataforma AppAnnie, em 2021 usuários brasileiros passaram, em média, 5,4 horas por dia no celular [7]. Metade da população mundial possui um *smartphone*, de acordo com dados de 2021 da Strategy Analytics,[8] e isso se reflete, por exemplo, no ato de ouvir música: 27 % do tempo em que se ouve música é em *smartphones* [9]. Isso implica numa limitação de usabilidade para considerável parcela da população, visto que fones de ouvido intra-auriculares são incompatíveis com aparelhos auditivos de determinados tipos. Os usuários se limitam a utilizar caixas de som ou fones maiores, muitas vezes inconvenientes.

Como uma das principais interfaces humano-computador de um celular é através de sinais sonoros, uma deficiência auditiva resulta em uma diminuição significativa na qualidade de experiência de usuários que a possuem. Em vista deste fato, e seguindo ambas as tendências detalhadas anteriormente, este projeto foi criado com o intuito de melhorar a acessibilidade de pessoas com deficiência auditiva leve ou moderada no celular, através de uma solução simples, eficiente e barata.

Trata-se de uma demanda não atendida pelo mercado de software, principalmente em um contexto nacional. Com isso, busca-se iniciar um processo possivelmente contínuo com possibilidade de expansão futura.

1.3 Objetivo

Este projeto tem por finalidade o desenvolvimento de uma aplicação para aparelhos móveis destinada a auxiliar indivíduos portadores de deficiências auditivas leves e moderadas, utilizando tecnologias de Processamento Digital de Sinais em tempo real.

Haverá um modo de calibração, para que os parâmetros de compensação sejam personalizados de acordo com a condição de perda auditiva de cada usuário, e o modo de execução, que vai filtrar e amplificar uma fonte de áudio do aparelho móvel.

Trata-se de uma prova de conceito, com a exploração de tecnologias e estratégias para o desenvolvimento deste tipo de aplicação. Posteriormente são descritas perspectivas de melhora e expansão do projeto, bem como experiências relevantes durante o desenvolvimento deste projeto.

2 ASPECTOS CONCEITUAIS

Neste capítulo serão analisados aspectos conceituais importantes para o desenvolvimento do projeto. Como os tipos de perdas auditivas, a percepção de frequências no ouvido humano, o limiar de audibilidade e o limiar de dor e o audiograma

2.0.1 Perdas Auditivas

As principais causas de deficiência auditiva são: presbiacusia (perda auditiva progressiva, decorrente da idade), surdez induzida por ruído, doença de Meniére (perda sensorioneural coclear, flutuante), otite média (processo inflamatório da orelha média) e otospongiose (invasão óssea no ouvido médio). Outras causas podem ser infecções virais (sarampo, caxumba), traumas crânio-encefálicos, exposição à radiação, drogas ototóxicas (certos antibióticos, diuréticos, anti-inflamatórios e anti-neoplásicos), distúrbios metabólicos e doenças auto-imunes. A incidência de deficiências auditivas congênitas varia de 1 a 3 casos por 1000 habitantes.

A criança e o adolescente podem ser acometidos de deficiências auditivas, nos mais variáveis níveis de perda, que comprometem, principalmente, seu nível de aprendizado.

Podem prejudicar especialmente o desenvolvimento da fala; é necessário ouvir a linguagem para poder aprendê-la.

A chamada Presbiacusia é a perda auditiva que atinge o idoso (principalmente a partir dos 60 anos), e decorre de várias causas. É uma patologia comum, que prejudica o desenvolvimento de todas as atividades sociais e profissionais de seu portador, à medida que progride, levando-o ao isolamento, depressão e riscos de vida pela dificuldade de ouvir. Ocasiona envelhecimento precoce, levando a desinteresse pela vida. O Relatório Mundial sobre Audição de 2021, da Organização Mundial da Saúde [10] mostra a distribuição da perda auditiva em função da idade, visto na Fig 1

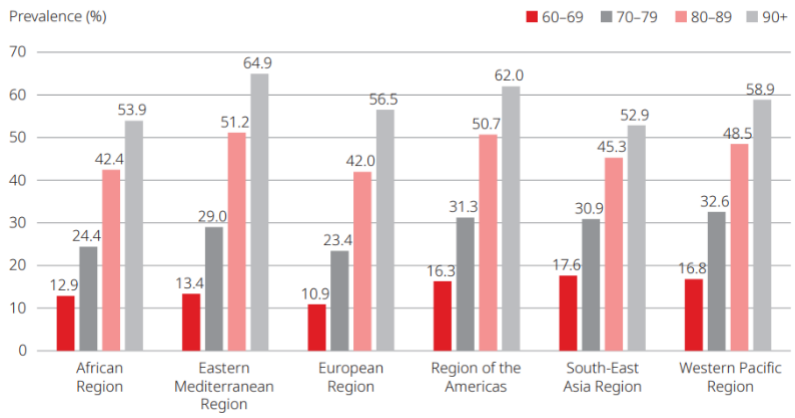

Figura 1: Perda Auditiva e idades

Segundo dados do IBGE a população de idosos no Brasil vem aumentando: entre 1991 e 2000, a parcela de idosos evoluiu de 7,3

A Fundação Otorrinolaringologia, no ano de 2003, estudou 2 grupos de idosos. No primeiro, junto ao Grupo de Atendimento Multidisciplinar do Idoso Ambulatorial – GAMIA (da Disciplina de Geriatria do Hospital das Clínicas da Universidade de São Paulo), dentre 85 pacientes atendidos, com idade entre 68 e 91, foram encontrados 61 com presbiacusia. Destes, 23 receberam indicação de uso de Aparelho de Amplificação Sonora Individual (AASI). No segundo grupo, na Favela de Paraisópolis, entre 44 indivíduos atendidos, com idade variando entre 56 e 84 anos, 27 pacientes apresentavam Presbiacusia e 6 receberam indicação de AASI. Os indivíduos com Presbiacusia que não receberam indicação de uso da prótese auditiva (60 % a 70 % dos casos estudados) têm prejuízo na comunicação, e sua queixa principal é: "eu escuto mas não entendo". Para eles, o nível de audição encontra-se dentro dos padrões da normalidade para as frequências mais baixas , entre 500 e 3000 Hz, e a perda auditiva maior ocorre em geral para as frequências mais altas.

Já a Perda Auditiva Induzida pelo Ruído, segunda causa de surdez no Brasil, compromete principalmente o adulto jovem. Dependendo da causa, pode manifestar-se como perda em uma faixa restrita de frequências. Nem sempre é possível boa adaptação do Aparelho de Amplificação Sonora Individual, levando a grandes prejuízos sociais, familiares e de produtividade.

Geralmente, o indivíduo portador de uma deficiência auditiva só a percebe quando ela passa a interferir no processo de comunicação, comprometendo a inteligibilidade de mensagens, como pelo fato de formular respostas erradas às questões que lhe foram dirigidas. Como consequência, o paciente tende a isolar-se, afastando-se cada vez mais de situações

de comunicação (Radini, 1994).

2.0.2 Classificação das Perdas Auditivas

A perda auditiva pode ser classificada da seguinte forma, de acordo com sua gravidade [**??**]:

- Perda leve – pequena dificuldade para entender a fala; ouve bem em ambientes silenciosos, mas tem dificuldade em ambientes ruidosos; dificuldade para vozes distantes ou muito baixas (sussurradas).
- Perda moderada – dificuldade para entender a fala mesmo em ambientes silenciosos; pede constantemente para repetir o que foi dito; o interlocutor precisa falar mais alto.
- Perda moderadamente severa – comunica-se com muita dificuldade em todas as situações; consegue entender a fala apenas quando os falantes estão muito próximos.
- Perda severa – muita dificuldade para entender a fala; consegue identificar ruídos ambientais; consegue perceber as vogais mas não entende as consoantes.
- Perda profunda – consegue perceber apenas sons muito elevados.

