UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI

Gabriel Rodrigues Chaves Carneiro

Pulsemeeter, um mixer virtual

São João Del Rei

2023

UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI

Gabriel Rodrigues Chaves Carneiro

Pulsemeeter, um mixer virtual

Monografia apresentada como requisito da disciplina de Projeto Orientado em Computação II do Curso de Bacharelado em Ciência da Computação da UFSJ.

Orientador: Flávio Luiz Schiavoni

Universidade Federal de São João del-Rei — UFSJ

Bacharelado em Ciência da Computação

São João Del Rei

2023

Gabriel Rodrigues Chaves Carneiro

Pulsemeeter, um mixer virtual

Monografia apresentada como requisito da disciplina de Projeto Orientado em Computação II do Curso de Bacharelado em Ciência da Computação da UFSJ.

Trabalho aprovado. São João Del Rei, 12 de junho de 2023:

Flávio Luiz Schiavoni
Orientador

Elverton Carvalho Fazzion
Convidado 1

Rômulo Augusto Vieira Costa
Convidado 2

São João Del Rei
2023

Agradecimentos

À todas as pessoas que contribuíram com o desenvolvimento do projeto no GitHub, criando issues e pull requests, o desenvolvimento teria sido bem mais árduo sem sua ajuda. À todas as pessoas no servidor do projeto no Discord por testarem, relatarem problemas e enviarem sugestões, sem sua ajuda teria que gastar muito mais tempo testando o sistema, sem sombra de dúvidas foram um componente essencial para o projeto ter chegado onde chegou. Ao meu orientador que nos momentos em que pensei em desistir de escrever essa monografia, me deu direções essenciais para a conclusão deste projeto. À minha família por todo suporte, e estrutura que me permitiram chegar onde cheguei. À todas as pessoas que fizeram doações no Patreon e no Paypal. Ao usuário "mk-fg"no GitHub por desenvolver a versão Python da biblioteca libpulse, que é o núcleo de todo o projeto.

Resumo

Palavras-chaves: mixer; linux; audio;

Assim como alguns mixers possuem a função de agrupamento de fontes, no sistema de áudio de um computador existem categorias de aplicações como jogos, players de música, sons do sistema e players de vídeo, que também podem ser agrupadas e ter seus volumes regulados para controlar a saída do som do sistema. As aplicações destas categorias não serão necessariamente sempre as mesmas, pois o usuário pode jogar mais de um jogo, ou usar mais de um player de música. A partir desta premissa, este trabalho apresenta uma aplicação que traz uma camada de abstração que agrupa aplicações similares e permite uma manipulação mais generalizada das fontes de som do sistema, sem ter o esforço de micro gerencia-las sempre que for necessário realizar uma alteração, mas ainda permitindo que elas sejam realizadas caso necessário. Este projeto busca ser um alternativa viável a mixers virtuais como Voicemeeter, porém desenvolvido exclusivamente para sistemas Linux.

Lista de ilustrações

Lista de abreviaturas e siglas

PA	Pulseaudio
PW	Pipewire
ALSA	Advanced Linux Sound System
API	Application Programming Interface (Interface de programação de aplicações)
GUI	Graphical User Interface (Interface gráfica do usuário)
MVC	Model View Controller (Modelo visualização controlador)

Sumário

1 Introdução

Mesas de áudio são usadas a décadas para a produção de áudio, seja para gravar músicas ou para transmissões em rádios e TVs. Essas mesas são extremamente úteis pois trazem consigo várias facilidades para a manipulação rápida de áudio. Embora extremamente úteis, deixam a desejar quando usadas para controlar o áudio de um computador, pois apesar de funcionar normalmente com o áudio das entradas e saídas físicas, não possuem nenhuma maneira de manipular o áudio das aplicações individualmente, que compõem a maior parte das fontes de áudio quando se está gravando ou transmitindo a partir do computador. Quando se realiza uma transmissão ao vivo, ou se grava um vídeo a partir do computador, o usuário necessita de uma maneira rápida e simples de controlar o volume e a saída de cada aplicação no sistema, além das entradas e saídas físicas. Um caso de uso que descreve bem este problema é quando se realiza uma transmissão ao vivo de jogos eletrônicos. É necessário capturar o áudio do microfone, do jogo, do tocador de música, do navegador de internet, dos alertas, precisa-se escutar todas essas fontes, e o volume de retorno não é necessariamente o mesmo volume da captura.

Existem diversas formas de contornar o problema. Uma dessas formas é utilizar as soluções nativas para controle de áudio fornecidas pelo sistema, levando em consideração que cada sistema fornecerá uma maneira diferente de lidar com o áudio. Essa solução chega a um resultado satisfatório, porém são bem mais trabalhosas pois não foram desenvolvidas com esse fluxo de trabalho em mente. Outra alternativa é utilizar algum outro software. Cada um busca um fluxo de trabalho para um propósito específico. Uma das aplicações disponíveis é o Voicemeeter (1), que é um software proprietário disponível apenas para o sistema operacional Microsoft Windows. É uma mesa de áudio virtual capaz de gerenciar múltiplas placas de áudio e aplicações, tendo suporte para uso de efeitos, controle de volume e roteamento de áudio, e é comumente utilizado para realizar transmissões ao vivo de jogos eletrônicos. Outro software disponível é o Jack Audio Conecction Kit (2), que é gratuito e desenvolvido com o objetivo de fornecer conexões em tempo real, disponível para BSD, Linux, macOS, Solaris, Windows, iOS.

A aplicação desenvolvida neste projeto, chamada de pulsemeter, busca solucionar o problema descrito, criando uma forma do usuário gerenciar o áudio de maneira simplificada, podendo controlar entradas e saídas de áudio físicas e aplicações. Semelhante ao Voicemeeter, no entanto desenvolvida exclusivamente para sistemas Linux, com APIs de áudio nativas. Uma é API é, como descrito por Schiavoni (3) (2012, p.1) "...uma série de funções que permitem ao programador acoplar novas funcionalidades a um programa já existente. A escolha de uma determinada API pode simplificar o desenvolvimento de aplicações mas também traz consigo limitações inerentes desta."

Assim como alguns mixers possuem a função de agrupamento de fontes, Pulsemeeter se baseia na ideia de que no sistema de áudio de um computador existem categorias de aplicações que tem o mesmo propósito, como jogos, players de música, sons do sistema, players de vídeo, etc. As aplicações destas categorias não serão necessariamente sempre as mesmas, pois o usuário pode jogar mais de um jogo, ou usar mais de um player de música. A partir disto, criar agrupa aplicações similares para que além de que possam ser controladas e mixadas individualmente, possam também ter seus controles reunidos em apenas um lugar.

2 Referencial Teórico

Este capítulo tem como objetivo discutir e detalhar os conceitos envolvidos e o referencial teórico utilizado no decorrer do desenvolvimento deste projeto, como também softwares e trabalhos relacionados com a proposta desta aplicação.

2.1 Som

A discussão sobre áudio digital e analógico e tudo que circunda este entorno não pode ser iniciada sem antes uma discussão sobre o funcionamento do som. O som é a propagação de uma onda mecânica acústica em um meio sólido, liquido ou gasoso, mas falaremos mais especificamente da propagação no ar. O som nada mais é do que variações na pressão do ar, e nossa habilidade de escutar é a percepção do nosso sistema auditivo dessas variações.O livro (4) discute este tema de maneira profunda. Assim como o ouvido humano, microfones são dispositivos capazes de capturar essas variações na pressão ar, no entanto eles a convertem em eletricidade (4). Caixas de som fazem o processo inverso, transformando eletricidade em variações na pressão do ar. A representação do som no domínio elétrico é chamado de áudio.

2.2 Mixers

Em casos no qual existe a necessidade de se conectar mais de um dispositivo em uma saída, se torna necessário a existência de um dispositivo ou mecanismo capaz de unir sinais distintos em um novo sinal de saída. Um exemplo deste problema é quando se necessita conectar múltiplos microfones simultaneamente em uma caixa de som para que uma banda possa tocar. Uma forma de contornar este problema seria ligando fisicamente os cabos dos dispositivos com algo tipo o adaptador y na imagem 1, de forma que o sinal dos dois se misture sem o auxilio de um dispositivo mais complexo. Esta solução pode chegar a funcionar em alguns casos, mas esta não é uma prática recomendada pelo fato disto introduzir uma série de problemas.

Para que os problema introduzidos na solução descrita anteriormente não ocorram, existem dispositivos chamados **mixers**. Mixers são dispositivos físicos analógicos ou digitais, que tem como objetivo combinar o áudio de múltiplas fontes em uma ou mais saídas, tendo maior controle elas, como o mixer Behringer Micromix Mx400, que possui quatro entradas e uma saída, como visto na figura 2.

