UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI
DEPARTMENTO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RÔMULO AUGUSTO VIEIRA COSTA

Sunflower: uma proposta de padronização para ambientes de Internet das Coisas Musicais

São João del-Rei
2021

RÔMULO AUGUSTO VIEIRA COSTA

Sunflower: uma proposta de padronização para ambientes de Internet das Coisas Musicais

Dissertação apresentada ao programa de Pós-Graduação em Ciência da Computação da UFSJ para obtenção do título de mestre em Ciência da Computação.

Flávio Luiz Schiavoni
Orientador

Daniel Ludovico Guidoni
Convidado 1

Filipe Carlos de Albuquerque Calegario
Convidado 2

Marcelo Soares Pimenta
Convidado 3

São João del-Rei
2021

Dedico este trabalho aos meus pais
pelo amor, carinho, afeto, dedicação e
cuidado que me deram durante toda a vida.

AGRADECIMENTOS

Meu interesse pela vida acadêmica começa em um ponto muito específico no tempo: quando fui cursar o ensino médio no Colégio Potência. Lá, recebi a orientação de grandes professores, como Érika Cerqueira, Érica Castro, Maria Gorete, Maria Elizabeth Vasconcelos e Sandro Nascimento. Nada disso teria sido possível sem a intervenção de Dona Lilinha (*in memoriam*) para que eu conseguisse uma bolsa de estudos naquele colégio. Bolsa esta que foi mantida posteriormente pelo professor Edelvando. A estes, meu eterno carinho e gratidão.

No ensino superior, entre indas e vindas, encontrei o meu lugar na Universidade Presidente Antônio Carlos, onde tive o prazer de trabalhar com excelentes professores, como Jim Marciano, Alfredo Ganime, Eugênio Pacelli, Jean Carlo Mendes e Alex Vitorino. Estes últimos dois que muito me incentivaram a seguir na vida acadêmica e me auxiliaram para que isso fosse possível. Meu agradecimento a eles.

Ao entrar no curso de mestrado, conheci excelentes pessoas no laboratório ALICE (Arts Lab in Interfaces, Computers, and Everything Else), como Jônatas Araújo, Luan Gonçalves, Lucas Estevão, Rafael Andrade, Mauro César, Gabriel Rocha, Gabriel Carneiro, Mariana Lellis, Avner Maximiliano, Igor Andrade, Carlos Eduardo de Souza, Fábio Passos, Igino Júnior, Thiago Morandi, Isadora Franco, João Lara, João Pedro Mendes, Tales Boratto, Rebeca Lima, Júlio César de Sousa, Luiz Gustavo Colzani, Samuel Rabay e tantos outros. Peço perdão se esqueci alguém, mas vocês fizeram este caminho ser mais leve e divertido.

Ainda sobre o ALICE, agradeço especialmente ao João Araújo, que muito me ajudou na adaptação ao curso e à universidade. Dedico uma menção especial ao meu orientador, Flávio Schiavoni, pelos conselhos, dicas, paciência e conversas intermináveis. Tudo isso contribuiu para minha formação como pesquisador e cientista, mas contribuiu ainda mais para que eu me tornasse uma pessoa melhor.

Agradeço a todos os professores do PPGCC, em especial aqueles que tive o prazer de trabalhar: Edimilson Batista, Daniel Guidoni, Vinícius Durelli, Carolina Xavier e Marcos Laia. Agradeço também a toda a secretaria, pela agilidade e presteza incondicional.

Aos amigos “De Fé", Iago, Matheus, Milady e Lara, meus eternos agradecimentos pelo apoio e aventuras de sempre.

Todo este caminho foi percorrido com a melhor trilha sonora possível, graças a Lucas Silveira, Rodrigo “Esteban” Tavares, Tyler Joseph, Josh Dun e Joe Mulherin. Minha gratidão por isso.

Reservo os melhores agradecimentos a minha família. Minha querida irmã e melhor amiga, Rosiane, por ser minha maior inspiração, e aos meus pais, Ronaldo e Rosângela, por todo o amor, carinho, apoio irrestrito e por superarem todas as adversidades para que eu pudesse realizar meus sonhos. Há muito de vocês em tudo que eu faço. “Tudo que nós tem é nós".

Este trabalho não seria possível sem o apoio financeiro da CAPES. Agradeço à fundação e reforço a importância do investimento em educação e ciência.

A todos que contribuíram para a expansão do meu conhecimento, minha eterna gratidão.

Nenhum agradecimento ao ano de 2020.

Venceremos!

“Vão tentar derrubar que é pra me ver crescer
E às vezes me matar que é pra eu renascer
Como uma supernova que atravessa o ar
Eu sou a maré viva, se entrar, vai se afogar”.
(Fresno)

RESUMO

A Internet das Coisas Musicais é uma área do saber posicionada entre a Internet das Coisas, novas interfaces para expressão musical, música ubíqua, inteligência artificial, arte participatória e interação humano-computador. Ela se propõe a melhorar a relação entre musicistas e seus pares, bem como a de musicistas e membros da audiência, favorecendo concertos, produções em estúdio e o aprendizado de música. Apesar de emergente, este campo já se depara com alguns desafios, como aqueles sociais, econômicos e ambientais que são decorrentes da inserção de um novo tipo de tecnologia na sociedade, além das instigações causadas nas práticas artísticas e pedagógicas. De um ponto de vista computacional, as adversidades recaem sobre a falta de privacidade e segurança, e principalmente, na falta de padronização e interoperabilidade entre os seus dispositivos. Sendo assim, o presente trabalho apresenta o *design* de ambiente, chamado Sunflower, destacando sua arquitetura, protocolos e características sonoras que podem contribuir para solucionar os problemas mais recorrentes a esta área. Para validação técnica, testes foram realizados em *localhost*, conexão cabeada de par trançado e conexão sem fio via Wi-Fi. Ao fim de tudo, é realizada uma análise comparativa com outros três modelos já existentes, de modo a tirar conclusões sobre quais comportamentos e protocolos são recorrentes nesta área, além de indicar particularidades que podem ajudar desenvolvedores a criarem seus próprios cenários.

Palavras-chaves: Internet das Coisas Musicais, *Design* de ambiente, definição de protocolo, Pipes-and-Filters, Sunflower, Performance musical em rede

ABSTRACT

Title: Sunflower: a proposal for standardization for Internet of Musical Things environments

The Internet of Musical Things is an area of knowledge positioned between the Internet of Things, new interfaces for musical expression, ubiquitous music, artificial intelligence, participatory art, and human-computer interaction. It aims to improve the relationship between musicians and their peers, as well as that of musicians and audience members, favoring concerts, studio productions, and music learning. Although emerging, this field is already facing some challenges, such as social, economic, and environmental ones that result from the insertion of a new type of technology in society, in addition to the instigation caused in artistic and pedagogical practices. From a computational point of view, the adversities fall on the lack of privacy and security, and mainly, on the lack of standardization and interoperability between its devices. Therefore, this work presents the environment design, called Sunflower, highlighting its architecture, protocols, and sound characteristics that can contribute to solving the most recurrent problems in this area. For technical validation, tests were performed on localhost, wired twisted pair connection, and wireless connection via Wi-Fi. Finally, a comparative analysis is performed with three other existing models, in order to draw conclusions about which behaviors and protocols are recurrent in this area, besides indicating particularities that can help developers to create their own scenarios.