2.1 Percepção de Som

A estrutura do ouvido humano pode ser dividida em 3 partes: ouvido externo, ouvido médio e ouvindo interno, sendo este o que será melhor detalhado adiante. O ouvido interno é composto basicamente da cóclea, uma cavidade cônica, enrolada na forma de caracol, preenchida por um meio aquoso. A cóclea é dividida longitudinalmente pela membrana basilar, onde estão distribuídos os Orgãos de Corti.

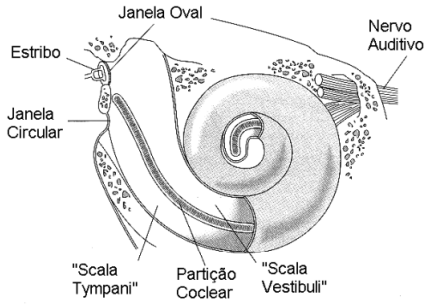

Figura 2: Esquema da cóclea

Os Orgãos de Corti são como sensores de vibração ligados à terminações nervosas. Esses sensores ressoam em determinadas frequências, espaçadas não linearmente. A cóclea, então, se assemelha à um banco de filtros seletivos distribuídos, capaz de distinguir sons na faixa de 20 Hz a 20 kHz, sendo que as baixas e altas frequências ressoam em extremidades opostas.

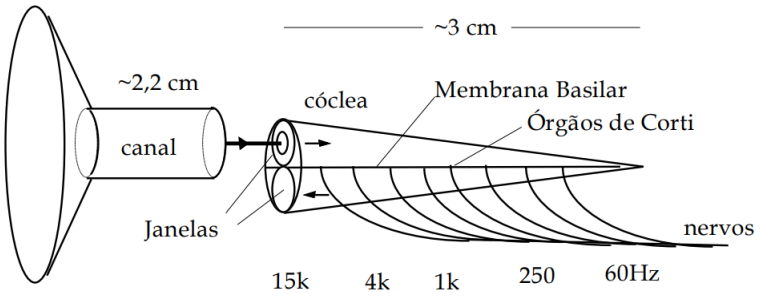

Figura 3: Seletividade de frequências

2.1.1 Diferença apenas perceptível de frequências (DAP)

Em resultado da seletividade não linear do ouvido, o ouvido humano responde de forma diferente com relação à percepção de variação de frequências. A diferença apenas perceptível de frequências é o menor valor, para uma variação de frequência, em que é possível perceber diferença no som. No ouvido humano, quando maior é a frequência, maior é a diferença apenas perceptível. Essa relação se dá de forma aproximada com: $DAP = 0.007f$.

2.2 Limiar de Audibilidade

O limiar de audibilidade, é a curva que define o limite da intensidade do som em que ainda é possível escutar o som. Esse limiar pode ser observado na curva de igual volume sonoro para tons senoidais.

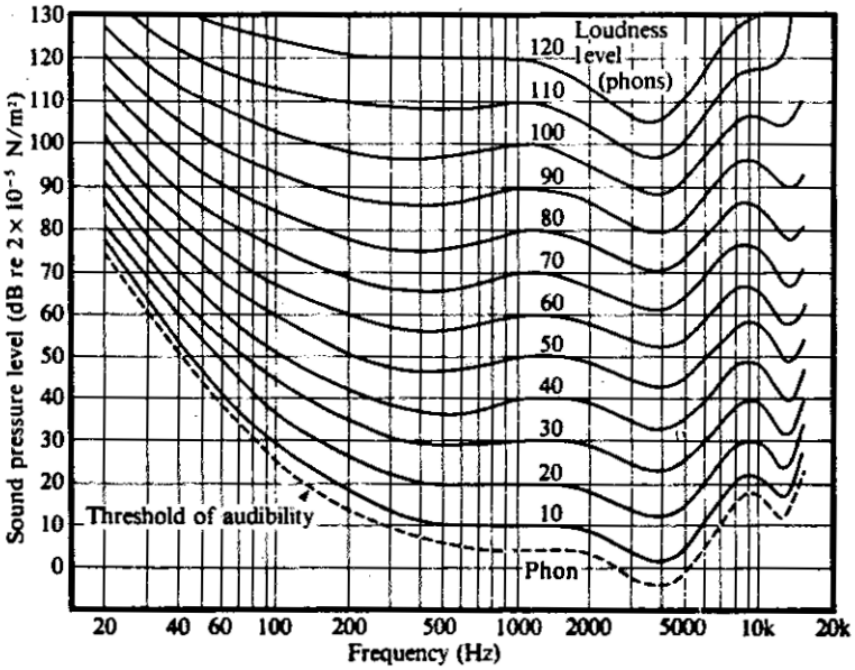

Figura 4: Curva de Igual Volume Sonoro para tons senoidais

A grandeza subjetiva associada à sensação de volume sonoro é medido em Phons. O eixo das ordenadas do gráfico é medido em pressão sonora, db SPL (Sound Pressure Level). A sensibilidade se torna bastante reduzida para baixas frequências, rejeitando sons produzidos pelo próprio corpo. A sensibilidade atinge um pico em torno de 4 kHz e volta a decrescer para frequências mais altas.

2.3 Audiograma

O Audiograma é um gráfico que descreve a capacidade e sensibilidade auditiva de uma pessoa em função da frequência. O gráfico é medido em dB HL (dB hearing loss), ou perda auditiva em dB. O audiograma geralmente é medido separadamente para o ouvido esquerdo e direito. O gráfico é construído através de valores discretos de frequências, onde é medido a perda auditiva naquela frequência.

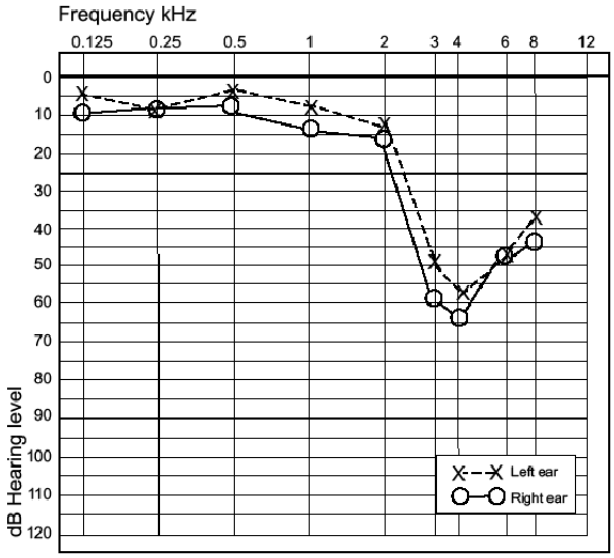

Figura 5: Exemplo de audiograma

A partir do gráfico de limiar de audibilidade para um ouvido típico, é possível obter a curva do limiar de audibilidade para o ouvido do usuário, somando a perda auditiva ao limiar. Estes dados servem como base para calcular o fator de amplificação das amostras.

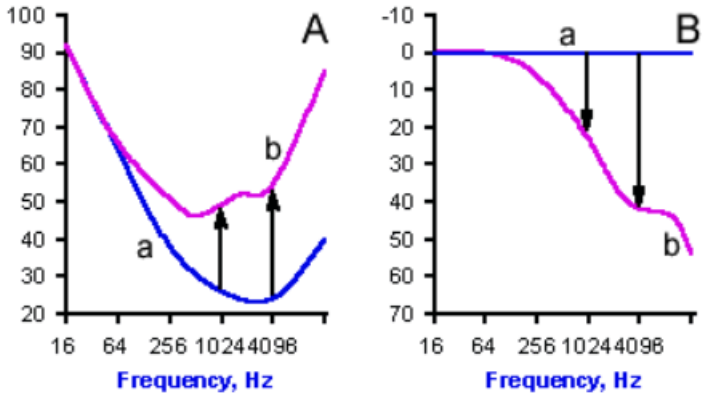

Figura 6: Perda auditiva e limiar de audibilidade

2.4 Filtro Digital

Um filtro é um circuito para processamento de sinal, que pode atenuar componentes indesejadas de certas frequências, ou amplificar outras componentes. O filtro digital utiliza atrasos, somadores e multiplicações por fatores para realizar a filtragem de sinais digitais.