Mixers mais avançados são chamados de **Mesas de áudio**. Estes dispositivos

Figura 1 – Adaptador Y possui duas entradas e uma saída

Figura 2 – Mixer Behringer Micromix Mx400, possui 4 entradas e 1 saída

possuem muito mais funções além de misturar o áudio de diversas fontes em uma saída, pois além disso são capazes de controlar volume, modulação, equalização, redução de ruídos, alteração de tom e etc, como a mesa de som Arcano Mm-4 na figura 3. Cada dispositivo de entrada e saída pode ser controlado individualmente, cada um com seus próprios botões e sliders. Algumas mesas de áudio possuem o que são chamados de grupos, que são formas de agrupar tipos semelhantes de fontes em uma saída separada, isto será explicado mais detalhadamente na próxima seção.

2.2.1 Grupos

Grupos em uma mesa de áudio tem o objetivo para associar tipos similares de dispositivos que têm seu áudio misturado separadamente dos outros, e que após configurados individualmente, são agrupados em uma nova fonte que reúne o controle deste grupo em um único lugar, como a mesa de som da figura 5. Um exemplo do uso de grupos é em uma banda que após acertar o volume individual de cada um dos instrumentos e microfones, é comum que eles sejam associados a grupos específicos, para que o controle de elementos similares seja concentrado em grupos, criando-se assim novas fontes permitindo que uma

Figura 3 – Mesa de som Arcano Mm-4

manipulação mais genérica seja feita.

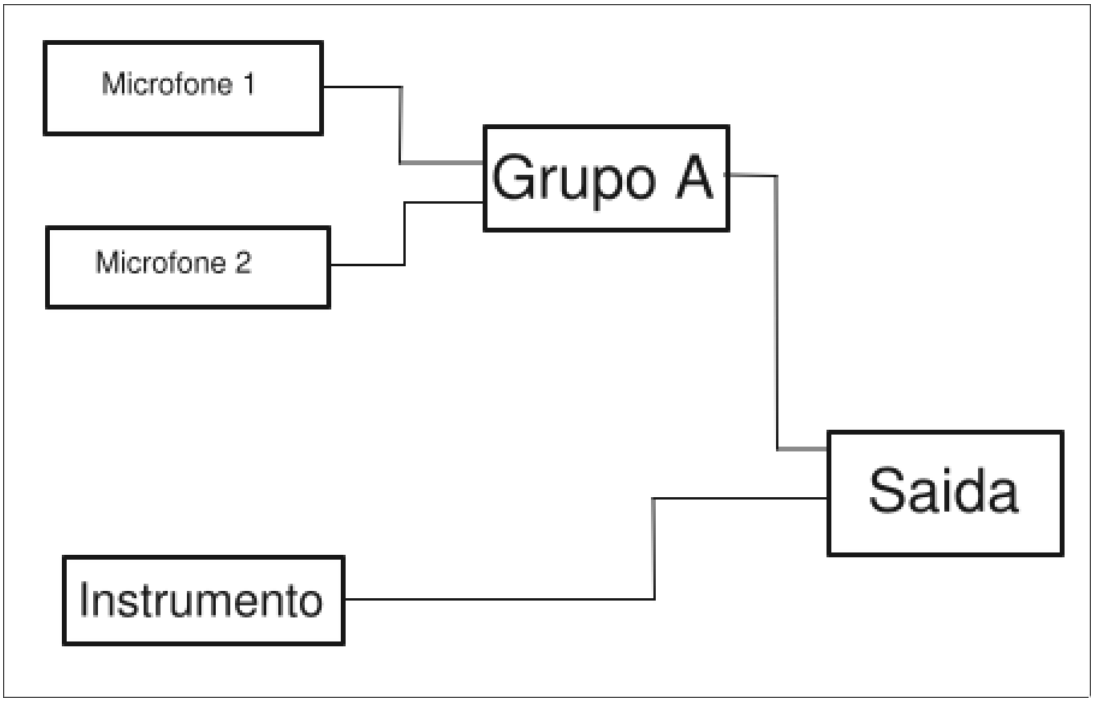

Figura 4 – Desenho do fluxo de áudio em um mixer com grupos

Figura 5 – Mesa de som Arcano Gran-12, possui controles como equalizadores e agrupamento de fontes

2.3 Áudio Digital

Com o advento da tecnologia e dos sistema digitais, surgiu-se a necessidade da criação de dispositivos que fossem capaz de converter o sinal elétrico em uma maneira que um sistema digital consiga interpretar, e assim foram criados os chamados DAC (digital to analog converter), como pode ser visto na figura 7. Os sistemas rodando em Computadores, TVs digitais, smartphones, não consegue trabalhar diretamente com o sinal elétrico, por isso é necessário o uso de um ADC/DAC. Este sinal digital é a representação binária do sinal elétrico, no caso de microfones o sinal elétrico é recebido e transformado em sua representação binária, e em caixas de som acontece o processo inverso, por meio de um ADC. A partir desta representação binária do som é possível que ele seja manipulado por sistemas digitais, ao contrário dos sistemas citados nas seções anteriores. No entanto, ao contrário de tais sistemas, o áudio digital não precisa se limitar à limitação física da quantidade de canais de entrada e saída existentes em um DAC/ADC.

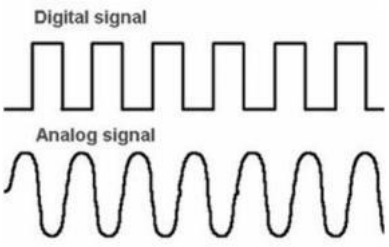

Figura 6 – Diferença de representação do sinal digital e analógico

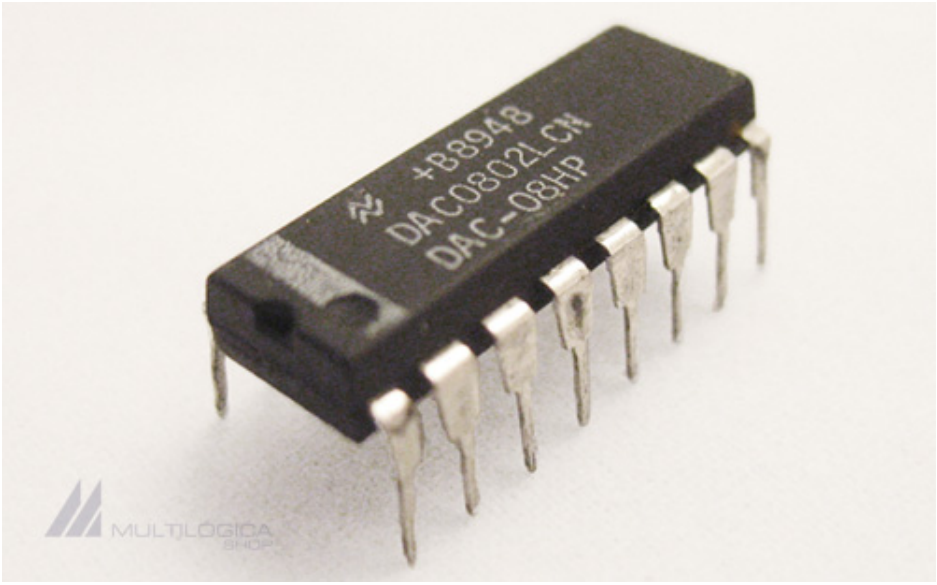

Figura 7 – Um dac de 8 bits

2.3.1 Áudio no computador

Além de necessitarem do auxilio de um DAC, computadores possuem toda uma estrutura desenvolvida para a reprodução e gravação de áudio. Para a comunicação de um sistema operacional com um DAC, é necessário o uso de softwares chamados **drivers**. Eles tem como objetivo fornecer uma maneira do sistema operacional se comunicar com as interfaces de áudio, para que haja uma camada de abstração entre os servidores de áudio e o hardware.

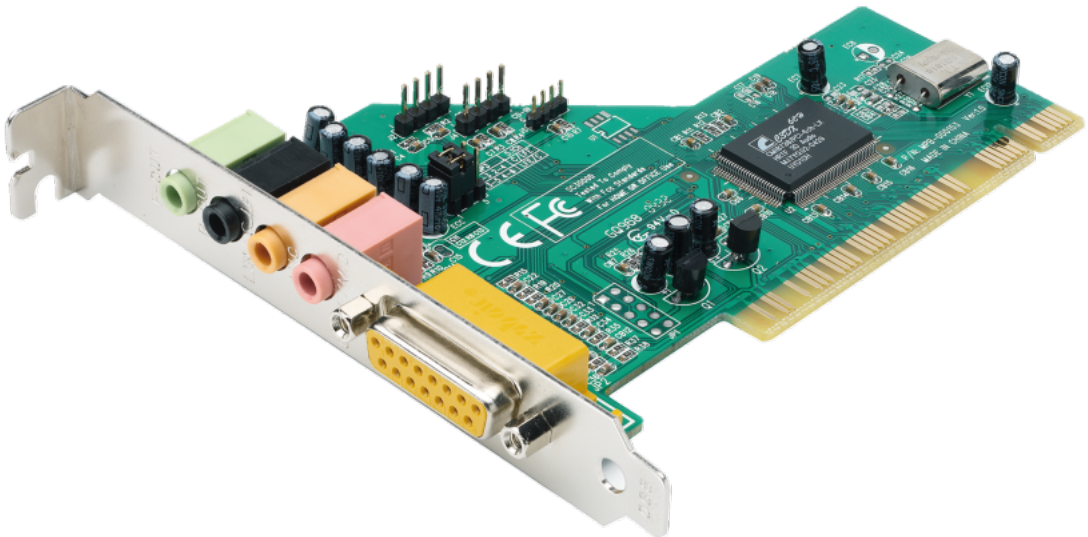

Figura 8 – Uma placa de áudio para computadores

Além do driver de dispositivo, é comum que outras aplicações façam parte do sistema de som do computador como, por exemplo, os servidores de áudio. Servidores de áudio são aplicações responsáveis por gerenciar o áudio no sistema, eles utilizam das ferramentas disponibilizadas pelos drivers para a comunicação com os as interfaces de áudio, adicionando mais uma camada de abstração entre as aplicações e o hardware.