Key-words: Internet of Musical Things, Environment design, Protocol definition, Pipes-and-Filters, Sunflower, Network Music Performance

LISTA DE ILUSTRAÇÕES

Figura 1 – Possíveis interações em um ambiente de Internet das Coisas Musicais. . . . 16
Figura 2 – Estrutura clássica de um IMD. 24
Figura 3 – Sistema básico de uma performance musical em rede. 25
Figura 4 – Arquitetura básica do modelo Pipes-and-Filters. 31
Figura 5 – Atraso entre pacotes que compõem o mesmo bloco de informação, caracterizando a ocorrência do *jitter*. 34
Figura 6 – Primeiro ambiente de IoMusT analisado. 37
Figura 7 – Comportamento prático dos filtros. 46
Figura 8 – Protocolo de comunicação do Sunflower. 57
Figura 9 – Divisão em camadas do Sunflower. 58
Figura 10 – Trecho responsável por capturar o áudio de um microfone e enviá-lo pela rede. 60
Figura 11 – Encapsulamento de informações MIDI em pacotes OSC. 60
Figura 12 – *Patch* responsável pelo vídeo no Sunflower. 61
Figura 13 – Artes presentes na camada gráfica. 62
Figura 14 – CLI do Sunflower e suas principais funcionalidades. 63
Figura 15 – CLI em operação, contendo mensagens que serão interpretadas e utilizadas pelo administrador para conectar ou desconectar determinadas coisas musicais. 64
Figura 16 – Elementos criados para o MobMuPlat. 65
Figura 17 – Logo do Sunflower, uma proposta de *design* para ambientes de Internet das Coisas Musicais. 74

LISTA DE TABELAS

Tabela 1 – Medições em *localhost*. 68
Tabela 2 – Medições no cabo de par trançado. 69
Tabela 3 – Medições no Wi-Fi. 70
Tabela 4 – Síntese das arquiteturas de Internet das Coisas Musicais. 77

LISTA DE ABREVIATURAS E SIGLAS

5G	*Fifth Generation*
API	*Application Programming Interface*
ASCII	*American Standard Code for Information Interchange*
AVI	*Audio Video Interleave*
BPM	*Beats Per Minute*
CD	*Compact Disc*
CLI	*Command Line Interface*
CPU	*Central Processing Unit*
DVD	*Digital Versatile Disc*
DSSS	*Direct Sequency Spread Spectrum*
GEM	*Graphics Environment for Multimedia*
IEEE	*Institute of Electrical and Electronics Engineers*
IMD	Instrumento Musical Digital
IoT	*Internet of Things*
IoMusT	*Internet of Musical Things*
IP	*Internet Protocol*
IRCAM	*Institut de Recherche et Coordination Acoustique/Musique*
LTS	*Long-term support*
MIDI	*Musical Instrument Digital Interface*
MP3	*MPEG-1/2 Audio Layer 3*
MPEG	*Moving Picture Expert Group*
OSC	*Open Sound Control*
PCM	*Pulse-code modulation*
Pd	*Pure Data*
PVC	*Policloreto de Vinila*

RAM	*Random Access Memory*
RFID	*Radio-frequency identification*
RFC	*Request for Comments*
RJ-45	*Registered Jack-45*
SMI	*Smart Musical Instrument*
STP	*Shielded Twisted Pair*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
UTP	*Unshielded Twisted Pair*
WAV	*Waveform Audio File Format*
Wi-Fi	*Wireless Fidelity*
WLAN	*Wireless Local Area Network*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO 14
1.1 Justificativa 16
1.2 Objetivos 17
1.3 Metodologia 18
1.3.1 Revisão bibliográfica 18
1.4 Organização do trabalho 19
2 TERMINOLOGIAS DO TRABALHO 20
2.1 Internet das Coisas 20
2.2 Internet das Coisas Musicais 20
2.2.1 Coisas Musicais 21
2.2.2 Campos relacionados 24
2.3 Sistemas distribuídos 26
2.3.1 Modelo Cliente-Servidor 28
2.4 Pipes-and-Filters 29
2.5 Meios de Transmissão pelas Redes 31
3 TRABALHOS RELACIONADOS 36
4 DESIGN DE UM AMBIENTE DE INTERNET DAS COISAS MUSICAIS 40
4.1 Características desejáveis 40
4.1.1 Características desejáveis para o ambiente 41
4.1.2 Características desejáveis para os dispositivos 42
4.2 Pensando em Pipes and Filters 44
4.3 Pensando em camadas 46
4.3.1 Camada de Áudio Digital 47
4.3.2 Camada Gráfica 48
4.3.3 Camada de Controle 49
4.3.4 Camada de Gerenciamento 50
4.4 Proposta de Protocolo 52
5 IMPLEMENTAÇÃO E TESTES 59
5.1 Implementando o *design* do Sunflower e utilizando o Pure Data 59
5.2 Utilizando o Processing 61
5.3 Utilizando Python 62
5.4 Utilizando o MobMuPlat 64
5.5 Testes práticos 65
5.5.1 O ambiente de testes 66
5.5.2 Testes em Localhost 67
5.5.3 Testes em cabo de Par trançado 67
5.5.4 Testes no Wi-Fi 68

6 DISCUSSÃO E RESULTADOS 71
6.1 Resultados dos testes 71
6.2 Sobre as características desejáveis 72
6.3 Reflexões sobre os protótipos desenvolvidos 74
6.4 Comparações com os trabalhos relacionados 76
6.5 Possíveis cenários de uso 77
6.5.1 Cenário 1: Uma *jam session* que combina elementos musicais tradicionais e coisas musicais 77
6.5.2 Cenário 2: Gravação em estúdio inteligente 78
6.5.3 Cenário 3: Apresentações ao vivo 78
7 CONCLUSÕES 80
7.1 Trabalhos Futuros 83
REFERÊNCIAS 85

1 INTRODUÇÃO

A prática musical é uma atividade inerente à cultura humana, permeando-a desde seus primórdios. Ao longo dos séculos, esse tipo de arte esteve suscetível às mais diversas mudanças, muito por conta das novidades tecnológicas que acompanhavam cada época. As mais notáveis são referentes às formas de gravação e reprodução do conteúdo sonoro, marcadas por quatro eras distintas (BURGESS, R. J, 2014; MILLARD, A, 2005). A primeira delas é chamada de "Era acústica" (1877 - 1925), caracterizada pelo uso exclusivo de dispositivos mecânicos para gravação.