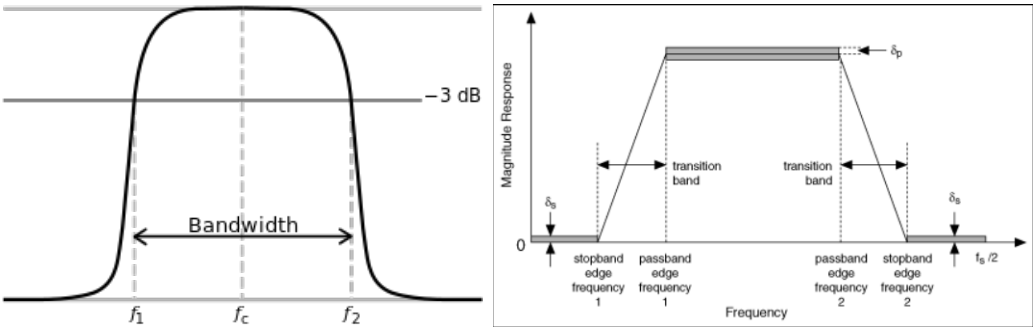

Figura 7: Filtro passa-faixas e máscara

Um filtro passa-faixas pode ser definido por sua amplitude e as frequências de corte, que correspondem às frequências em que a amplitude cai à 3 dB do valor máximo. Mas além dessas características, é preciso levar em conta outros aspectos importantes quando se vai projetar um filtro. Para planejar o filtro, deve-se levar em conta a máscara, que define as especificações desejadas para o filtro projetado.

Na máscara do filtro são definidos a banda de passagem, onde se deve manter ou amplificar o sinal; a banda de rejeição, onde se deve atenuar o sinal; a faixa de transição, área entre a banda de passagem e rejeição; os valores máximos de ripple na faixa de passagem e na faixa de rejeição (relacionados a δ). Para atender esses requisitos, várias técnicas de projeto de filtro existem e podem ser aplicadas.

3 METODOLOGIA DE TRABALHO

3.1 Etapas do desenvolvimento

A partir da definição inicial dos objetivos do desenvolvimento, tem-se uma análise profunda tanto do estado da arte no meio, tecnologias disponíveis, contexto e planejamento. Em todas as etapas do desenvolvimento, requisitos, metodologias, tecnologias e objetivos podem ser alterados e lapidados para melhor atender ao projeto final, porém inicialmente foram estabelecidos requisitos e marcos para guiar o trabalho.

Para haver uma progressão visível e melhor se adequar a modelos contemporâneos de desenvolvimento, o trabalho se dividirá em ciclos de produção, com produtos parciais a cada um desses ciclos; inicia-se com simples testes iniciais, para verificar-se a viabilidade de conceitos.

A partir de então, novas funcionalidades são inseridas para avançar pouco a pouco para um resultado final.

3.1.1 Escopos Planejados

Podem ser definidas alguns pontos para nortear o desenvolvimento:

- Teste do processamento em MATLAB
- Aplicação inicial vazia
- Simples reprodutor de arquivos de áudio
- Possibilidade de filtragem simples
- Implementação de filtragem por bandas
- Input personalizado de configurações
- Obtenção e processamento de sinal geral do aparelho

Serão feitos testes intermediários, porém no fim devem se buscar voluntários para um teste controlado. Para isso, devem ser seguidas normas de testagem às cegas estabelecidas.

4 ESPECIFICAÇÃO DE REQUISITOS

4.1 Persona

Para guiar o desenvolvimento, é válido estabelecer uma persona, construto que representa uma gama considerável dos usuários de forma mais concreta, que facilita decisões de projeto.

> Cleusa é uma mulher aposentada, que utiliza seu celular principalmente para se comunicar com sua família e consumir conteúdos de mídia. Seu conhecimento tecnológico é limitado às funcionalidades básicas e mais corriqueiras desses aplicativos.
> Nos últimos anos, começou a apresentar dificuldades de entender áudios enviados nos grupos, ligações e os programas que assiste.

4.2 Requisitos do Sistema

Os requisitos de sistema dependem da descoberta de viabilidade de cada caminho possível. Estes caminhos e seus requisitos estão detalhados abaixo.

4.2.1 Aplicação que intercepta a saída de áudio do aparelho móvel

Este é o fluxo ideal para o projeto, ele consiste em desenvolver uma aplicação que intercepta a saída de áudio do celular, aplica uma transformação sobre esse sinal, calibrada para a perda auditiva do usuário, e repassa o mesmo para o dispositivo de saída de áudio em tempo real. Seus requisitos funcionais podem ser encontrados na listagem abaixo:

Requisito funcional 1

A aplicação deve interceptar o sinal de áudio antes dele ser reproduzido no transdu-

tor, sendo o sinal de aplicação específica (como um reprodutor de músicas, telefone, etc) ou o *mix* de áudio do sistema.

Requisito funcional 2

A aplicação deve possuir um modo de calibração, na qual um audiograma do usuário é levantado, seguindo o método utilizado nos exames médicos: o de usar tons puros de diferentes frequências e amplitudes para determinar o limiar de audição para cada frequência.

Requisito funcional 3

Passar o sinal de áudio por um equalizador não-linear cujos parâmetros são definidos pelo audiograma. O número de canais deste equalizador depende do modelo de celular.

Requisito funcional 4

Deve haver compressão de faixa dinâmica, de forma que o limiar da dor do usuário não seja atingido, e para produzir uma sensação auditiva semelhante à típica, quando uma das bandas do equalizador é amplificada.

4.2.2 Aplicação que faz processamento de arquivos de áudio

Este é o primeiro fluxo alternativo do projeto, caso o primeiro se mostre inviável, neste caso o aplicativo não iria interceptar o áudio do celular, apenas iria aplicar o processamento sobre um arquivo de áudio no armazenamento interno do aparelho móvel. Como não seria necessário depender de uma API do sistema operacional, seria possível processar cada amostra individual, então os seguintes requisitos funcionais poderiam ser levantados:

Requisito funcional 1

A aplicação deve processar as amostras de áudio contidas em um arquivo no armazenamento interno do dispositivo, enquanto vai enviando o resultado para a saída de áudio.

Requisito funcional 2

Se mantém o mesmo do fluxo principal.

Requisito funcional 3

Se mantém o mesmo do fluxo principal.

Requisito funcional 4

Se mantém o mesmo do fluxo principal.