2.4 Mixers Virtuais

Mixers virtuais são aplicações que têm o objetivo de fornecer o mesmo fluxo de trabalho que um mixer físico fornece, e existem diversos disponíveis para todos os sistema operacionais, cada um deles tem suas particularidades voltadas a algum caso de uso específico. Nesta seção iremos discutir sobre alguns deles.

2.4.1 Audio Mixer Simulator

Audio Mixer Simulator desenvolvido por Britt Carr, a pedido de Jay Rozema, um professor do departamento de teatro na Universidade de Miami, é um simulador que busca mostrar visualmente o funcionamento de uma mesa de áudio, com o intuito de auxiliar estudantes do curso de teatro a entenderem fluxo do áudio de cada fonte dentro de um mixer, como ele reage a cada interação do usuário, como mostrado na figura 9. Assim como mixers mais complexos, o usado neste simulador possui controles para efeitos, volumes, e também o agrupamento de dispositivos em grupos.

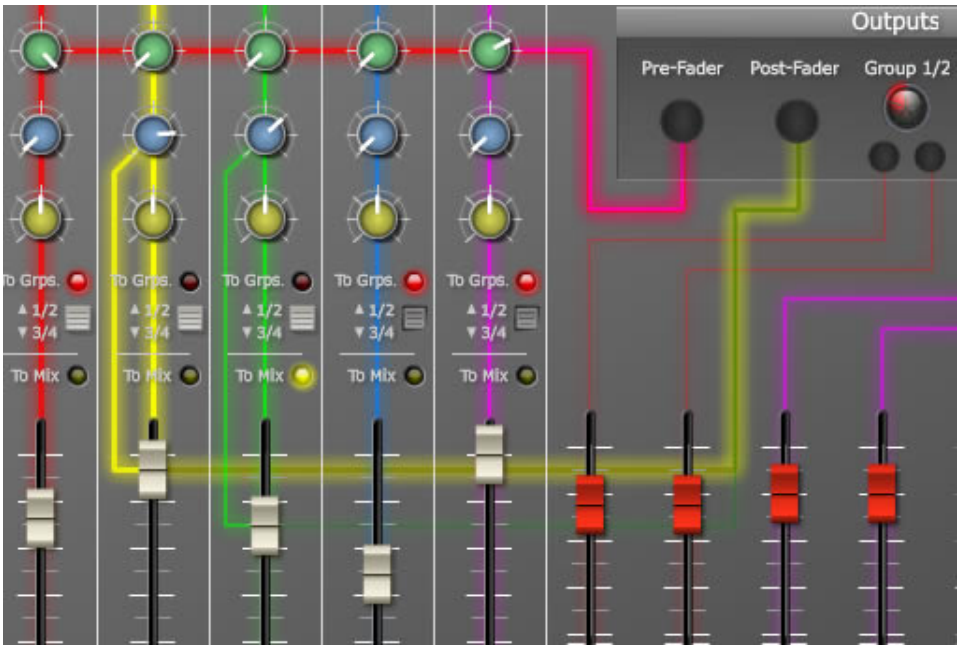

Figura 9 – Tela do Audio mixer simulator

2.4.2 Voicemeeter

Voicemeeter(1) é um software proprietário e donationware (um software que a compra é realizada através de uma doação) exclusivo do sistema operacional Microsoft Windows, disponibilizado em 3 versões: Voicemeeter (figura 10), Voicemeeter Banana (figura 11) e Voicemeeter Potato (figura 12), a única diferença entre elas é o número de entradas e saídas suportadas, e a versão potato exige ativação depois de um tempo de uso, enquanto as outras duas permitem uso gratuito.

Todas as 3 versões tem como objetivo fornecer uma mesa de áudio virtual capaz de mixar cada dispositivo individualmente, selecionar múltiplas saídas de áudio para as entradas, sejam elas entradas físicas como microfone, ou entradas virtuais, como uma música. Tornando possível, por exemplo, transmitir o áudio do microfone para o fone de ouvido enquanto a música toca nos fones e nas caixas de som, sendo o volume de cada um controlado individualmente.

Voicemeeter também permite o uso de dispositivos MIDI[1] para executar comandos. O usuário mapeia os botões de acordo com as funções que necessite que sejam executadas, podendo também ser comandos de controle para por exemplo alterar o volume.

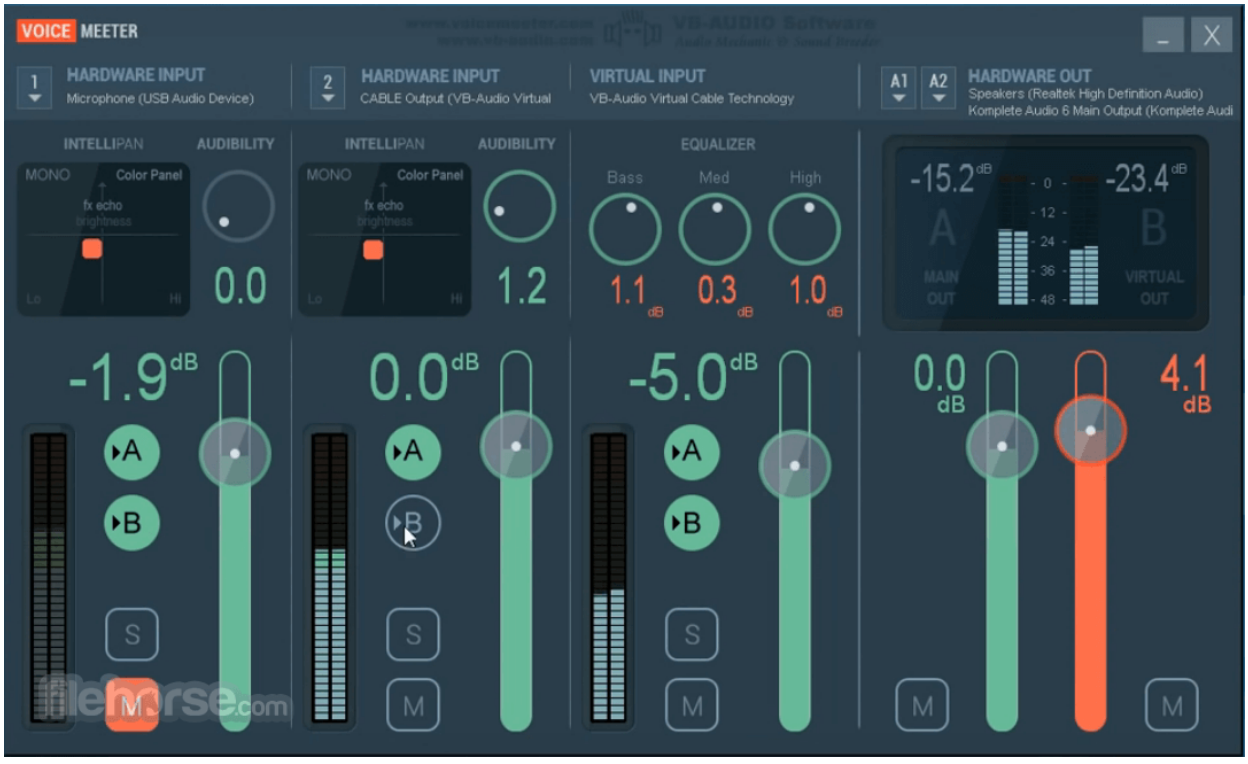

Figura 10 – Voicemeeter standart, uma das 3 versões disponíveis do software

[1] MIDI é um protocolo criado para comunicação de instrumentos musicais digitais e computadores

Figura 11 – Voicemeeter banana, uma das 3 versões disponíveis do software

Figura 12 – Voicemeeter potato, uma das 3 versões disponíveis do software

Grupos

Voicemeeter utiliza o conceito de grupos usado em mesas de áudio, onde são criados grupos de entradas e grupos de saída. Grupos são vistos como dispositivos físicos pelo

sistema operacional, seu objetivo é capturar ou fornecer áudio para aplicações e separá-las em grupos distintos, a fim de reunir o controle de várias aplicações simultaneamente, sem que elas interfiram com outros grupos. Um grupo de entrada individualmente não é capaz de reproduzir áudio pois se trata apenas de um dispositivo lógico, para contornar este problema o usuário pode selecionar para **quais** dispositivos de saída (grupos de saída ou saídas físicas) o áudio deverá ser mixado.