Em seguida, veio a "Era elétrica" (1925 - 1945), que a partir do uso da eletricidade, como sugere o nome, permitiu a criação de novos artefatos, como microfones, amplificadores e gravadores. Esses equipamentos propiciaram as primeiras transmissões de rádio, favorecendo o surgimento de gêneros da música popular, como jazz, blues e rock and roll. Na década de 1940, a introdução do disco de vinil marcou o início da indústria fonográfica, responsável pela gravação, promoção e distribuição de músicas. Criou-se então uma nova lógica de mercado, onde os fonogramas eram comercializados na forma de *singles* e álbuns, com título, capa e encarte, padrão que persiste até hoje.

Após a Segunda Guerra Mundial, começou-se a usar fitas magnéticas para a gravação de áudio, iniciando assim a "Era magnética" (1945 - 1975). A criação da fita cassete, na década de 1960, talvez tenha sido o fato mais marcante desse período.

Por fim, veio a "Era digital", iniciada em 1975 e que perdura até os dias atuais. Ela alterou de forma rápida e dramática a maneira como a música é criada e consumida ao substituir o som analógico pela codificação digital. O surgimento do CD foi o marco inicial da digitalização nessa indústria. A maior mudança, no entanto, ocorreu na década de 1990, com a difusão da internet e compressão acústica. O formato MP3 se popularizou e eliminou a necessidade de uma mídia física para comportar os arquivos musicais.

A partir da virada do século, as diversas transformações tecnológicas continuaram impactando o mundo. Dentre elas, pode-se citar a expansão da internet banda larga e o barateamento de mídias de armazenamento de dados e dos produtos eletrônicos, o que culminou no surgimento e expansão da Internet das Coisas, cujo conceito básico é definido como:

> "*A presença generalizada ao nosso redor de uma variedade de objetos, que através de endereços exclusivos, são capazes de interagir entre si e cooperar com seus vizinhos para alcançar objetivos comuns*[1]*"* (ATZORI; IERA; MORABITO, 2010).

Quando os domínios da IoT são expandidos para a prática musical, surge a Internet

[1] *"The widespread presence around us of a variety of objects, which through exclusive addresses, are able to interact with each other and cooperate with their neighbors to achieve common goals".*

das Coisas Musicais. Esta área é caracterizada por ser multidisciplinar, englobando conceitos oriundos da música ubíqua (KELLER; LAZZARINI; PIMENTA, 2014), música móvel (GAYE et al., 2006), inteligência artificial (BURGOYNE; FUJINAGA; DOWNIE, 2016), interação humano-computador (ROGERS; SHARP; PREECE, 2011) e outros campos da computação. Ela pode ser descrita como:

> "*Conjunto de interfaces, protocolos e representações de informações relacionadas com a música, que possibilitam serviços e aplicações com uma finalidade artística a partir de interações entre humanos e coisas musicais ou entre estas mesmas coisas musicais em meios físicos e/ou digitais*[2] *(...)*" (TURCHET et al., 2018).

Uma coisa musical, por sua vez, pode ser definida como um dispositivo eletrônico capaz de adquirir, processar, atuar ou trocar dados que sirvam a um propósito musical (TURCHET et al., 2018). Uma possibilidade para criar este aparelho é utilizar um equipamento de uso comum na IoT para a prática musical. Outra perspectiva é adaptar instrumentos, equipamentos de áudio (microfones, mesas de som, alto-falantes, etc.), unidades de efeito (pedais de guitarra e *plugins*) e ferramentas que auxiliam na apresentação (como canhões de luz e outros dispositivos de iluminação) para serem usados com conexão à rede. Para que isso ocorra, é desejável atribuir-lhes a capacidade de inserção ou remoção de botões e sensores, de modo que os usuários possam adaptá-las para o novo ambiente que se vislumbra diante deles. Além das mudanças físicas, alterar o código de controle e o *software*, quando presentes, também pode permitir a adequação de um determinado dispositivo a um novo contexto. Ainda, algumas funcionalidades podem ser incorporadas para transformar seu uso em uma atividade coletiva (VIEIRA; GONÇALVES; SCHIAVONI, 2020).

A junção de dispositivos com capacidade de conexão a rede com serviços e aplicações de cunho musical criam um ambiente interoperável, responsável por interligar músicos, instrumentos e membros da audiência, o que multiplica as possibilidades de interação em espetáculos artísticos e cria uma relação de interdependência, como pode ser observado na Figura 1.

A partir desta ilustração, observam-se as inúmeras possibilidades de relacionamento entre musicistas, membros da audiência e coisas musicais. A comunicação pode ocorrer de duas maneiras distintas. A primeira delas é co-localizada, onde os participantes estão no mesmo espaço físico, como, por exemplo, casas de show, teatros e espaços públicos. Esse comportamento é indicado pelas setas negras. Já as setas marrons são responsáveis por retratar o diálogo entre artista e público, enquanto setas roxas traduzem a comunicabilidade entre o intérprete e as coisas musicais (representadas pelo ícone de comunicação sem fio), e finalmente, as setas cinzas denotam a relação entre membros da audiência. Essas trocas de informações se dão a partir de métodos de endereçamento como multicast, *broadcast* ou ponto-a-ponto.

[2] "*The ensemble of interfaces, protocols and representations of music-related information that enable services and applications serving a musical purpose based on interactions between humans and Musical Things or between Musical Things themselves, in physical and/or digital realms (...)*".