Nesse caso, o seguinte diagrama de sequência representa o sistema:

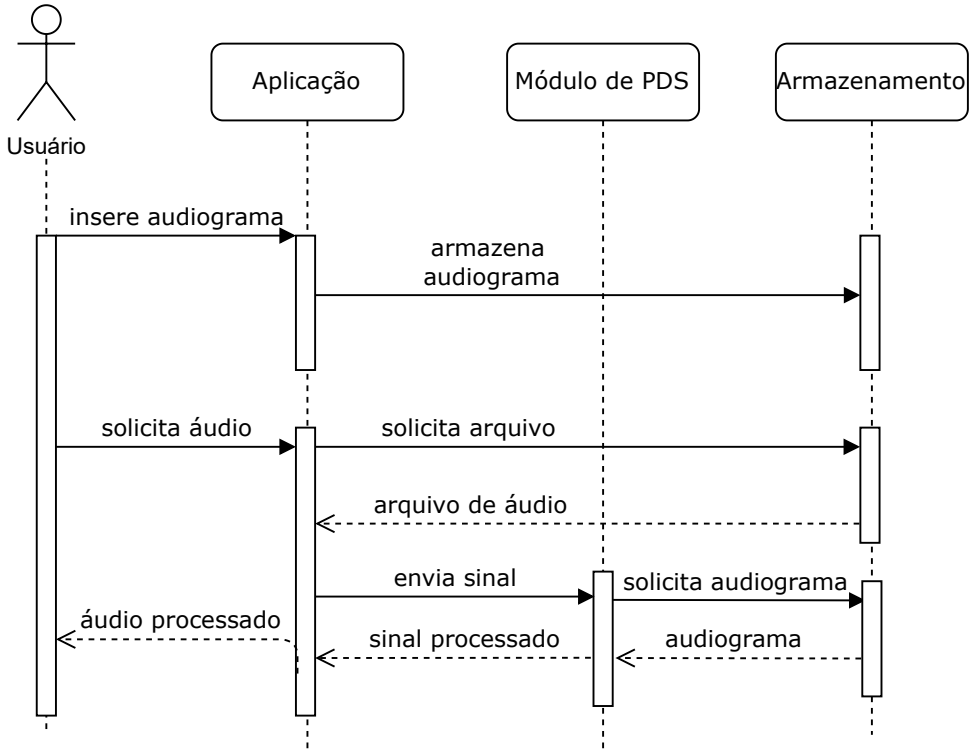

Figura 8: Diagrama de sequência

4.2.3 Chatbot de aplicativo de mensagens

Este é o segundo fluxo alternativo, caso nenhum dos fluxos anteriores se mostre viável. Neste caso uma aplicação backend iria ser servida por um frontend utilizando a própria interface de algum aplicativo de mensagens instantâneas. O único requisito funcional que seria anterado em relação ao fluxo anterior é que, agora, os arquivos de áudio de entrada seriam mensagens de áudio encaminhadas para o chatbot.

5 DESENVOLVIMENTO DO TRABALHO

5.1 Tecnologias Utilizadas

A abordagem pode seguir diferentes rumos, a depender das limitações técnicas apresentadas pelo sistema operacional e plataformas de desenvolvimento.

5.1.1 Aplicativo Android

A trajetória ideal e esperada visa utilizar Android Studio, em linguagem Java para desenvolver uma aplicação Android, de forma a interceptar o áudio de saída, realizar um processamento digital, e repassar para o dispositivo de saída de áudio, em tempo real. Seria tomado como base a classe Equalizer e sua classe-mãe AudioEffect para executar essa captura. Também há a possibilidade de utilizar AudioCapture A partir desse ponto, o áudio seria processado utilizando bibliotecas de transformações de sinais, a partir das técnicas acima.

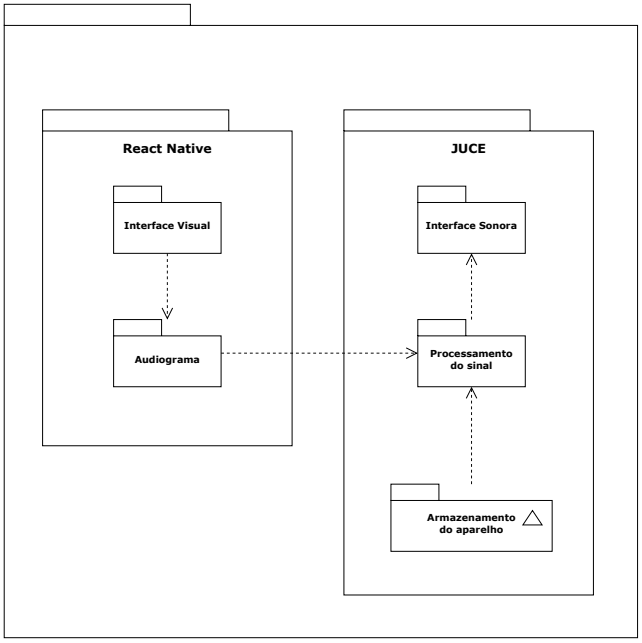

Figura 9: Diagrama para aplicação de multas

- Metodologias ágeis de projeto, como Scrum e Kanban
- Técnicas de criatividade
- Comportamento no trabalho
- Gestão de mudanças
- Visão estratégica

5.1.2 Aplicativo alternativo

Simplificando o escopo, existe a possibilidade de reduzir o processamento a arquivos de áudio, não necessitando a utilização de captura de áudio. Dessa forma, para agilizar o desenvolvimento, pode ser utilizado frameworks como React Native, com o qual os integrantes têm mais familiaridade.

5.1.3 Chatbot de whatsapp

Uma outra alternativa é a criação de uma interface baseada em outra já existente e comummente utilizada pelo público alvo: aplicativos de mensagens instantâneas. Dessa forma, através da API de Chatbot para Empresas, é possível executar um processamento server-side, possibilitando um desenvolvimento em python ou outras linguagens mais familiares.

5.2 Projeto e Implementação

5.2.1 Escolha de caminho de projeto

O caminho ideal, o correspondente à "Aplicação que intercepta a saída de áudio do aparelho móvel", não foi escolhido por uma série de razões, as quais estão listadas abaixo:

- Falta de programabilidade: a interface oferecida pela API `Equalizer`[11] da SDK do Android, que apenas apresenta o método `setBandLevel(short band, short level)`, sem apresentar uma forma de definir o funcionamento do algoritmo de filtragem de cada banda do equalizador, e sem garantia de compressão de faixa dinâmica;
- número de bandas do equalizador depende do dispositivo[12];
- a funcionalidade de aplicar o efeito de áudio sobre o *mix* principal do sistema está *deprecated*[11].

Logo, a segunda opção foi escolhida, "Aplicação que faz processamento de arquivos de áudio", que é uma aplicação que realiza processamento digital sobre amostras em um arquivo de áudio em tempo real, de forma que o áudio possa começar a ser tocado com o mínimo de atraso.

5.2.2 Fases do Processo do Desenvolvimento de Software

5.2.2.1 *JUCE Framework*

Existem muitas formas de criar um aplicativo Android: pode ser usando diretamente a SDK do Android, ou, outra possibilidade, usar um framework como o *JUCE Framework*, que pode exportar o projeto para diversas plataformas, como Windows, macOS, Linux, iOS e Android, que inclui bibliotecas de interface de usuário, processamento de áudio, manipulação de arquivos etc[13]

Ela foi explorada como primeira escolha de framework, e, em um primeiro momento, se desenvolveu somente no computador.

Ao final desta fase, foi criada uma prova de conceito de que é possível obter as amostras de áudio de um arquivo no armazenamento do computador dentro do programa, para assim efetuar algum processamento.

Porém, ao exportar o projeto para *Android*, e abri-lo no emulador do *Android Studio*, o seletor de arquivos não funcionou, o que provavelmente implicaria que alterações precisariam ser feitas no código fonte gerado. Então a equipe cessou em continuar nesta direção para o projeto, em favor a programar diretamente no *Android Studio*, já que isto seria necessário de qualquer forma.

5.2.2.2 Seleção de Arquivo

Com o novo rumo do projeto definido, começando o mesmo já no Android Studio, primeiro foi implementado um botão no aplicativo que chamava a atividade correspondente ao seletor nativo de arquivos do Android.

Como não foi necessário implementar esta funcionalidade do zero, houve grande ganho de tempo.

5.2.2.3 Tocando Arquivos de Som

Foram exploradas diversas técnicas para a reprodução de áudio, buscando atingir um equilíbrio adequado entre simplicidade de implementação e capacidade de customização do funcionamento para os processamentos futuros. Uma das possibilidades descartadas foi a busca de eficiência ao se desenvolver uma biblioteca *C++* para o processamento de sinais. Isto se mostrou ineficaz no contexto do projeto atual pois a transmissão de dados era dificultada, aumentando drasticamente o tempo de desenvolvimento. Notou-se que era suficiente uma implementação nativa em *Java* com configurações de performance adequadas.