Um exemplo do uso deste tipo de grupo é para associar várias aplicações de música em um único lugar, tendo o controle do volume e saída de todas em um único lugar. Um grupo de saída é visto como uma entrada física pelo sistema operacional, seu objetivo é receber áudio de grupos de entrada, ou entradas físicas, e fornecer uma nova saída para as aplicações. Um exemplo de uso deste tipo de grupo é para criar um microfone que uma aplicação de comunicação possa usar, e desta forma mixar o áudio de um microfone real, um instrumento, e de outras aplicações em uma nova saída.

2.4.3 CMixer

CMixer(5) é um software proprietário desenvolvido para Windows e macOS. Consiste em uma mesa de áudio digital, com o objetivo de replicar uma mesa de áudio física para controlar dispositivos de entrada e saída externos, com foco em transmissões ao vivo, como rádio, TV e concertos ao vivo.

Figura 13 – CMixer

2.4.4 SoudDesk

SoundDesk (6) é um software proprietário, desenvolvido exclusivamente para macOS, tem como objetivo fornecer uma mesa de áudio digital. SoundDesk é capaz de transmitir e modular o áudio entre qualquer dispositivo. Tem o funcionamento similar ao

Voicemeeter, sendo uma alternativa interessante a ele para macOS. Tem foco em criação profissional de áudio, transmissões ao vivo e gravações, sejam elas em performances ao vivo ou em um estúdio.

Figura 14 – Tela principal do SoundDesk

2.5 Arquitetura de áudio no Linux

A arquitetura de áudio no Linux é composta por diversas camadas de drivers, servidores de áudio e abstrações para que o usuário final tenha um sistema que não precise necessariamente configurar.

2.5.1 Advanced Linux Sound Architecture (ALSA)

ALSA é um protocolo e módulo que faz parte do Kernel do Linux, é responsável por gerenciar áudio e MIDI no sistema. Este protocolo gerencia as placas de áudio no sistema, fornecendo configuração automática para os dispositivos, e consegue gerenciar múltiplos dispositivos simultaneamente. ALSA é o sucessor do Open Sound System (OSS), pois além de disponibilizar funções antes não presentes nele, também possuía uma camada de compatibilidade com a API do OSS. Porém nos dias atuais esta funcionalidade não é habilitada na maior parte das distribuições Linux. A API do ALSA é relativamente maior e mais complexa à do OSS, o que torna seu uso mais difícil. ALSA foi projetado com o objetivo de oferecer funções que não eram disponíveis no OSS como sintetizadores MIDI de hardware, mixagem de múltiplos canais e operações de entrada e saída de áudio simultâneas.

Em um sistema Linux moderno, ALSA é a camada de mais baixo nível no sistema quando se lida com áudio, o que significa que servidores de áudio como Pulseaudio, Pipewire e Jack rodam em cima dela, utilizando sua API. Estes servidores de áudio também podem rodar em cima de outras APIs de áudio como OSS, porém na maior parte dos casos não são distribuídas desta maneira. Mais informações sobre alsa e servidores de áudio estão disponíveis no artigo "Introduction to sound programming with ALSA"(7).

2.5.2 Pulseaudio

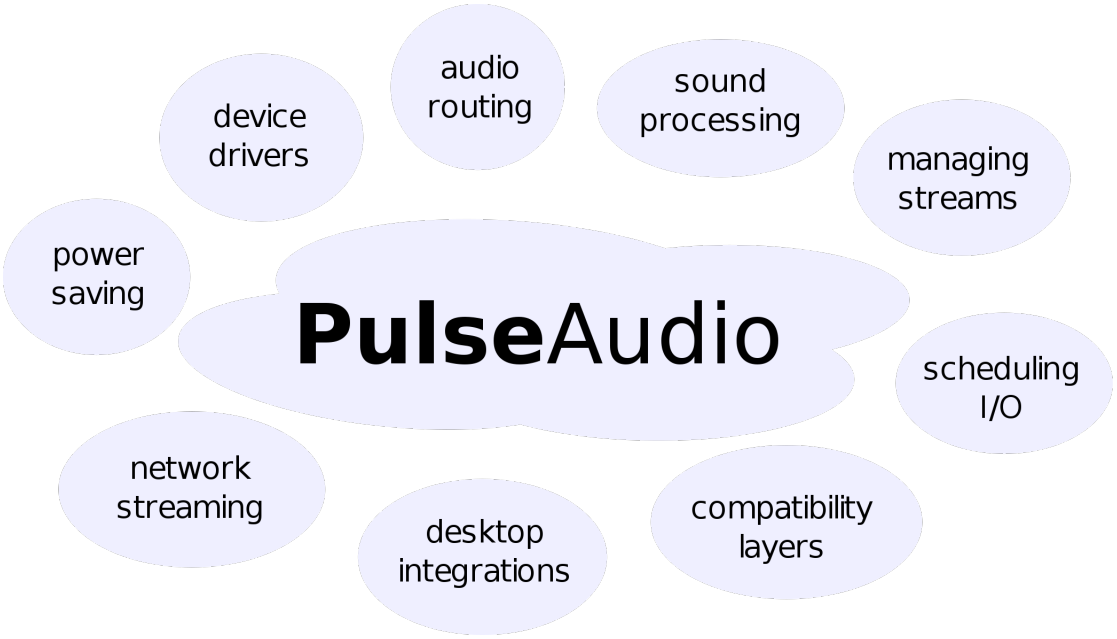

Figura 15 – Pulseaudio

Pulseaudio(8) é um servidor de áudio em rede multi-plataforma desenvolvido para sistemas POSIX e Windows, usado em grande parte das distribuições Linux modernas. Pulseaudio realiza operações no áudio entre aplicações e o hardware, como transferir o áudio entre máquinas distintas, alterar a taxa de amostragem, mixar diversos sons em uma saída/entrada. Pulseaudio não possui a capacidade de lidar com o áudio em baixo nível (no nível do hardware), sendo necessário a utilização de uma API como ALSA e OSS.

Pulseaudio possui uma série de abstrações criadas com o objetivo de fornecer ferramentas de alto nível que não eram possíveis anteriormente usando apenas o hardware, como por exemplo fornecer controle de volume para cada canal de um dispositivo, criação de dispositivos virtuais e uso de efeitos.

Uma fonte de áudio é um dispositivo que possui algumas propriedades básicas como número de canais, volume de cada canal, taxa de amostragem, nome, descrição, etc. Pulseaudio divide as fontes de áudio em algumas categorias e subcategorias. As mais

básicas são as **entradas** e **saídas** de áudio. Uma entrada de áudio é um dispositivo que fornece áudio ao computador, como um microfone, ou um celular conectado numa caixa de som para tocar música. Uma saída de áudio é um dispositivo que recebe o áudio do computador e reproduz para o usuário, como um fone de ouvido ou uma caixa de som. Dentre as entradas e saídas, elas podem ser classificadas em alguns subgrupos: fontes físicas, lógicas e de aplicações.

Fontes Físicas

Fontes físicas são dispositivos conectados fisicamente no computador como fones de ouvido, caixas de som, microfones, mesas de áudio, etc. Elas são divididas em dois tipos, **Source** (entrada) e **Sink** (saída). Entradas de áudio são dispositivos que fornecem áudio ao sistema, como microfones e instrumentos musicais, portanto não é possível fornecer áudio à ela, apenas capturar. Saídas de áudio são dispositivos que recebem áudio do sistema para reprodução, como fones e caixas de som. Porém ao contrário das entradas, Pulseaudio permite que seja possível capturar e transmitir áudio de uma saída.

Fontes Lógicas/Virtuais

Posteriormente chamaremos fontes lógicas de **Grupos** quando entrarmos no contexto da aplicação desenvolvida neste trabalho, porém no Pulseaudio são chamadas de fonte lógicas ou fontes virtuais. Assim como fontes físicas, fontes virtuais são divididas em entradas e saídas. Saídas virtuais são como se fossem microfones virtuais, o usuário enxerga no sistema como se fossem dispositivos físicos, mas não possuem um dispositivo real fornecendo áudio, sendo necessário fornecer áudio à elas de outras maneiras, como veremos na seção sobre Pipewire. Entradas virtuais são como se fossem saídas reais, porém não existem dispositivos físicos conectados de fato à elas, sendo necessário transmitir o áudio destas fontes para saídas reais ou virtuais.

Fontes de Aplicações

São dispositivos virtuais/lógicos criados e usados por aplicações para capturar ou fornecer áudio à um outro dispositivo, seja ele físico ou lógico. Fontes de aplicações podem ser de entrada ou saída, dependendo da necessidade do programa que criou. Toda fonte de aplicação precisa necessariamente capturar (no caso das entradas) ou fornecer (no caso das saídas) áudio de **um** outro dispositivo.

Transmissão em rede

Pulseaudio conta com alguns módulos para transmissão de áudio e controle do servidor via rede. Os módulos de transmissão permitem que o usuário transmita áudio para um servidor remoto de Pulseaudio rodando em outra máquina, também é possível

se conectar em um servidor remoto e usa-lo da mesma maneira que se usaria na própria máquina.