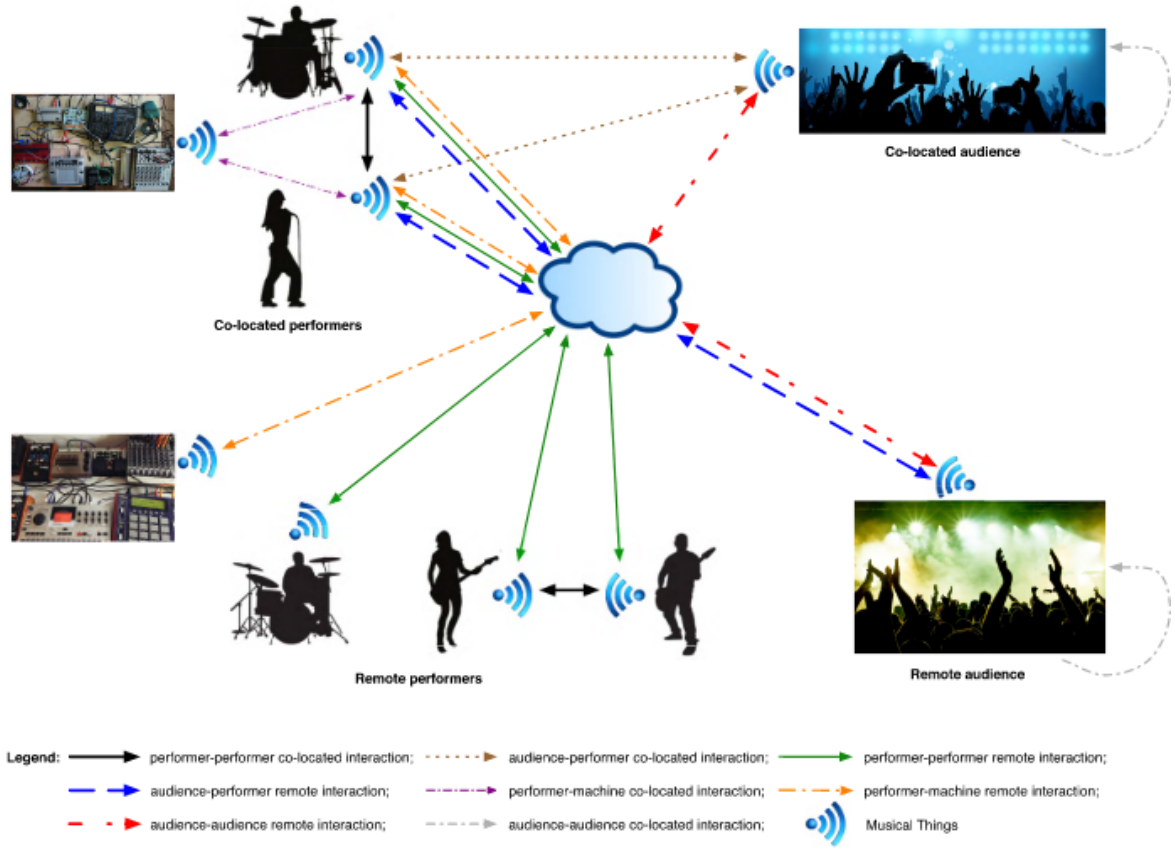


Figura 1 – Possíveis interações em um ambiente de Internet das Coisas Musicais.
Fonte: (TURCHET et al., 2018)

A segunda maneira de comunicação se dá de forma remota, quando os usuários estão em partes distintas do globo. Aqui é possível visualizar as mesmas quatro interações que ocorrem na comunicação co-localizada, representadas respectivamente pelas setas verdes, azuis, amarelas e vermelhas.

Embora não estejam presentes na Figura, o sistema pode comportar também engenheiros de som, produtores musicais, compositores, maestros e educadores. É arquitetado para funcionar tanto em pequena escala, como em apresentações em uma sala de ensaios, quanto em cenários maiores, como festivais, conseguindo reunir milhares de usuários em um ambiente virtual.

1.1 Justificativa

Apesar de ser uma área com grande potencial artístico e computacional, a IoMusT carece de padronização nos seus ambientes, a exemplo da Internet das Coisas, muito por conta de também ser uma novidade. Em razão disso, surgiram alguns trabalhos na área para atacar este problema. Entretanto, eles são fragmentados, focando majoritariamente em como se dá a comunicação pela rede e como são as estruturas das coisas musicais, deixando de lado

uma discussão sobre o controle em alto nível desses dispositivos. Além disso, a maioria dos ambientes utiliza conceitos e tecnologias que seguem as preferências de seus desenvolvedores, não permitindo que eles sejam adaptados para os mais variados cenários de uso.

Todas essas dificuldades e anseios motivaram a elaboração de um *design* de ambiente, chamado de Sunflower, que apresenta modo de funcionamento baseado na arquitetura Pipes-and-Filters e que visa atenuar os problemas de falta de interoperabilidade nesta área. O *design* de um sistema, propriamente dito, consiste no processo de definição da estrutura, protocolos, módulos, interfaces e dados que atendam aos requisitos especificados para o cumprimento de determinada tarefa (BURNS, 2018). A arquitetura Pipes-and-Filters (BUSCHMANN et al., 1996), por sua vez, compreende a divisão de trabalhos complexos em subtrabalhos, onde o filtro é um componente computacional responsável por receber dados e, a partir de um ou mais algoritmos, transformá-los e enviá-los para o próximo filtro por meio dos dutos. Uma importante característica desse sistema é que um filtro não conhece as propriedades de seus vizinhos. Desse modo, o ambiente aqui idealizado também deve dividir a comunicação em subtarefas, onde cada ator envolvido terá seu microssistema musical que posteriormente irá se comunicar com todos os outros, formando um sistema único e heterogêneo.

Quanto às coisas musicais, elas serão análogas aos filtros e compostas de múltiplas entradas e saídas. Devem apresentar ainda métodos para tratar os diferentes tipos de dados que navegarão pela rede.

Vale destacar que este ambiente foi desenvolvido para atender de forma generalizada às requisições da área de IoMusT, permitindo a existência de algumas lacunas em seu escopo.

1.2 Objetivos

Este trabalho tem como objetivo principal elaborar um *design* para um ambiente de IoMusT. Ademais, ele se propõe a atender os seguintes objetivos específicos:

- Sintetizar as características desejáveis para cenários de IoMusT;
- Definir protocolo de uso e arquitetura para este ambiente;
- Criar prótotipos de coisas musicais capazes de gerar áudio e/ou vídeo, emular instrumentos eletroacústicos tradicionais ou expandir suas capacidades (como, por exemplo, permitir que se comuniquem pela rede);
- Criar uma CLI para configurar e gerenciar este ambiente;
- Manter tanto a escalabilidade e heterogeneidade do sistema quanto seu caráter personalizável;
- Avaliar o comportamento da rede;

- Aproximar a prática musical de conceitos comuns na ciência da computação, como redes de computadores;
- Expor, de maneira geral, os conceitos pesquisados e o conhecimento adquirido ao longo do curso de pós-graduação;

1.3 Metodologia

A questão central do trabalho diz respeito a criação e análise de um *design* de ambiente de Internet das Coisas Musicais e comparação com outros ecossistemas, abordando os protocolos, dispositivos e dados que compõem cada um deles, de modo a encontrar padrões e soluções comuns a todos. Para alcançar este objetivo, são usados conceitos de sistemas distribuídos, Internet das Coisas, arquitetura de *software*, novas interfaces para expressão musical, interação humano-computador e performances artísticas. Portanto, é uma pesquisa **aplicada**.

Como a pesquisa apresenta caráter exploratório e busca compreender o comportamento de ambientes de Internet das Coisas Musicais, estudando suas particularidades, motivações e processos, a pesquisa também será **qualitativa**.

A investigação de técnicas e protocolos, estado da arte e justificativas para este trabalho passam por uma **pesquisa bibliográfica seletiva** em materiais publicados em livros, revistas e anais de congressos nacionais e internacionais que tratam do tema principal (IoMusT), assim como dos conceitos periféricos que servem de sustentação para este trabalho.

Por último, uma **pesquisa experimental** e **quantitativa** averigua o funcionamento da rede, a partir de medições feita no *software* Wireshark.

1.3.1 Revisão bibliográfica

Dentre os materiais pesquisados referentes ao campo de Internet das Coisas Musicais, estão: **Internet of Musical Things: Vision and Challenges** (TURCHET et al., 2018), artigo seminal na área; **The Internet of Musical Things Ontology** (TURCHET et al., 2020), que apresenta a ontologia desse campo, ou seja, seus principais conceitos e como eles interagem entre si, e **Everyday Use of the Internet of Musical Things: Intersections with Ubiquitous Music** (VIEIRA; BARTHET; SCHIAVONI, 2020), que trata dos desafios comunicacionais, artísticos e socioambientais enfrentados por esta linha de pesquisa.