Define-se então a utilização da classe *AudioTrack* para a reprodução, primeiramente com uma implementação simples e seguindo-se com otimizações de performance, processamento paralelo, etc.

5.2.2.4 Desenvolvimento do Filtro

Foi desenvolvida uma classe de filtro, com um método capaz de processar as amostras obtidas antes de serem escritas na saída. Este processamento foi então replicado um número de vezes equivalente ao número de bandas representadas no audiograma.

5.2.2.5 Amplificação Inteligente

Com a possibilidade de tratar faixas de frequência separadamente, é então possível obter o perfil de audibilidade via interface e transmitir esses dados para o processamento. Durante essa etapa do desenvolvimento, também é feita a obtenção das equivalências entre sinais e pressões sonoras no mundo físico, a fim de utilizar os dados de audiograma e curvas de referência com as unidades padronizadas.

5.2.2.6 Ajustes Visuais e Testes

Como ajustes finais do projeto, a interface gráfica é atualizada, para ficar melhor integrada ao código de fundo. Testes são realizados com pessoas com perda auditiva.

5.3 Processamento do Sinal

Após feita a aquisição do sinal em tempo real, será feito a etapa de processamento, que inclui a filtragem, amplificação e reconstrução do sinal. O tempo de execução do processamento deve ser menor que o período de amostragem do sinal, para não haver problemas na reprodução do áudio. As etapas serão detalhadas a seguir.

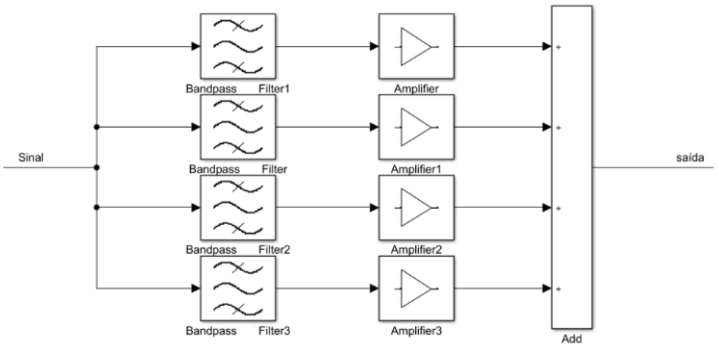

Figura 10: Diagrama de processamento

Inicialmente o sinal recebido será filtrado por um banco de filtros passa-faixa. O sinal será dividido em torno de 6 à 8 bandas distintas, com tamanhos diferentes, de forma similar ao que ocorre na região da cóclea do ouvido humano. Os filtros, de N coeficientes, terão um buffer com as amostras de entrada $x(n), x(n-1), ..., x(n-N+1)$. O valor de N deve ser o menor possível para consumir menos tempo de processamento. Após a filtragem será feita a amplificação da saída de cada filtro.

5.3.1 Amplificação Inteligente

Para fazer a amplificação, primeiro será calculado a potência do sinal de saída de cada filtro, medida por $P_k(n) = \lambda P_k(n-1) + (1-\lambda)v_k^2(n)$ sendo $\lambda \approx 0.9$ e k a saída do k_{esimo} filtro

Com base no audiograma do usuário, como dito na seção do Audiograma, a partir do limiar de audibilidade típico do ouvido humano, será obtido o limiar de audibilidade do usuário. Com base na curva do limiar de audibilidade de uma pessoa típica será feita a correção para a curva do limiar de audibilidade do usuário, através de um fator c_k.

A potência original (isto é, não corrigida para deficiência auditiva) produzida em uma dada banda, com fone de sensibilidade g naquela banda e fator de tensão $b \leq 1$ do botão de volume é, em escala linear,

$$p_o = g(bf)^2 \frac{\overline{x_o^2}}{R} 10^3,$$

onde $g = 10^{S/10}$ (S é a especificação em dB SPL/mW do fone), R é a impedância do fone e f é a tensão em Volts na saída do fone quando $x_o = 1$ e $b = 1$. O valor médio quadrático $\overline{x_o^2}$ é atualizado a cada amostra x_o segundo

$$\overline{x_o^2} = \lambda \overline{x_o^2} + (1-\lambda)x_o^2, \quad \lambda < 1,$$

sendo o valor de λ escolhido para ter uma resolução temporal adequada. Em dB SPL a potência original é então $P_o = 10 \log p_o$. Sejam, na banda em questão, em dB SPL:

- P_l a potência de limiar normal
- $P_d = P_l + A$ a potência de limiar com deficiência, onde A é a perda apontada no audiograma
- $P_m = 10\log(g(max[b]f)^2 \frac{max[x_o^2]}{R} 10^3$ a potência máxima que pode ser produzida
- Pa a potência corrigida para deficiência, produzida pelo fone

Queremos, de início, que a relação entre P_a e P_o seja uma reta passando pelos pontos (P_l, P_d) e (P_m, P_n), ou seja

$$P_a = (P_o - P_l)\frac{P_m - P_d}{P_m - P_l} + P_d.$$

Re-escrevendo esta relação com $p_l = 10^{P_l/10}, p_d = 10^{P_d/10}$ e $\Delta = (P_m - P_d)/(P_m - P_l) \leq 1$, resulta

$$p_a = \left(\frac{p_o}{p_l}\right)^{\Delta} p_d.$$

A correção é realizada um fator multiplicativo, isto é, $x_o = cx_o$, o que implica em $p_a = c^2 p_o$ e, portanto, $c = \sqrt{p_a/p_o}$ Inserindo a expressão acima para p_a e a expressão inicial para p_o temos

$$x_a = k(b^2\overline{x_o^2})^d x_o,$$

onde

$$k = \sqrt{\frac{p_d}{p_l^{\Delta}}}\left(\frac{gf^2 10^3}{R}\right)^d, \quad d = \frac{\Delta - 1}{2}.$$

Como $\Delta - 1 \leq 0, |x_a| \to \infty$ quando $|x_o| \to 0$, o que não é desejável. Para resolver isso sem introduzir variações abruptas em x_a, podemos impor que quando $p_o < p_l$ a correção c é fixada no seu valor para $p_o = p_l$ que é $c_max = \sqrt{p_d/p_l}$. A condição $p_o = p_l$ implica em

$$b^2\overline{x_o^2} = \frac{Rpl}{gf^2 10^3} \triangleq \overline{x_{omin}^2},$$

logo, a fórmula final de correção é

$$x_a = cx_o, \text{com} \begin{cases} c = k(b^2\overline{x_o^2})^d, & \text{se } b^2\overline{x_o^2} \geq \overline{x_{omin}^2} \\ c = c_max, & \text{caso contrário} \end{cases}$$

5.3.2 Reconstrução do sinal

Após feita a amplificação do sinal será feita a reconstrução do sinal, para isto basta somar cada banda, obtendo a saída $y(n) = \sum_k c_k(n)v_k(n)$. Essa saída será, então, reproduzida pelo celular.

Para maior eficiência considera-se, também, usar um buffer de $N + L - 1$ amostras, de forma que os filtros processem L saídas de cada vez para serem reproduzidas.

6 PROJETO E IMPLEMENTAÇÃO

Neste capítulo, são abordados os aspectos técnicos da implementação

6.1 Arquitetura geral

A classe MainActivity administra o funcionamento geral do software, executando o processamento dos elementos gráficos, abrindo o seletor de arquivos e instanciando a classe Player

6.1.1 Classe Player

Esta classe administra o funcionamento das posteriores, orquestrando a extração, filtragem e amplificação dos arquivos.

6.1.2 Classe FileExtractor

Esta classe é responsável por extrair as amostras de áudio a partir de formatos comprimidos. Uma função de callback é definida para executar o processamento quando existem amostras disponíveis.