Módulo Loopback

Pulseaudio oferece um módulo chamado loopback que permite que o áudio de uma fonte seja mixado com o áudio de outra fonte. Ele funciona de forma a criar uma nova fonte que recebe o áudio da original, que logo em seguida envia para a fonte e destino, como pode ser visto na figura 16. Este módulo introduz muita latência, podendo chegar a 200ms, e para poder reduzir isto, é necessário sacrificar a qualidade do áudio.

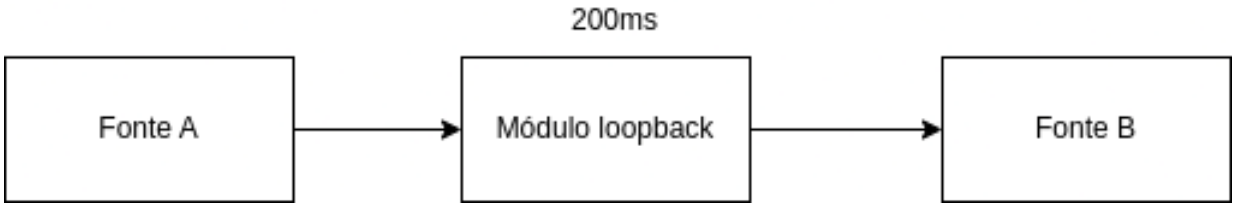

Figura 16 – Funcionamento do módulo loopback

2.5.3 Pipewire

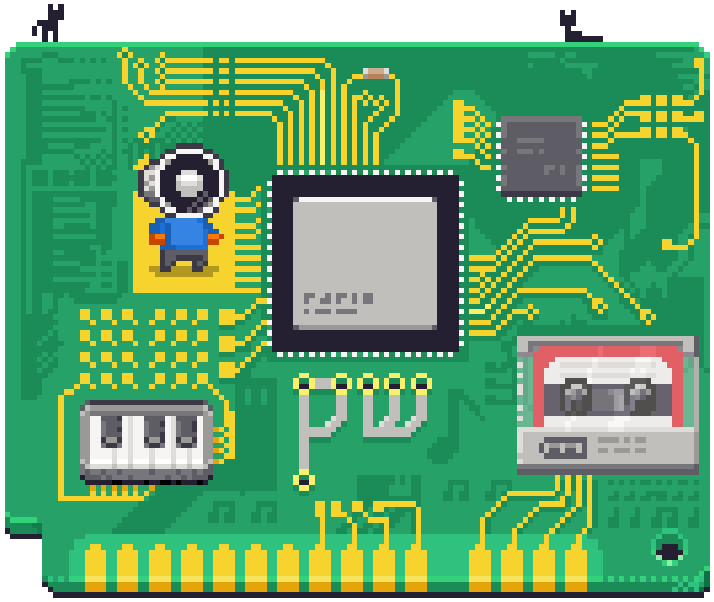

Figura 17 – Pipewire

Pipewire é um servidor de áudio de baixa latência que lida com stream de áudio, vídeo e hardware para Linux, sendo capaz de realizar roteamento de media. Pipewire busca unificar os servidores de áudio disponíveis para Linux, sendo capaz de lidar com clientes Alsa, Pulseaudio e Jack simultaneamente, e permitir que interajam entre si, algo que não é possível sem ele. Pipewire permite que o áudio seja transmitido e mixado através das fontes, com baixa latência.

Mixagem de áudio

O motivo principal pela utilização do servidor Pipewire no desenvolvimento do projeto é a funcionalidade de transmitir áudio entre dispositivos com latência bem mais baixa do que as implementações de transmissão do Pulseaudio. Pipewire trata um dispositivo como um objeto que possui entradas e saídas, e é capaz de criar "fios"virtuais entre esses canais, mixando o áudio entre eles, como demonstrado na figura 18. A aplicação demonstrada na figura 18 chama-se Helvum(9), ela é capaz de conectar as fontes de áudio graficamente através desta representação por fios, onde o usuário seleciona a porta que deseja capturar o áudio, e cria um fio virtual até a porta de entrada da outra fonte de áudio.

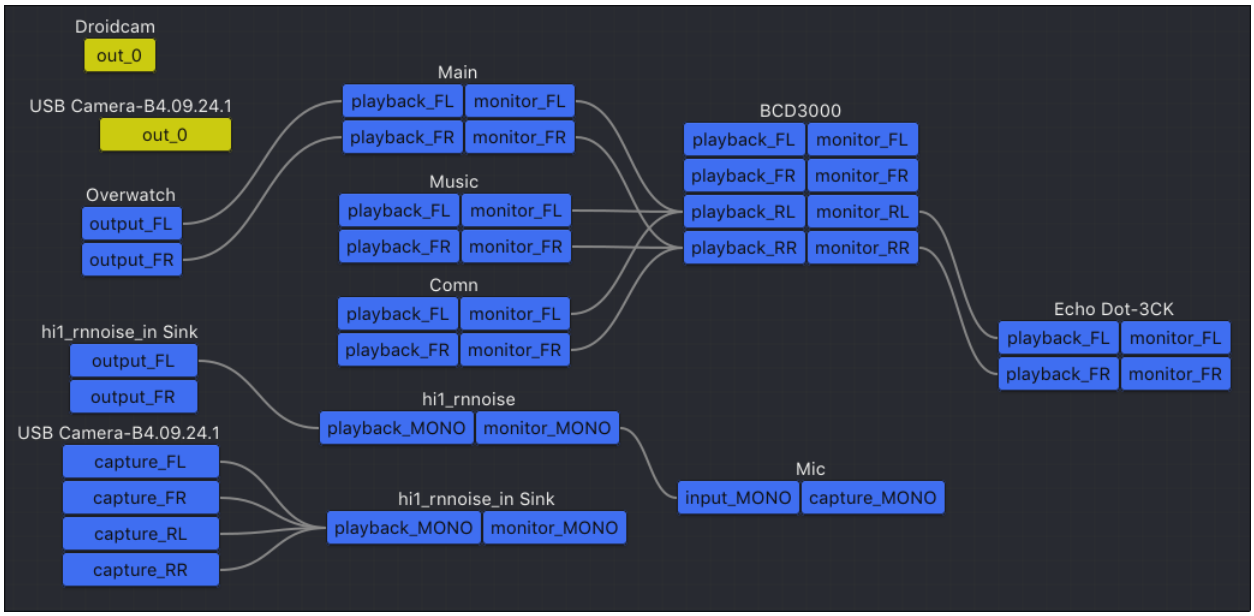

Figura 18 – Helvum, um patchbay disponível para pipewire

2.5.4 JACK

Figura 19 – Jack

Jack(2) é um servidor de áudio profissional com foco em conexões em tempo real e baixa latência para áudio e MIDI, disponível para Linux, BSDs, Windows, macOS, iOS e Solaris. A API do Jack é padronizada, existindo algumas implementações: jack1, jack2 e pipewire-jack. jack1 é a implementação em C desta API, e não recebe grandes modificações a alguns anos, sendo apenas mantida atualizada de acordo com que suas dependências necessitarem. jack2 é a implementação em C++, que introduz suporte à maquinas com múltiplos processadores e sistemas operacionais além de Linux. Pipewire-jack é a implementação da API pelo Pipewire, que mapeia as chamadas do Jack à chamadas equivalentes em sua API (10).

2.5.5 LADSPA

LADSPA(11) é um padrão para softwares processadores de áudio e efeitos para uso em aplicações que sejam compatíveis com o formato. Por exemplo, LADSPA permite que o usuário escreva um programa com algum efeito, como por exemplo redução de ruido, empacote isso em uma biblioteca LADSPA e possa utiliza-lo com aplicações compatíveis.

3 Projeto

Este capítulo busca descrever o projeto do sistema, e posteriormente relatar seu desenvolvimento. Existem diversos cenários nos quais as ferramentas disponibilizadas por padrão nas distribuições Linux não são suficientes para se lidar com o fluxo de áudio de maneira satisfatória, pois algumas funcionalidades apenas estão disponíveis em interfaces em linha de comando. Um usuário normalmente lida com diversas aplicações que capturam ou reproduzem áudio, e cada uma delas é tradada como uma fonte distinta para os servidores de áudio.

Quando se realiza uma transmissão ao vivo, ou se grava um vídeo a partir do computador, se torna necessário uma maneira rápida e simples de se controlar o volume e a saída de cada fonte no sistema, pois não é desejável que estas operações interfiram na experiencia do espectador. É necessário capturar o áudio do microfone, do jogo, do player de música, do navegador de internet, dos alertas, precisa-se escutar todas essas fontes, e o volume de retorno não é necessariamente o mesmo volume da captura.

Após realizar todas configurações o usuário percebe que a música que gostaria de tocar não está disponível no tocador atual e precisa ir para outro, e desta forma terá que configurar esta nova fonte para o estado necessário, e o mesmo se aplica aos jogos e navegadores, pois mesmo que funcionem de maneira similar, ainda são fontes distintas. Neste caso se torna necessário que o usuário realize todas essas operações manualmente, tornando a tarefa bem mais complexa e interferindo na experiência dos espectadores.