Os exemplos de trabalhos que lidam com as coisas musicais são: **The things of the Internet of Musical Things: defining the difficulties to standardize the behavior of these devices** (VIEIRA; GONÇALVES; SCHIAVONI, 2020), que oferecem uma visão geral destes dispositivos, bem como as dificuldades que surgem ao tentar padronizá-los; **Smart Musical Instruments: Vision, Design Principles, and Future Directions** (TURCHET, 2019), que aborda a conectividade destes objetos; **Designing Digital Musical Instruments Using Probatio: A**

Physical Prototyping Toolkit (CALEGARIO, 2019), livro que explora os desafios enfrentados para construir e utilizar tais elementos em apresentações musicais e artísticas; e as teses **Integrando Sensores para a Criação de Instrumentos Musicais Digitais** (MONTEIRO, 2012), que lida principalmente com os problemas funcionais de alguns componentes eletrônicos, e **Instrumentos Musicais Digitais — Uma abordagem composicional** (PATRICIO, 2010), que se ocupa de aspectos da luteria digital e as possibilidades prático-musicais que caracterizam estes instrumentos.

Quanto aos materiais pesquisados referentes aos sistemas distribuídos, *design* de criação e música em rede, destacam-se: **Engineering Distributed Objects** (EMMERICH, 2000), **Sistemas Distribuídos: Conceitos e Projetos** (COULOURIS et al., 2013), **Sistemas Distribuídos: Princípios e Paradigmas** (TANENBAUM; STEEN, 2007), **Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services** (BURNS, 2018), e **Open Sound Control: an enabling technology for musical networking** (WRIGHT, 2005), que aplica o protocolo OSC[3] na prática musical em rede.

1.4 Organização do trabalho

O presente trabalho encontra-se dividido da seguinte maneira: o Capítulo 2 exibe os principais conceitos que permeiam o texto, enquanto o Capítulo 3 apresenta trabalhos relacionados que serviram de base de comparação para o modelo aqui apresentado. O Capítulo 4 define os requisitos, arquitetura, dados e protocolos presentes no ambiente desenvolvido. Capítulo 5 apresenta a implementação do mesmo. Capítulo 6 discute os resultados obtidos após o ambiente ser finalizado e testado, além de realizar uma comparação com sistemas já existentes, debatendo as características de cada modelo. Finalmente, o Capítulo 7 exibe conclusões resumidas acerca do trabalho realizado e indica os projetos futuros a serem desenvolvidos.

[3] Criado em 1997 por Adrian Freed e Matt Wright para controlar algoritmos de síntese sonora. Hoje o OSC é empregado na comunicação entre computadores, sintetizadores de som e outros dispositivos multimédia.

2 TERMINOLOGIAS DO TRABALHO

Para proporcionar ao leitor uma maior compreensão do trabalho, o presente Capítulo exibe os principais conceitos e termos utilizados ao longo do texto, passando por uma contextualização da Internet das Coisas até uma explicação mais aprofundada sobre os aspectos de rede utilizados neste texto.

2.1 Internet das Coisas

A ideia inicial da Internet das Coisas foi desenvolvida por Kevin Ashton (ASHTON, 1999), em 1999, para se referir a uma tecnologia RFID aplicada em cadeias de suprimento, agindo como ponteiros para bancos de dados na internet que continham informações sobre os objetos presentes em um estoque.

Esta formulação durou até 2004, quando Gershenfeld e colaboradores estipularam o conceito de objetos do cotidiano com capacidade de se conectar a uma rede de dados, abordando também a heterogeneidade de dispositivos e a pilha de protocolos usados para comunicação (GERSHENFELD; KRIKORIAN; COHEN, 2004).

A partir disso, a IoT ganhou diversas definições, sempre enfatizando aspectos da rede que os autores consideraram importante para cada caso particular. Na totalidade, elas concordam que esse campo trata da capacidade de objetos, equipamentos, sensores e itens do dia a dia se conectarem à internet, gerando, trocando e consumindo dados a uma miníma intervenção humana (MARSAN, 2015).

De um ponto de vista tecnológico, pode ser vista como resultado de uma junção de tecnologias, onde o objetivo final é facilitar a endereçabilidade, identificação, sensoreamento e atuação de objetos, além de permitir processamento de informação embarcada, e principalmente, melhorar a comunicação e cooperação em rede. Também considera a escalabilidade e interoperabilidade do ambiente, suportando a locomoção geográfica dos objetos e uma comunicação em tempo real entre eles (MATTERN; FLOERKEMEIER, 2010).

Devido à sua versatilidade, começou a adentrar os mais diversos campos de atuação, como gestão da cadeia de suprimentos, rede de energia, saúde, segurança pública (HALLER, 2010) e, mais recentemente, música (TURCHET et al., 2018).

2.2 Internet das Coisas Musicais

Quando a IoT expande seus domínios para a prática musical, ela gera um campo de atuação chamado Internet das Coisas Musicais (TURCHET et al., 2018). Um ecossistema de IoMusT é composto por vários objetos dedicados à produção e/ou recepção de conteúdo musical, provedores de serviços e informações, como *hardwares*, *softwares*, sensores, atuadores,

protocolos de comunicação, APIs e serviços em nuvem, além de instrumentos musicais acústicos, eletrônicos e digitais, e meios para reprodução de som, como alto-falantes. São necessários também aplicativos e serviços para lidar com os usuários presentes nestes ambientes (musicistas, atores, engenheiros de áudio e membros do público) (ALMEIDA et al., 2021).

Sua infraestrutura suporta comunicação multidirecional sem fio, seja ela local ou remota. Como esta área tem propósito artístico e na maioria das vezes trata de informações em tempo real, é de suma importância que esta infraestrutura apresente baixa latência, alta qualidade de áudio e sincronização entre os dispositivos que estão trocando informações. Deve apresentar ainda aspectos que garantam a segurança e privacidade do usuário. Finalmente, esta estrutura pode ser aplicada aos mais diversos cenários de uso, indo desde salas de ensaio até espaços com grandes dimensões, como teatros, estádios e festivais ao ar livre (TURCHET et al., 2018).

As vantagens dessa forma de planejamento passam por uma melhora da conectividade e do fluxo de trabalho do projeto, além de tornar o ecossistema um terreno fértil para o desenvolvimento de novos artefatos criativos. Isso permite que usuários tenham vários pontos em comum e alinhem suas expectativas com a realidade proposta por esse sistema no momento da criação musical.

O uso da IoMusT oferece novas oportunidades para a indústria musical, fomentando aplicações e serviços capazes de explorar uma relação entre elementos digitais e aqueles pertencentes ao mundo físico (TURCHET et al., 2018; VIEIRA et al., 2020).

2.2.1 Coisas Musicais

O campo de Internet das Coisas Musicais prediz uma nova classe de equipamentos conectados na rede que transformam as relações artísticas. Esses equipamentos são chamados de coisas musicais, apresentados e conceituados, de forma breve, no Capítulo 1.