6.1.3 Classe IIRFilter

Implementação de um filtro IIR, instanciado para cada banda do audiograma. Possui método *process*, capaz de executar a filtragem das amostras

6.1.4 Classe FilterConstants

Armazena constantes calculadas via MATLAB, que definem os filtros IIR.

6.1.5 Classe Amplifier

Recebe os dados do audiograma e as faixas separadas do áudio. Executa o algoritmo de amplificação definido anteriormente.

7 CONSIDERAÇÕES FINAIS

Neste capítulo é abordado um apanhado das conclusões obtidas no desenvolvimento e conclusão do projeto, bem como planos e perspectivas futuros.

7.1 Conclusões do Projeto de Formatura

Com o desenvolvimento deste projeto, foi possível chegar a algumas conclusões acerca do mercado de software e acessibilidade, bem como do desenvolvimento de aplicações de processamento em tempo real.

Primeiramente, durante as etapas iniciais do projeto, notou-se uma escassez de recursos e referências disponíveis. Compreender melhor sobre tipos específicos de deficiência, suas dificuldades e características e associá-las a ferramentas já disponíveis no mercado mostrou-se uma tarefa árdua. Usuários que de fato necessitariam este tipo de apoio possivelmente teriam muita dificuldade em encontrar tais materiais, inclusive pela maioria não existir em Português. Nota-se uma falta de alcance de softwares capazes de auxiliar deficientes auditivos, o que é capaz de segregá-los mais ainda do meio digital.

Concomitantemente, percebeu-se quão inexplorada é a área de processamento digital de sinais no contexto móvel. Muitas das bibliotecas e frameworks encontrados se mostraram de nível demasiadamente alto ou foram incompatíveis com o contexto da aplicação que é, de certa forma, geral.

Por fim, evidenciou-se a importância de um bom planejamento e gestão no projeto de software. A capacidade do time poderia ser aproveitada com mais eficiência com um plano de execução mais detalhado. Indefinições arquiteturais e tecnológicas também prejudicaram o andamento.

7.2 Contribuições

Com este projeto foi desenvolvido uma estrutura expansível e customizável para compensação auditiva, baseado em tecnologias recentes e necessidades evidentes. Como contribuição, tem-se a disponibilização desta aplicação, capaz de auxiliar na experiência de indivíduos com deficiências auditivas Além disso, o projeto tem seu código aberto e disponível online, permitindo expansões futuras de colaboradores interessados.

7.3 Perspectivas de Continuidade

A primeira fase após a apresentação desta prova de conceito, é iniciar uma rodada extensiva de metodologias de Experiência do Usuário. É essencial compreender melhor o público afetado, efetuar testes da solução e implementar mudanças pontuais observadas. Nesta mesma linha de atuação, deve-se retrabalhar toda a interface e interação com a aplicação, permitindo um Design mais minimalista e eficaz, seguindo princípios heurísticos e buscando uma usabilidade mais facilitada.

Em seguida, existe a perspectiva de expandir o escopo do projeto. Utilizando a mesma base de código, com os algoritmos desenvolvidos, é possível alterar as fontes de áudio utilizadas. No conceito inicial do projeto, todos os sinais sonoros do aparelho seriam tratados pelo software. Essa ideia pode ser resgatada, o que significaria uma usabilidade completamente nova porém ainda mais impactante.

REFERÊNCIAS

1 PEREIRA, M.; FERES, M. Próteses auditivas. *Medicina (Ribeirão Preto)*, v. 38, n. 34, p. 257–261, 2005. Disponível em: ⟨https://www.revistas.usp.br/rmrp/article/view/455⟩. Acesso em: 9 jul. 2022.

2 NATIONAL INSTITUTE ON DEAFNESS AND OTHER COMMUNICATION DISORDERS. *Hearing Aids*. Bethesda, 2017. Disponível em: ⟨https://www.nidcd.nih.gov/health/hearing-aids#hearingaid_04⟩. Acesso em: 9 jul. 2022.

3 MOREIRA, P. *Porque APARELHOS AUDITIVOS custam CARO: os MOTIVOS*. 2022. Disponível em: ⟨https://cronicasdasurdez.com/aparelhos-auditivos-custam-tao-caro/⟩. Acesso em: 9 jul. 2022.

4 BURT, P.; STOLFI, G. *DESENVOLVIMENTO DE UM TERMINAL TELEFÔNICO COM COMPENSAÇÃO PARA DEFICIENCIAS AUDITIVAS*. São Paulo: [s.n.], 2004.

5 CONTRATAÇÕES de profissionais com deficiência disparam 147% até agosto, aponta pesquisa. 2021. Disponível em: ⟨https://g1.globo.com/trabalho-e-carreira/noticia/2021/09/23/contratacoes-de-profissionais-com-deficiencia-disparam-147percent-ate-agosto-aponta-pesquisa.ghtml⟩. Acesso em: 9 jul. 2022.

6 AGÊNCIA BRASIL: País tem 10,7 milhões de pessoas com deficiência auditiva, diz estudo. São Paulo: [s.n.], 2019. Disponível em: ⟨https://ilocomotiva.com.br/clipping/agencia-brasil-pais-tem-107-milhoes-de-pessoas-com-deficiencia-auditiva-diz-estudo⟩. Acesso em: 15 mai. 2022.

7 BRASILEIROS são os que passam mais tempo por dia no celular, diz levantamento. 2022. Disponível em: ⟨https://g1.globo.com/tecnologia/noticia/2022/01/12/brasileiros-sao-os-que-passam-mais-tempo-por-dia-no-celular-diz-levantamento.ghtml⟩. Acesso em: 9 jul. 2022.

8 MAWSTON, N. Half the world owns a smartphone. 24 jun. 2021. Disponível em: ⟨https://www.strategyanalytics.com/access-services/devices/mobile-phones/smartphone/smartphones/reports/report-detail/half-the-world-now-owns-a-smartphone⟩. Acesso em: 12 dez. 2022.

9 INTERNATIONAL FEDERATION OF THE PHONOGRAPHIC INDUSTRY. *Music Listening*. 2019. Acesso em: 12 dez. 2022.

10 WORLD HEALTH ORGANIZATION. *World Report On Hearing*. 2021.

11 EQUALIZER | Android Developers. Disponível em: ⟨https://developer.android.com/reference/android/media/audiofx/Equalizer⟩. Acesso em: 16 out. 2022.

12 STACK OVERFLOW. *Number of bands in Android Equalizer*. 2013.

13 JUCE. Disponível em: ⟨https://juce.com/⟩. Acesso em: 14 dez. 2022.

14 AGÊNCIA BRASIL: País tem 10,7 milhões de pessoas com deficiência auditiva, diz estudo. Disponível em: ¡https://ilocomotiva.com.br/clipping/agencia-brasil-pais-tem-107-milhoes-de-pessoas-com-deficiencia-auditiva-diz-estudo/¿. Acesso em: 15 maio. 2022.

15 NEWBY, H. A.; POPELKA, G. R. – "Audiology". Prentice-Hall, N. Jersey, 1992.

16 RABINER, L. R. ; SCHAFER, R. W. – "Digital Processing of Speech Signals" – Prentice-Hall, 1978

17 Silêncio e no Ruído em Portadores de Perda Auditiva Induzida pelo Ruído" . Revista Brasileira de Otorrinolaringologia, 66 (4), 362-370, 2000.