Outro caso de uso interessante é quando se realiza a transmissão da tela do computador em aplicações como Google Meet e Discord, pois desta maneira o áudio do sistema não é capturado. A única maneira de se contornar isto é criando um dispositivo virtual que recebe o áudio do microfone e do sistema, e usá-lo como dispositivo de entrada, e esta operação deve ser realizada usando a linha de comando, tornando-a inviável para a maior parte dos usuários.

Tendo estes problemas em vista, a aplicação desenvolvida neste projeto tem como objetivo principal a criação de um Mixer virtual que leva em consideração todas as fontes lógicas e físicas presentes no áudio de um computador, fornecendo ferramentas para realizar operações antes já possíveis, porém extremamente complexas pelo fato de exigirem certo conhecimento técnico.

Assim como alguns mixers possuem a função de agrupamento de fontes, Pulsemeeter se baseia na ideia de que no sistema de áudio de um computador existem categorias de aplicações que tem o mesmo propósito, como jogos, players de música, sons do sistema, players de vídeo, etc. As aplicações destas categorias não serão necessariamente sempre

as mesmas, pois o usuário pode jogar mais de um jogo, ou usar mais de um player de música. A figura 20 ilustra este funcionamento.

A partir desta premissa, o objetivo final é de que aplicações similares devem ser agrupadas para que além de que possam ser controladas e mixadas individualmente, possam também ter seus controles abstraídos e reunidos em uma nova fonte, como mostrado na figura 21.

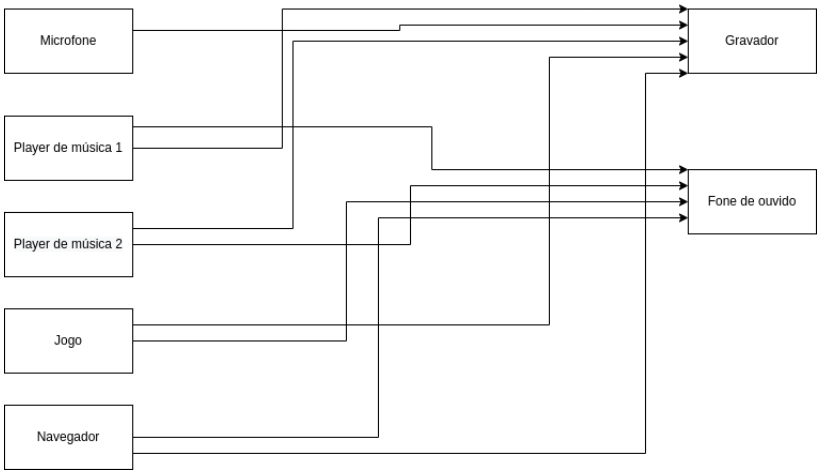

Figura 20 – Uma possível rota do áudio das aplicações em um sistema quando não se utiliza grupos

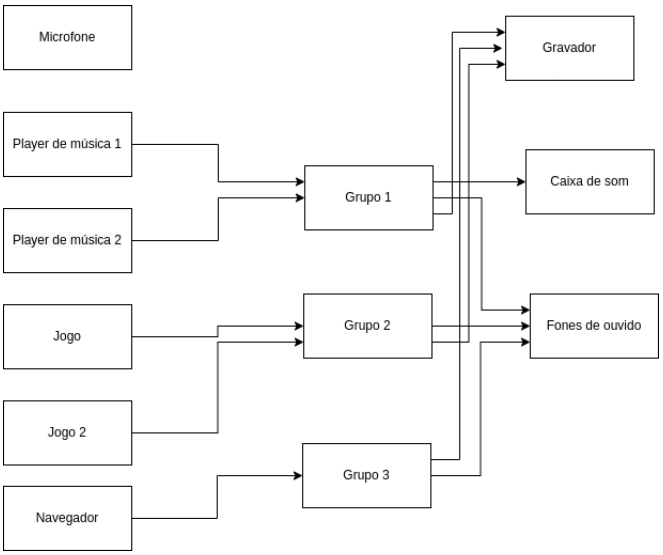

Figura 21 – Uma possível rota do áudio das aplicações em um sistema quando se utiliza grupos

3.1 Funcionalidades desejáveis para a aplicação

Levando em consideração os problemas citados, foram levantados os seguintes requisitos para o sistema.

- Arquitetura que separe clientes e servidor

- Gerenciamento de grupos
- Mixagem do áudio de entradas e de aplicações
- Transmissão de áudio por rede
- Controle de volume individuais e por grupos
- Medidores de pico de volume (VUmeter)
- Uso de efeitos
- Interfaces (GUI, CLI, MIDI)
- Uso de servidores de som nativos de Linux

Tais funcionalidades serão apresentadas a seguir.

3.1.1 Arquitetura

Para que o sistema não tenha alto acoplamento de módulos, é interessante uma implementação que realiza a separação dos módulos de controle e de interface, de modo que novas interfaces possam ser implementadas sem necessidade de se realizar alterações em outras partes do sistema. O servidor de áudio deve funcionar de maneira separada das interface, de forma que múltiplas instâncias de interfaces possam ser usadas simultaneamente, e que seja possível a implementação de outras formas de se comunicar com o servidor que não sejam apenas a interface gráfica principal.

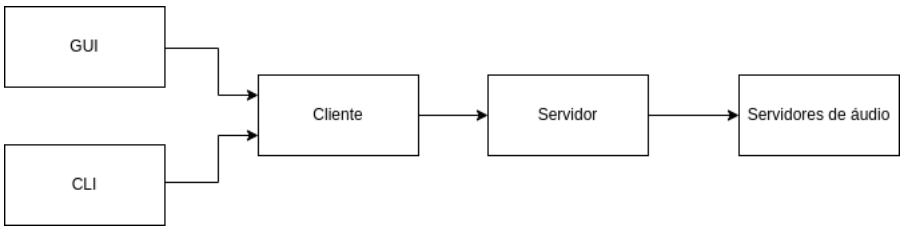

Figura 22 – Arquitetura do sistema, com uma clara separação entre clientes e servidor

3.1.2 Gerenciamento de grupos

No contexto desta aplicação, existem dois tipos de grupos, que são grupos de entradas e grupos de saídas. Passaremos a chamá-los de entradas virtuais e saídas virtuais.

Uma entrada virtual é vista pelo sistema de som da mesma maneira que uma saída física, porém não representa nenhum dispositivo real. Seu objetivo é capturar o áudio de aplicações misturando-as em uma nova fonte, que pode ser mixada em qualquer saída física ou virtual. São chamadas de entradas pelo fato de que pela perspectiva da

aplicação, elas recebem áudio, sendo necessário fornecê-lo a uma saída física ou virtual para que tenham uma utilidade.

Uma saída virtual é vista pelo sistema de som da mesma maneira que uma entrada física, basicamente um microfone virtual. A única função de uma saída virtual é que o áudio de outra fonte seja transmitido para ela, permitindo que aplicações a para que aplicações a usem como um microfone.

Mesas de áudio tem um número fixo de grupos, mas como esta aplicação é virtual, o número de grupos não é uma limitação, então seria interessante que o usuário possa criar, excluir e editar os grupos, tanto de entrada quanto de saída. Cada grupo deve ter um nome único para identificação, número de canais e o volume de cada canal.

3.1.3 Mixagem do áudio de entradas e de aplicações

Todo o núcleo da aplicação se baseia na ideia de agrupamento de fontes em dispositivos virtuais, porém dispositivos virtuais sozinhos não são capazes de nada, já que não representam nenhum objeto no mundo real. No caso das entradas virtuais, seu áudio deve ser mixado em alguma saída física ou virtual. No caso de uma saída virtual, seu objetivo é receber o áudio das entradas físicas e virtuais, e ser utilizada como um microfone pelas aplicações. O usuário deve ser capaz de transmitir o áudio de entradas físicas e virtuais para uma ou mais saídas a partir de uma interação simples como o clique de um botão. Também seria interessante a possibilidade de escolher quais canais devem ser conectados em cada dispositivo, e que não seja necessário seleciona-los toda vez que for realizar a transmissão.

3.1.4 Transmissão de áudio por rede

Algumas aplicações como Voicemeeter possuem maneiras de se transmitir áudio através a rede, seria interessante a possibilidade de transmitir áudio através de instâncias diferentes em máquinas diferentes, ou então algo nos entornos do que o Pulseaudio faz e permitir que o usuário se conecte em uma seção remota.

3.1.5 Controle de volume

Um dos pontos mais importantes e difíceis durante a tarefa de mixagem, é encontrar o balanço correto entre o volume das fontes, e é muito fácil perceber quando o volume de algo não está configurado corretamente. A possibilidade de alteração de volume das fontes embora não seja o objetivo principal, ainda é um dos requisitos mais importantes de serem implementados, devido ao fato de que seria extremamente inconveniente ter que usar outra aplicação apenas para controlar o volume. Cada fonte, seja ela física, virtual, ou uma aplicação deve ter seu próprio controle de volume.