Como elas não são úteis sozinhas, devem estar inclusas em um ecossistema. Além disso, é necessário pensar em dois pontos essenciais para haver interação entre elas. O primeiro deles diz respeito ao comportamento destes dispositivos. Eles devem permitir que algumas alterações ocorram no seu funcionamento a partir de características específicas do ambiente onde estão inseridos. Um exemplo é permitir que adaptem suas condições para se assemelhar a outros elementos encontrados na mesma rede. É importante que o *software* também possa ser atualizado de forma remota, ou até mesmo alterado para realizar uma tarefa diferente daquela para qual foi inicialmente programado. Desta forma, o usuário pode usar o mesmo objeto de diversas maneiras. Esta interface programável não é uma ideia nova na música, estando presente em diversos outros instrumentos musicais, como sequenciadores e baterias eletrônicas.

O segundo ponto é também o mais sensível, pois lida com a troca de dados. Diferentemente dos dispositivos musicais tradicionais, onde o sinal é transferido de um nó para outro de forma direta, aqui eles não estão fisicamente conectados e tampouco são úteis sozinhas, exigindo

que a conexão seja pela rede. Deve-se atentar, portanto, para seus diferentes formatos de dados, sensibilidade à latência e aos demais aspectos desse tipo de comunicação, como possíveis perdas de mensagens, sobrecarga dos equipamentos, interferência nos sinais, etc.

Para além da comunicação das coisas musicais, deve-se destacar também os desafios para sua concepção. Ao contrário dos equipamentos tecnológicos criados para desempenhar tarefas específicas, as coisas musicais devem atender a um propósito geral e se adaptarem a diferentes condições de uso. Processamento, armazenamento, fonte de energia e preço do *hardware* também devem ser considerados, uma vez que estes fatores estão diretamente relacionados com a sua usabilidade em apresentações artísticas e influenciam no preço final do produto (VIEIRA; GONÇALVES; SCHIAVONI, 2020).

Uma coisa musical pode ser representada na figura de um *smart musical instrument* (instrumento caracterizado por apresentar inteligência computacional embarcada, conectividade sem fio, sistema de geração sonora e *feedback* ao usuário) (TURCHET; MCPHERSON; FISCHIONE, 2016); instrumentos musicais digitais, que podem ser utilizados para gerar e controlar conteúdo musical (TURCHET et al., 2018); e equipamentos musicais vestíveis (TURCHET; WEST; WANDERLEY, 2020). Por somente os dois primeiros modelos terem contribuído para o presente trabalho, apenas eles serão detalhados a seguir.

Smart musical instrument

O termo "*smart instruments*" apresenta diversas definições, baseadas nas características individuais de cada instrumento e na experiência de seus criadores. Pesquisadores do IRCAM, por exemplo, atribuem esta nomenclatura a instrumentos com controle de acústica embarcado (MEURISSE et al., 2013; MEURISSE et al., 2015). A empresa HyVibe indica que o termo se refere a guitarras que apresentam conectividade sem fio, ainda que careçam de um sistema de coleta de dados e recuperação da informação musical (TURCHET, 2019).

De todo modo, esta categoria de instrumento abrange diferentes vertentes da tecnologia, recorrendo a sensores e atuadores, inteligência embarcada capaz de suportar áudio em tempo real e conectividade sem fio para redes locais ou remotas. Eles foram projetados para criarem uma ponte de comunicação entre musicistas, profissionais da área e o público de maneira geral. Para além de permitir a troca de dados e informações em um ambiente musical, conseguem agregar suas funcionalidades a diversos outros equipamentos, como placas de som, controladores e pedais de efeito para guitarra (TURCHET, 2019; TURCHET; MCPHERSON; FISCHIONE, 2016).

Um SMI têm cinco características que merecem destaque: i) gestão de conhecimento, o que indica que este instrumento consegue manter informações sobre si mesmo e sobre o ambiente onde ele está inserido; ii) raciocínio, sendo a capacidade de tomar decisões e atuar de forma independente; iii) aprendizagem, que permite assimilar ações a partir de experiências anteriores;

iv) interação humano-SMI, onde instrumento pode se adaptar às habilidades e exigências do musicista, alterando seu comportamento e serviços oferecidos; e v) conectividade, onde um SMI deve trocar informações pela rede, preferencialmente de forma sem fio e com uma gama diversificada de aparelhos (TURCHET, 2019).

Apesar do potencial de sua infraestrutura tecnológica e de corroborarem para a relação entre computação ubíqua e prática musical, a pesquisa em SMI está em estágio inicial e quase todos os modelos existentes são protótipos, existindo apenas alguns casos de implementação real, como a Sensus Smart Guitar[1] (TURCHET; BENINCASO; FISCHIONE, 2017) e o Smart Cajón (TURCHET; MCPHERSON; BARTHET, 2018).

Instrumento musical digital

Um instrumento musical digital pode ser descrito de diversas formas. Robert Moog, mais conhecido pelo seu pioneirismo na criação de instrumentos eletrônicos, define estes dispositivos a partir de sua composição estrutural, que segundo ele, deve apresentar um gerador de som, uma interface visual e tátil que receba os comandos do músico, e um meio físico que conecta estes dois componentes (MOOG, 1988). Já Pressing define esses instrumentos como toda interface com um campo de controle e interação, embutida de um processador de saída de áudio. Esta descrição coincide com a definição proposta por Miranda e Wanderley (PRESSING, 1990; MIRANDA; WANDERLEY, 2006).

A estrutura básica de um DMI adota uma divisão tripartite, composta por uma entrada ou interface gestual, responsável por captar os comandos inseridos pelo músico; uma unidade de geração sonora, que nada mais é que um sintetizador digital que produz e organiza a saída de sons; e finalmente, uma unidade de mapeamento, capaz de interconectar às duas primeiras partes e atribuir um significado musical às ações realizadas nos instrumentos (PATRICIO, 2010; ROCHA; ARAÚJO; SCHIAVONI, 2019). Esta estrutura é apresentada na Figura 2.

Os instrumentos musicais digitais podem ser categorizados em quatro grupos: i) versões eletrônicas de instrumentos acústicos, ii) instrumentos aumentados, iii) instrumentos inspirados nos modelos tradicionais e iv) instrumentos alternativos. A primeira classe utiliza a mesma estrutura física e os mesmos conceitos sonoros de um instrumento orgânico, com o diferencial de realizar a geração de som por meios eletrônicos. Semelhante a esta classe, a segunda categoria aumenta as capacidades de um instrumento acústico a partir do uso de dispositivos eletroeletrônicos. O terceiro nível, no que lhe diz respeito, consiste na criação de instrumentos musicais digitais inspirados nos modos de construção, funcionamento e execução de instrumentos tradicionais. A quarta categoria trata do uso de dispositivos não-musicais para criação de IMD, com destaque para *joysticks*, mesas digitalizadoras, *tablets*, e é claro, teclado QWERTY e mouse (JORDÀ, 2005; PATRICIO, 2010).