8 APÊNDICES

.1 Códigos fonte

.1.1 MainActivity

```
package com.example.filterproject;

import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

import android.Manifest;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.net.Uri;

import com.example.filterproject.databinding.ActivityMainBinding;
import com.google.android.material.floatingactionbutton.
   FloatingActionButton;

import java.io.File;
import java.io.IOException;

public class MainActivity extends AppCompatActivity {

    // Used to load the 'filterproject' library on application startup.
```

```
static {
    System.loadLibrary("filterproject");
}

private ActivityMainBinding binding;

public File file;
Player mPlayer = new Player();

FloatingActionButton pickAFileButton;
Button[] plusButton = new Button[Player.NUM_BANDS];
Button[] minusButton = new Button[Player.NUM_BANDS];
float value250 = 0;
int[] audiogram = { 0, 0, 0, 0, 0, 0 };

ActivityResultLauncher<String> mGetContent = registerForActivityResult
    (

        new ActivityResultContracts.GetContent(),
        uri -> {

            this.file = new File(uri.getPath());
            playWav(uri, audiogram);
            String message = String.format(
                    "Reproduzindo␣%s␣com␣valor␣%s",
                    this.file.getPath(), this.audiogram);

            binding.sampleText.setText(message);
        });

public void playWav(Uri uri, int[] audiogram) {
    try {
        mPlayer.initialize(uri, getApplicationContext(), audiogram);
        mPlayer.start();
    } catch (IOException e) {
        e.printStackTrace();
```

```
    }
}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    binding = ActivityMainBinding.inflate(getLayoutInflater());
    setContentView(binding.getRoot());

    pickAFileButton = binding.pickAFileButton;

    plusButton[0] = binding.plusButtonA;
    minusButton[0] = binding.minusButtonA;
    plusButton[1] = binding.plusButtonB;
    minusButton[1] = binding.minusButtonB;
    plusButton[2] = binding.plusButtonC;
    minusButton[2] = binding.minusButtonC;
    plusButton[3] = binding.plusButtonD;
    minusButton[3] = binding.minusButtonD;
    plusButton[4] = binding.plusButtonE;
    minusButton[4] = binding.minusButtonE;
    plusButton[5] = binding.plusButtonF;
    minusButton[5] = binding.minusButtonF;


    for (int i=0; i < Player.NUM_BANDS; i++) {
        int finalI = i;
        plusButton[i].setOnClickListener(v -> {
            audiogram[finalI] += 1;
            updateAudiogram();
        });

        int finalI1 = i;
        minusButton[i].setOnClickListener(v -> {
            audiogram[finalI1] -= 1;
```

```
                updateAudiogram();
            });
        }

        pickAFileButton.setOnClickListener(v -> {
            // start file chooser
            mGetContent.launch("audio/*");
        });

        // Example of a call to a native method
        TextView tv = binding.sampleText;
        ActivityCompat.requestPermissions(MainActivity.this,
                new String[] { Manifest.permission.READ_EXTERNAL_STORAGE },
                1);

    }

    public void updateAudiogram() {
        binding.valueA.setText(String.valueOf(audiogram[0]));
        binding.valueB.setText(String.valueOf(audiogram[1]));
        binding.valueC.setText(String.valueOf(audiogram[2]));
        binding.valueD.setText(String.valueOf(audiogram[3]));
        binding.valueE.setText(String.valueOf(audiogram[4]));
        binding.valueF.setText(String.valueOf(audiogram[5]));
    }

    /**
     * A native method that is implemented by the 'filterproject' native
         library,
     * which is packaged with this application.
     */

}
```

.1.2 Player

```
package com.example.filterproject;

import android.annotation.SuppressLint;
import android.content.Context;
import android.media.AudioAttributes;
import android.media.AudioFormat;
import android.media.AudioTrack;
import android.media.MediaCodec;
import android.media.MediaCodecList;
import android.media.MediaExtractor;
import android.media.MediaFormat;
import android.net.Uri;

import androidx.annotation.NonNull;

import java.io.FileDescriptor;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;

public class Player {

    public static int NUM_BANDS = 6;

    private MediaExtractor mExtractor;
    private int pos;
    float[] floatSamples;
    double[] doubleSamples;
    float[] filteredA, filteredB, filteredC, filteredD, filteredE,
       filteredF;

    double[] doubleFiltered;
    short[] shortArray;

    private IIRFilter[] filters = new IIRFilter[NUM_BANDS];
    private Amplifier[] Amp = new Amplifier[NUM_BANDS];
```

```
AudioTrack mAt;
private MediaCodec mCodec;

static private MediaCodec createMediaCodec(MediaFormat format)
        throws IOException, IllegalArgumentException {
    assert format != null;
    String codecName = new MediaCodecList(MediaCodecList.ALL_CODECS).
       findDecoderForFormat(format);
    MediaCodec codec = null;
    String mime = format.getString(MediaFormat.KEY_MIME);
    // Log.d(, "try to create a codec mime="+mime+" codecName="+
       codecName);
    if (codecName != null)
        codec = MediaCodec.createByCodecName(codecName);
    else if (mime != null)
        codec = MediaCodec.createDecoderByType(mime); // may be throw
           IllegalArgumentException
    return codec;
}

static private MediaExtractor createMediaExtractor(FileDescriptor fd)
   throws IOException {
    MediaExtractor extractor = new MediaExtractor();
    extractor.setDataSource(fd);
    extractor.selectTrack(0);

    return extractor;
}

@SuppressLint("NewApi")
public void initialize(Uri uri, Context context, int[] audiogram)
   throws IOException {
    stop();
    int[] Pl = {10, 6, 4, 4, -5, 16};

    for(int num_amp = 0; num_amp < NUM_BANDS; num_amp++){
```

```
    Amp[num_amp] = new Amplifier(96,1.5f,16,0.9f,Pl[num_amp],
       audiogram[num_amp]);
}

FileDescriptor mFd = context.getContentResolver().
   openFileDescriptor(uri, "r").getFileDescriptor();

mExtractor = createMediaExtractor(mFd);

MediaFormat mFormat = mExtractor.getTrackFormat(0);

mCodec = createMediaCodec(mFormat);
mCodec.configure(mFormat, null, null, 0);

Integer sampleRate = mFormat.getInteger(MediaFormat.
   KEY_SAMPLE_RATE);
Integer numChannels = mFormat.getInteger(MediaFormat.
   KEY_CHANNEL_COUNT);

filters = FilterConstants.getFilters(sampleRate);

int channelConfig = AudioFormat.CHANNEL_OUT_MONO;
int minBufferSize = AudioTrack.getMinBufferSize(sampleRate,
   channelConfig, AudioFormat.ENCODING_PCM_FLOAT);
mAt = new AudioTrack.Builder()
        .setAudioAttributes(new AudioAttributes.Builder()
                .setUsage(AudioAttributes.USAGE_MEDIA)
                .setContentType(AudioAttributes.CONTENT_TYPE_MUSIC)
                .build())
        .setPerformanceMode(AudioTrack.PERFORMANCE_MODE_LOW_LATENCY
           )
        .setTransferMode(AudioTrack.MODE_STREAM)
        .setAudioFormat(new AudioFormat.Builder()
                .setEncoding(AudioFormat.ENCODING_PCM_FLOAT)
                .setSampleRate(sampleRate)
                .setChannelMask(channelConfig)
```

```
                .build())
        .setBufferSizeInBytes(minBufferSize * 3)
        .build();

floatSamples = new float[0];
doubleSamples = new double[0];
filteredA = new float[0];
filteredB = new float[0];
filteredC = new float[0];
filteredD = new float[0];
filteredE = new float[0];
filteredF = new float[0];
doubleFiltered = new double[0];
shortArray = new short[0];


mCodec.setCallback(new MediaCodec.Callback() {
    @Override
    public void onInputBufferAvailable(@NonNull MediaCodec codec,
       int index) {
        ByteBuffer inputBuffer = codec.getInputBuffer(index);
        if (inputBuffer == null) {
            return;
        }
        int size = mExtractor.readSampleData(inputBuffer, 0);
        if (size < 0) {
            return;
        }
        long time = size > 0 ? mExtractor.getSampleTime() : 0;
        int flags = size > 0 ? 0 : MediaCodec.
           BUFFER_FLAG_END_OF_STREAM;

        codec.queueInputBuffer(index, 0, size, time, flags);
        mExtractor.advance();
    }
```

```
@Override
public void onOutputBufferAvailable(@NonNull MediaCodec codec,
    int index,
        @NonNull MediaCodec.BufferInfo info) {
    ByteBuffer outputBuffer = mCodec.getOutputBuffer(index);
    if (outputBuffer == null) {
        return;
    }

    int isStereo = numChannels == 2 ? 1 : 0;
    int bufferSizeInBytes = info.size;
    int desiredDoubleArraySize = bufferSizeInBytes / 2;
    if (desiredDoubleArraySize > shortArray.length) {
        doubleSamples = new double[desiredDoubleArraySize /
            numChannels];
        shortArray = new short[desiredDoubleArraySize];
        filteredA = new float[desiredDoubleArraySize /
            numChannels];
        filteredB = new float[desiredDoubleArraySize /
            numChannels];
        filteredC = new float[desiredDoubleArraySize /
            numChannels];
        filteredD = new float[desiredDoubleArraySize /
            numChannels];
        filteredE = new float[desiredDoubleArraySize /
            numChannels];
        filteredF = new float[desiredDoubleArraySize /
            numChannels];
    }

    outputBuffer.order(ByteOrder.LITTLE_ENDIAN).asShortBuffer()
        .get(shortArray, 0,
            desiredDoubleArraySize);

    for (int t = 0; t < desiredDoubleArraySize / numChannels; t
        ++) {
```

```
    // Importante ser blocking
    doubleSamples[t] = (double) (shortArray[numChannels * t
       ] + shortArray[numChannels * t + isStereo])
           / 0x10000;
}
float[] output = new float[desiredDoubleArraySize /
   numChannels];
filters[0].process(doubleSamples, filteredA,
   desiredDoubleArraySize / numChannels);
filters[1].process(doubleSamples, filteredB,
   desiredDoubleArraySize / numChannels);
filters[2].process(doubleSamples, filteredC,
   desiredDoubleArraySize / numChannels);
filters[3].process(doubleSamples, filteredD,
   desiredDoubleArraySize / numChannels);
filters[4].process(doubleSamples, filteredE,
   desiredDoubleArraySize / numChannels);
filters[5].process(doubleSamples, filteredF,
   desiredDoubleArraySize / numChannels);

Amp[0].amplify(filteredA, desiredDoubleArraySize /
   numChannels);
Amp[1].amplify(filteredB, desiredDoubleArraySize /
   numChannels);
Amp[2].amplify(filteredC, desiredDoubleArraySize /
   numChannels);
Amp[3].amplify(filteredD, desiredDoubleArraySize /
   numChannels);
Amp[4].amplify(filteredE, desiredDoubleArraySize /
   numChannels);
Amp[5].amplify(filteredF, desiredDoubleArraySize /
   numChannels);

for(pos = 0; pos < desiredDoubleArraySize / numChannels;
   pos++) {
    output[pos]= filteredA[pos] + filteredB[pos] +
```

```
                    filteredC[pos] + filteredD[pos] + filteredE[pos] +
                    filteredF[pos];
            }
            mAt.write(output, 0, desiredDoubleArraySize / numChannels,
                AudioTrack.WRITE_BLOCKING);

            mCodec.releaseOutputBuffer(index, false);
        }

        @Override
        public void onError(@NonNull MediaCodec codec, @NonNull
            MediaCodec.CodecException e) {

        }

        @Override
        public void onOutputFormatChanged(@NonNull MediaCodec codec,
            @NonNull MediaFormat format) {

        }
    });

}

public void start() {
    mAt.play();
    mCodec.start();
}

public void stop() {
    if (mCodec != null) {
        mCodec.stop();
        mCodec.release();
        mCodec = null;
    }
    if (mExtractor != null) {
```

```
            mExtractor.release();
            mExtractor = null;
        }
        if (mAt != null) {
            mAt.release();
        }
    }
}
```