3.1.6 Medidor de pico de volume (VUmeter)

Cada uma das fontes de áudio, sejam elas físicas ou virtuais devem ter um medidor de pico de volume, para o usuário possa ter um indicativo visual do áudio de cada fonte. É imprescindível que seja qual for a implementação escolhida para este função, que seja a mais otimiza possível, pois por se tratarem de muitas fontes de áudio apresentadas simultaneamente na tela, o custo para atualizar a interface gráfica é bem alto, necessitando que o algoritmo seja o mais rápido possível e evite cálculos desnecessários.

3.1.7 Uso de Efeitos

É comum no meio de produção de áudio a utilização de efeitos para modificar o áudio original. Existem uma vasta gama de efeitos que podem ser utilizados no áudio como supressores de ruido, equalizadores, limitadores, moduladores. Em alguns casos mesmo com bons microfones ainda é necessária a utilização efeitos para mitigar problemas externos como ruídos, barulhos indesejados e sons muito altos.

3.1.8 Interfaces

Tendo em vista a separação entre clientes e servidor, surge a possibilidade de criação de tipos diferentes de interfaces, cada uma com um propósito distinto, sendo que as opções mais interessantes são:

- Interface Gráfica
- Interface Midi
- Interface em linha de comando

Interface gráfica

A interface gráfica deve ser composta por elementos iguais a um Mixer real como botões, sliders, faders e knobs. A GUI do Voicemeeter(1) é um bom ponto de partida para a criação de algo novo. Voicemeeter recebe algumas críticas por sua interface não ser intuitiva para novos usuários, já que possui uma grande quantidade de informações dispostas simultaneamente e sem muita explicação (além do manual). Deve-se haver uma separação clara entre os tipos de dispositivos, agrupando-os em blocos distintos para não haver confusão. Esta separação é necessária pelo fato de que cada tipo se comportar de maneira diferente, com funções diferentes. A funcionalidade de criar e remover dispositivos deverá estar contida nestes blocos. Cada bloco será constituído de uma lista de dispositivos. Cada dispositivo será composto dos elementos que manipularão esta fonte de áudio como botões e sliders.

Interface Midi

Como a possibilidade do áudio ser controlado por um servidor, é possível a criação de uma interface de comunicação MIDI para executar comandos. Ela pode funcionar de forma que mapeia as notas e os sinais de controle enviados pela interface MIDI à ações no servidor, podendo também realizar o processo inverso, recebendo mensagens do servidor para ligar Leds na interface.

Interface em linha de comando

A implementação de uma interface em linha de comando para Linux é algo desejável, pelo fato de que além de que isto permitiria controlar o servidor usando simples comandos, também se criaria a possibilidade que eles ativados por ações como por exemplo teclas de atalho, ou em formas de scripts para executar várias operações simultaneamente.

3.1.9 Servidores de som

Com a existência de servidores de áudio com escopos diferentes no Linux, seria interessante a possibilidade do controlador de áudio ser agnóstico em relação a qual API de áudio está sendo utilizada, deixando o este trabalho para uma camada mais baixa da aplicação, permitindo ao usuário a escolha de qual servidor será usado.

4 Desenvolvimento

Pulsemeeter é uma mesa de áudio virtual, desenvolvida com o intuito de ter um fluxo de trabalho baseado na ideia de agrupamento de fontes, permitindo uma abstração do controle de múltiplas aplicações simultaneamente, especificamente tendo em mente os problemas citados na seção anterior.

4.1 Primeira versão

A primeira versão foi desenvolvida como uma prova de conceito para implementar um mixer virtual semelhante ao Voicemeeter, devido ao fato de se tratar de um software proprietário e exclusivo para Windows. Esta primeira versão foi implementada em Python, utilizando a biblioteca Gtk para a construção da interface gráfica, que se limitava a trabalhar com no máximo 12 dispositivos (3 entradas virtuais, 3 entradas físicas, 3 saídas físicas e 3 saídas virtuais), não possuía indicadores de pico de volume, uso de efeitos, e nem controles individuais para as aplicações, como pode ser visto na figura 23.

A interface gráfica na primeira versão utilizava diretamente a interface em linha de comando do Pulseaudio para realizar as operações. Cada dispositivo físico possui uma caixa de seleção em que é possível escolher qual dispositivo real será usado, semelhante a ideia de um mixer de conectar uma das saídas em uma caixa de som ou um fone de ouvido.

Cada dispositivo virtual possui um nome único usado para identificação, e cada dispositivo de entrada (virtual e físico) possui 6 botões: A1, A2, A3, B1, B2, B3. Cada um desses botões representa uma saída, os que iniciam com "A"são saídas físicas, as que iniciam com "B"são saídas virtuais.

A mixagem das fontes era realizadas utilizando o módulo loopback do Pulseaudio. Após a comprovação de que seria sim possível a implementação de um software com estes requisitos, iniciou-se de desenvolvimento da segunda versão.

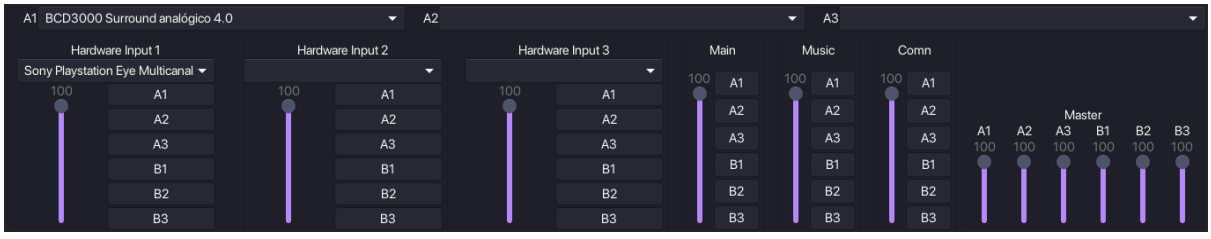

Figura 23 – Primeira versão da aplicação, a interface era extremamente similar a do Voicemeeter

4.2 Segunda versão

Como a primeira versão era apenas um prova de conceito, para a segunda versão foi necessário recriar todo o sistema do zero. Algumas novas funcionalidades foram adicionadas, e algumas antigas foram modificadas. Foram introduzidos medidores de pico de volume, redutores de ruídos para entradas físicas, equalizadores para as saídas físicas e virtuais, e controle individual das aplicações, como pode ser visto na figura 24. Notou-se então que o sistema era extremamente dependente da interface gráfica, pois todas as operações eram processadas e realizadas por ela, e que para remover esta restrição, seria necessário uma refatoração completa do sistema.

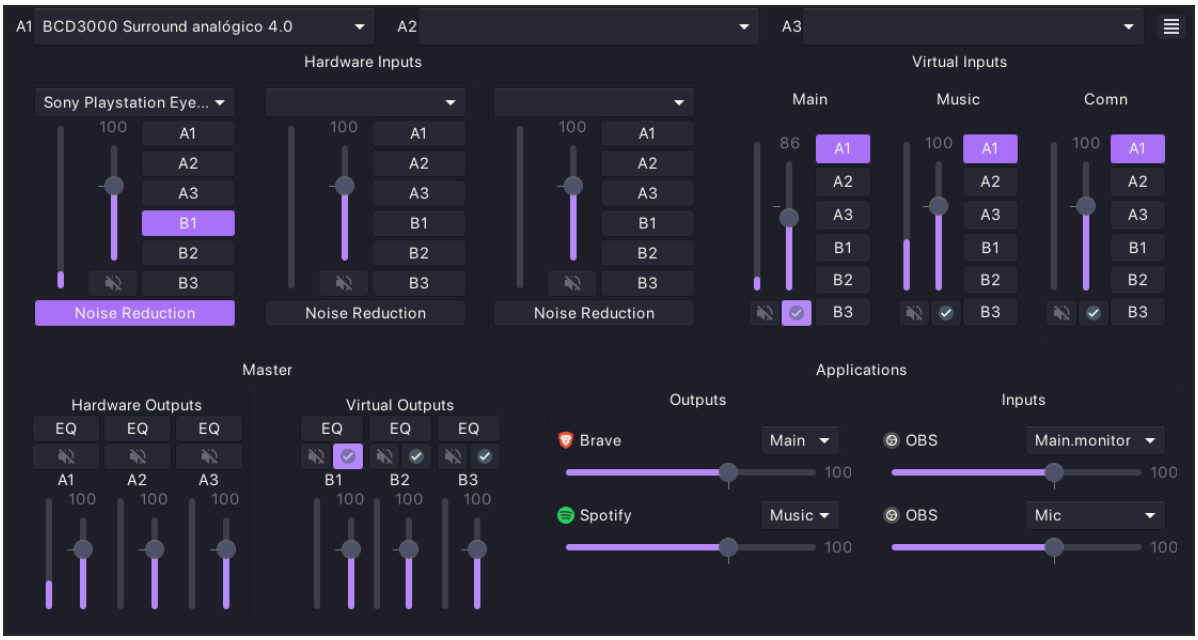

Figura 24 – Segunda versão da aplicação, a interface ainda era similar a do Voicemeeter, porém já incluía controles para cada aplicação

4.3 Terceira versão

Como as primeiras versões eram completamente dependentes da interface gráfica, foi necessário repensar a forma como ela e o servidor de áudio se comunicam. O sistema permaneceu visualmente o mesmo, não foram realizadas grandes modificações que afetassem a experiência de usuário.