[1] <https://elk.audio/sensus-smart-guitar/>

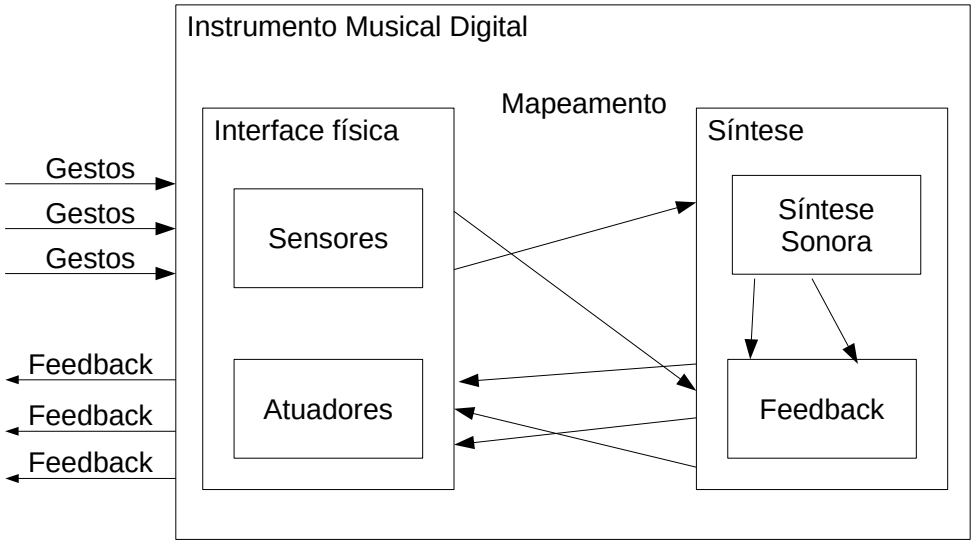


Figura 2 – Estrutura clássica de um IMD.
Fonte: O autor
Adaptado de: (CALEGARIO, 2019)

2.2.2 Campos relacionados

Esta Seção revisa alguns dos vários domínios de aplicação que auxiliam um ambiente de IoMusT. A análise não pretende ser exaustiva, visando descrever as principais características e funcionalidades de cada área.

Performance musical em rede

As tentativas de performances em rede datam dos primórdios deste tipo de comunicação. Ainda na década de 1970, um grupo de musicistas e compositores conectaram três computadores para a troca de informação musical (WEINBERG, 2002). Com a expansão da Internet, houve uma melhora significativa nesta prática. Uma performance por rede é, portanto, uma interação em tempo real via máquina, que permite que artistas em diferentes localidades interajam entre si como se eles estivessem no mesmo ambiente. Embora não pretenda substituir o modelo tradicional, ela traz um efeito catalítico na criação musical e nas interações sociais, promovendo a criatividade e o intercâmbio de culturas (LAZZARO; WAWRZYNEK, 2001).

Das principais características que este modelo deve apresentar, estão: baixa latência, onde os sons produzidos devem ser ouvidos quase instantaneamente; sincronização, para evitar que longos atrasos prejudiquem os músicos; interoperabilidade por meio de padronização, que permite que aparelhos distintos possam se comunicar pela rede; fácil integração e participação, aspectos que garantem que os usuários não tenham dificuldade para encontrar dispositivos na rede, bem como possam se conectar ou desconectar quando quiserem; e por fim, escalabilidade, que torna o sistema abrangente e permite a participação distribuída entre os usuários (ALEXANDRAKI et al., 2008). Um exemplo deste ambiente pode ser observado na Figura 3.

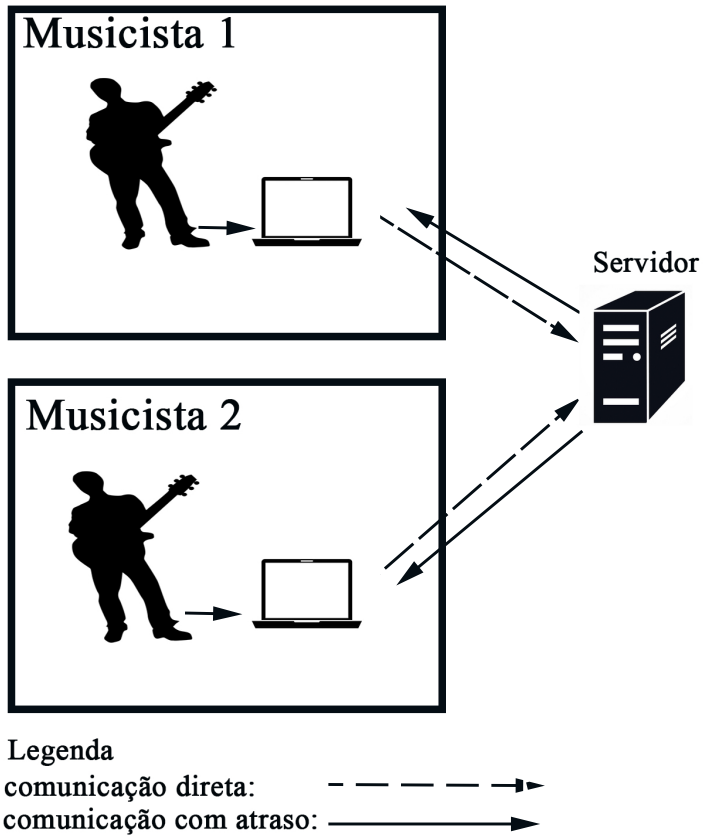


Figura 3 – Sistema básico de uma performance musical em rede.
Fonte: O autor
Adaptado de: (SMUS, 2013)

Quanto aos desafios a serem superados, pode-se citar a alta largura de banda exigida, sensibilidade ao atraso na entrega de pacotes de dados e exigência de ordenação no fluxo transmitido (GU et al., 2004).

Arte interativa

O conceito de interação é oriundo da física, onde é usado para se referir a partículas que tem o seu movimento alterado por um grupo de outras partículas. Posteriormente, o conceito se expandiu para a psicologia e sociologia, com a premissa de que nenhuma ação humana ou social existe sem interação. Somente na década de 1970 que o termo foi incorporado na tecnologia, transmutado para interatividade e representando a transição da máquina computacional rígida para uma máquina conversacional (FORNY, 2006).

A arte, por sua vez, sempre teve sua interação marcada pela relação do artista (emissor) com o meio que ele dispõe para materializar a obra (transmissor), enquanto ao público (receptor) cabia apenas o papel de observar tudo de forma passiva. Isso começou a mudar quando movimentos artísticos liderados por Allan Kaprow e pelos grupos Fluxos e Gutai começaram a permitir uma participação mais ativa da audiência, sendo muitas vezes ela a responsável pela execução da obra de arte em si (NARDIM, 2009; VIEIRA et al., 2020). Nesse contexto, surge

a arte interativa, caracterizada por conferir ao espectador um grau de envolvimento ativo no espetáculo, seja caminhando entre as instalações e esculturas, ou produzindo sons, imagens e movimentos (FORNY, 2006).

Essa interação pode ocorrer de duas formas: imediata ou reflexiva. A primeira delas destaca as qualidades emergentes da participação do público, permitindo que ele altere a obra de maneira mais significativa. Já a segunda configuração foca mais nas reações da audiência, caracterizando a participação a partir de suas experiências pessoais. Elas estão intrinsecamente relacionadas, visto que uma experiência estética combina níveis individuais e coletivos de interação (PARGMAN; ROSSITTO; BARKHUUS, 2014).