.1.3 IIRFilter

```
package com.example.filterproject;

import android.util.Log;

import java.util.Arrays;

public class IIRFilter {

    public IIRFilter(double a_[], double b_[]) {
        // initialize memory elements
        N = Math.max(a_.length, b_.length);
        x = new double[N]; y = new double[N]; m = 0;
        for (int i = 0; i < x.length; i++) {
            x[i] = 0.0f;
            y[i] = 0.0f;
        }
        // copy filter coefficients
        a = new double[N];
        int i = 0;
        for (; i < a_.length; i++) {
            a[i] = a_[i];
        }
        for (; i < N; i++) {
            a[i] = 0.0f;
        }
```

```
    b = new double[N];
    i = 0;
    for (; i < b_.length; i++) {
        b[i] = b_[i];
    }
    for (; i < N; i++) {
        b[i] = 0.0f;
    }
}

// Filter samples from input buffer, and store result in output
    buffer.
// Implementation based on Direct Form II.
// Works similar to matlab's "output = filter(b,a,input)" command
public void process(double input[], float output[], int size) {
    for (int i = 0; i < size; i++) {

        x[m] = input[i];
        yaux = b[0]*x[m];
        k = 1;
        while(k <= m)
        {
            yaux += b[k]*x[m-k] - a[k]*y[m-k];
            k++;
        }
        while(k < N)
        {
            yaux += b[k]*x[m+N-k] - a[k]*y[m+N-k];
            k++;
        }
        y[m] = yaux;
        output[i] = (float )yaux;

        m++;
        if (m >= N) //m = (m mod N);
        {
```

```
            //Log.i("audioFloats1", Arrays.toString(x));
            //Log.i("audioFloats2", Arrays.toString(y));
            m = 0;
        }
        /*
        float in = input[i];
        float out = 0.0f;
        for (int j = memory.length-1; j >= 0; j--) {
            in -= a[j+1] * memory[j];
            out += b[j+1] * memory[j];
        }
        out += b[0] * in;
        output[i] = out;
        // shift memory
        for (int j = memory.length-1; j > 0; j--) {
            memory[j] = memory[j - 1];
        }
        memory[0] = in;
         */

    }
    //Log.i("MyAndroidClass", Arrays.toString(output));

}
public void process2(){
    Log.d("valor22",String.valueOf(m));
    m++;
}

private double[] a;
private double[] b;
private int m;
int N;
private double yaux;
private double[] x;
private double[] y;
```

```
    int k;
}
```

.1.4 Amplifier

```
package com.example.filterproject;
import static java.lang.Math.log10;
import static java.lang.Math.sqrt;
import static java.lang.Math.pow;

public class Amplifier {
    private int audiogram;
    private float g, Pm, d, pl, pd, k, f, x2_min, cmax, lambda, delta, Pd,
        R, x2_m;
    public Amplifier(float sensibilidade, float f, float R, float lambda,
       float Pl, int A) {
        g = (float) pow(10, (sensibilidade/10)); this.f = f; this.R = R;
           this.lambda = lambda;
        Pm = 10*((float) log10(g*pow(f, 2)*(1/R)*pow(10, 3)));
                Pd = Pl + A; //Limiar para pessoa com deficiencia
                delta = (Pm - Pd)/(Pm - Pl);
        d = (delta-1)/2;
        pl = (float) pow(10,(Pl/10));
        pd = (float) pow(10,(Pd/10));
        k = (float) (sqrt(pd/pow(pl,delta))*pow(g*pow(f,2)*pow(10,3)/R,d))
           ;
        x2_min = (float) (R*pl/(g*pow(f,2)*pow(10,3)));
        cmax = (float) sqrt(pd/pl);
    }


    public void amplify(float[] amostras, int N) {
        int i; float c;
        for(i = 0; i < N; i++){
            x2_m = (float) (lambda * x2_m + (1 - lambda) * pow( amostras[i
               ], 2 ));
```

```
            if(x2_m < x2_min)
            {
                c = cmax;
            }
            else
            {
                c = (float) (k * pow(x2_m, d));
            }
            amostras[i] = c*amostras[i];
        }
    }
}
```