Apesar disto, a maneira que ela realiza as operações mudou drasticamente, pois agora o controlador de áudio se tornou um servidor, que utiliza uma Unix Socket(12) para se comunicar com múltiplos clientes conectados simultaneamente. Isso permitiu que múltiplas instâncias da interface gráfica fossem utilizadas ao mesmo tempo, além de possibilitar que outros tipos de interfaces independentes fossem criadas.

Algumas questões continuavam existindo, como as versões anteriores que eram limitadas a trabalhar com o máximo 12 dispositivos. Também existia o problema de que todas as operações de controle no servidor eram realizadas por Shell Scripting, o que acabava sendo prejudicial para o desempenho da aplicação. Além de que o módulo de loopback do Pulseaudio introduzia uma latência mínima de 200ms, o que não era nem um pouco agradável.

4.4 Versão atual

Tendo em vista todos os problemas remanescentes das versões anteriores, esta nova versão da aplicação precisou reformular a maneira que a interface gráfica funciona. Esta versão remove completamente o limite imposto anteriormente a 12 dispositivos, permitindo que eles sejam criados e removidos a qualquer momento, como pode ser visto na figura 25.

Esta nova versão também contava com suporte ao servidor de áudio Pipewire, que foi usado para realizar a mixagem de fontes ao invés do módulo loopback do Pulseaudio (embora ainda use esse módulo caso Pipewire não esteja disponível), agora sendo possível mapear individualmente as portas de um dispositivo, permitindo que canais possam ter volume próprio e sejam individualmente mixados.

Também passou-se a utilizar a biblioteca pulsectl, que é a implementação em python da biblioteca libpulse do Pulseaudio, reduzindo a dependência do sistema em Shell Scripting, embora algumas partes ainda dependam dele. Ao contrário das versões anteriores, agora os dispositivos de saída não são mais identificados com a nomenclatura A e B, mas sim por apelidos dados pelos usuários.

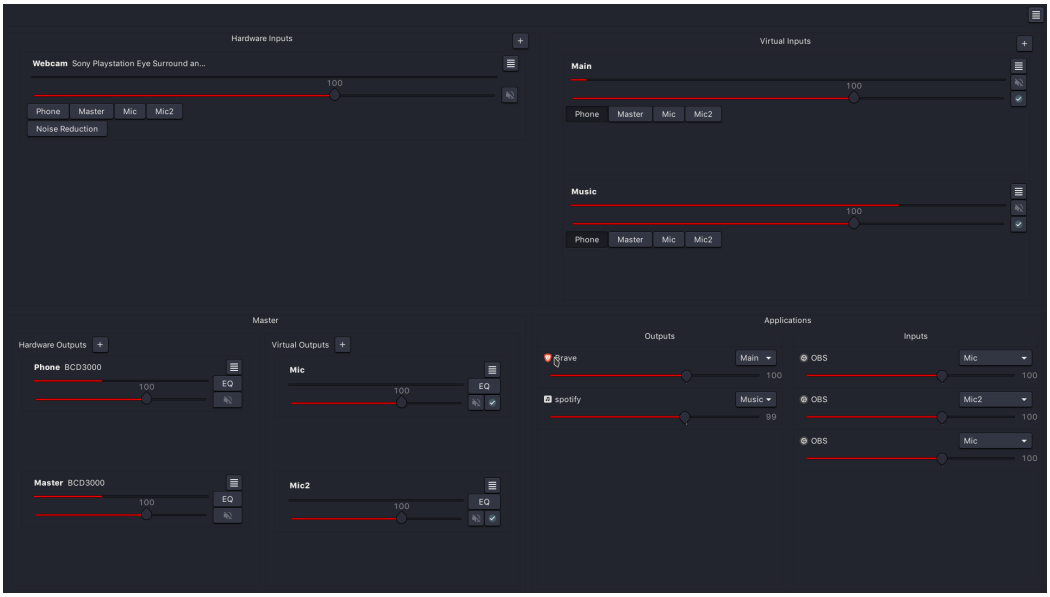

Figura 25 – Versão atual da aplicação, a interface se distanciou bastante da do Voicemeeter

5 Conclusão e trabalhos futuros

O presente trabalho detalhou o desenvolvimento das etapas para o estado atual da aplicação Pulsemeeter. O projeto foi iniciado sem a pretensão de se tornar um trabalho de conclusão de curso, estava apenas tentando solucionar um problema que tive desde o momento que migrei de Windows para Linux: a ausência de uma aplicação similar ao Voicemeeter. No inicio deste processo de migração aprendi a como utilizar a API em linha de comando do pulseaudio para realizar operações similares as que precisava. No meio da pandemia de COVID decidi pesquisar novamente se alguém havia implementado alguma aplicação com estes moldes, pois ainda estava interessado no assunto, quando percebi que não havia nenhuma. Decidi então que havia aprendido o suficiente (não era verdade) sobre Pulseaudio e que poderia desenvolver uma aplicação do gênero. Após a implementação da primeira versão, decidi então divulgar o resultado no Reddit, e obtive respostas bem animadoras de pessoas interessadas na aplicação, que testaram e deram sugestões para versões futuras. Com o decorrer do desenvolvimento de novas versões, novas pessoas se interessavam pelo projeto e se engajavam, assim foi-se criando uma comunidade ao redor dele.

A aplicação desenvolvida possui o código aberto e disponibilizado gratuitamente para qualquer pessoa acessar no GitHub. Atualmente conta com alguns contribuidores ao redor do mundo, e é buscada streamers, gamers e músicos migrando do Windows para sistemas Linux, por ser uma alternativa viável ao Voicemeeter. Foi criado um servidor de suporte à aplicação no Discord, no qual diversas pessoas utilizam para relatar problemas, sugerir novas funcionalidades e ajudar pessoas tendo problemas com o software.

O projeto definitivamente foi essencial para meu desenvolvimento pessoal e profissional, aprendi muito sobre diversos aspectos de desenvolvimento de software, software livre, comunidades de software livre e o mais importante de todos, a importância do planejamento no desenvolvimento de um software, pois muitas vezes fiquei sem direções pois não planejava e nem sabia medir corretamente a prioridade dos próximos passos a serem tomados.

5.1 Dificuldades encontradas

Até este momento não tinha tido nenhum tipo de experiência com projetos open source, muito menos comunidades de projetos open source. No inicio tive dificuldades em lidar com a quantidade de sugestões e problemas relatados, pois tentava dar prioridade a todos, e no fim não dava prioridade a nada.

O que mais dificultou na implementação deste trabalho foi a falta de materiais didáticos sobre a biblioteca em C do Pulseaudio, embora a documentação oficial faça um bom trabalho descrevendo as funções e objetos.

5.2 Trabalhos futuros

Alguns dos requisitos levantados na seção de projeto não foram implementadas, embora não essenciais para o funcionamento da aplicação, ainda seriam extremamente interessantes de se implementarem. Existe também a possibilidade da rescrita da aplicação em C++ ou Rust, já que são linguagens bem mais performáticas do que Python, além de existirem bibliotecas do Pulseaudio e Pipewire disponíveis nessas linguagens.

Referências

1 VB-Audio. *Voicemeeter*. Disponível em: <https://www.voicemeeter.co>.

2 Paul Davis. *Jack Audio Connection Kit*. Disponível em: <https://jackaudio.org>.

3 SCHIAVONI, F. L.; GOULART, A. J. H.; QUEIROZ, M. Apis para o desenvolvimento de aplicações de áudio. *Seminário Música Ciência Tecnologia*, v. 1, n. 4, 2012.

4 BISTAFA, S. R. *Acústica aplicada ao controle do ruído*. [S.l.]: Editora Blucher, 2018.

5 Digital Brain Instruments. *CMixer*. Disponível em: <https://www.digitalbrain-instruments.com/cmixe>.

6 LoudLab. *SoundDesk*. Disponível em: <https://www.loudlab-app.com/sounddesk-ap>.

7 TRANTER, J. Introduction to sound programming with alsa. *Linux Journal*, Belltown Media Houston, TX, v. 2004, n. 126, p. 4, 2004.

8 FREEDESKTOP. *Pulseaudio*. Disponível em: <https://www.freedesktop.or>.

9 FREEDESKTOP. *Helvum*. Disponível em: <https://gitlab.freedesktop.org/pipewire-/helvu>.

10 TAYMANS, W. *Pipewire-jack*. 2021. https://gitlab.freedesktop.org/pipewire/pipewire/-/wikis/JACK. Acessado: 2022-12-01.

11 Richard Furse. *LADSPA*. Disponível em: <https://www.ladspa.or>.

12 ANôNIMO. *unix(7) Linux User's Manual*. 6.01. ed. [S.l.], Outubro 2022.