A arquitetura desses ambientes é concebida para lidar com diferentes tipos de sinais, que vão desde áudio e vídeo até aqueles produzidos pelo corpo humano, como as batidas do coração. Sendo assim, elas também requerem funcionalidades que garantam interoperabilidade e lidem com o atraso na entrega de dados.

Música ubíqua

A música ubíqua (KELLER; LAZZARINI; PIMENTA, 2014), geralmente abreviada para ubimus, é um campo de pesquisa que combina música, tecnologia e processos criativos com uma forte base social e comunitária. O termo foi inspirado na computação ubíqua e parte da ideia que a música é onipresente na sociedade moderna, mesmo que de forma passiva. Embora a proposta original da ubimus seja voltada para a produção musical e essa ideia ainda permaneça como peça central nos debates, as atuais novidades tecnológicas têm aberto novas dimensões sociais e cognitivas para esta área, levando-a a se interessar cada vez mais por temas educacionais e artísticos. Assim, as perspectivas atuais abrangem uma ampla diversidade de assuntos e atores, que vão desde participantes casuais até músicos altamente treinados (KELLER; SCHIAVONI; LAZZARINI, 2019).

O ecossistema ubimus suporta a integração de ferramentas de áudios e a interação com o público, podendo ser reconfigurável para atender as necessidades dos usuários. Consequentemente, os conceitos almejados não dependem de implementações específicas. Outras características importantes são as abordagens conceituais e a confiança em métodos empíricos. Essas visões estimulam o desenvolvimento de tecnologias para a criação de música, utilizando principalmente objetos e espaços comuns ao cotidiano.

2.3 Sistemas distribuídos

A área da computação passou por grandes mudanças a partir da década de 1980, especialmente pelo surgimento dos microprocessadores e das redes de transmissão em alta velocidade. Esse cenário permitiu a chegada de sistemas abundantes em números de computadores conectados à rede, que se comunicam de forma bidirecional e coordenam ações apenas trocando

mensagens. Eles receberam o nome de sistemas distribuídos (COULOURIS et al., 2013). Várias definições já foram oferecidas para este conceito, sendo a de Tanembaum uma das mais proeminentes, definindo-o como:

> *"Um sistema distribuído é um conjunto de computadores independentes que se apresenta a seus usuários como um sistema único e coerente"* (TANENBAUM; STEEN, 2007).

Este enunciado indica alguns aspectos importantes a serem observados, como sua estrutura formada por diversos computadores autônomos e seu caráter colaborativo (sendo este o aspecto mais importante a ser tratado quando se fala em sistemas distribuídos). Pode-se citar ainda a facilidade de expansão e a disponibilidade contínua, onde usuários e aplicações não devem perceber quais partes foram substituídas ou consertadas, ou ainda quais novos componentes foram adicionados (CORRÊA; CERQUEIRA, 2009).

O objetivo principal de um sistema distribuído é facilitar a usuários e aplicações o acesso a recursos remotos e seu compartilhamento de maneira controlada e eficiente. Para tal, três metas importantes devem ser cumpridas, a saber: ocultar razoavelmente bem o fato de que os recursos são distribuídos na rede, ser expansível e ser aberto (oferecer serviços de acordo com regras padronizadas que descrevem sua sintaxe e semântica) (TANENBAUM; STEEN, 2007).

Suas principais características são a interoperabilidade, que indica até que ponto sistemas diferentes devem coexistir e trabalhar juntos; portabilidade, que diz até onde uma aplicação desenvolvida para um sistema A pode ser executada, sem modificação, em um sistema B; e escalabilidade (ou extensibilidade), onde deve ser fácil a configuração e adesão de componentes diferentes (TANENBAUM; STEEN, 2007; FUENTES, F, 2015).

A motivação para criar um sistema nestes moldes é o compartilhamento de recursos. Esse termo é bastante abstrato e depende de seu contexto de uso, mas aqui, ele se refere ao conjunto de coisas que podem ser compartilhadas em um sistema de computadores interligados, abrangendo desde *hardwares* até arquivos e dados de todos os tipos possíveis (SANTANA, 2019).

A separação espacial dos computadores que formam esses sistemas variam desde poucos metros, como em uma rede local, até quilômetros de distância, em uma rede dispersa. Da mesma maneira, variam bastante os desafios enfrentados pelos projetistas, sendo os principais: segurança, escalabilidade, tratamento de falhas, concorrência e transparência (TANENBAUM; STEEN, 2007; SANTANA, 2019).

A segurança é um aspecto primordial na comunicação via redes de computadores, com destaque para a criptografia que protege os recursos compartilhados e mantém as informações sob sigilo. Destaca-se também a necessidade de prevenção aos ataques de negação de serviço.

Um sistema distribuído é dito escalável se o custo de adição de um usuário for constante em termos de recursos que devem ser adicionados. Deve-se evitar gargalos de desempenho,

estruturar dados hierarquicamente para melhores tempos de acessos e replicá-los quando eles são frequentemente acessados.

Como todo elemento computacional está sujeito a complicações, os componentes de um sistema distribuído devem ser projetados para tratar essas falhas apropriadamente, no conceito chamado tolerância a falhas.

A concorrência, no que lhe concerne, diz respeito à tentativa de múltiplos usuários tentarem acessar, de forma simultânea, o mesmo recurso. Sendo assim, deve-se projetar o ambiente de tal modo que seja garantida a consistência nos dados que circulam por ele.

O último desafio aqui pontuado é a transparência, cujo objetivo é tornar certos aspectos "invisíveis" para o desenvolvedor, de modo a se preocupar somente com seu projeto. Por exemplo, ele não precisa de detalhes sobre como suas operações serão acessadas por outros componentes.

A partir dessa breve apresentação, percebe-se que um sistema distribuído pode ser difícil de implementar e de manter, mas ele também apresenta muitos benefícios, como: escalonamento, que permite contabilizar mais tráfego; crescimento modular, onde quase não existem limites para o tanto que um sistema pode crescer; tolerância a falhas, com maior ocorrência em um sistema do que em uma única máquina; viabilidade econômica, graças ao seu caráter escalável; baixa latência, devido à dispersão dos nós que compõem o ambiente; e paralelismo, principalmente pela sua característica de dividir processamentos complexos em tarefas menores (GABRIEL et al., 1998).

Dentre estas vantagens, destaca-se a escalabilidade. Ela pode ser classificada em horizontal, onde mais servidores são adicionados ao "*pool*" de recursos, e vertical, onde poder de *hardware*, como processamento, memória RAM, armazenamento, fonte de energia e outros recursos são adicionados ao ambiente (SANTANA, 2019).

Suas aplicações são as mais diversas possíveis, sendo a Internet o maior exemplo. Está presente também em qualquer aplicação de intranet e *mobile*, e também em serviços baseados na computação em nuvem. De forma menos geral, compõe os motores de busca, sistemas financeiros, jogos *online* e redes sociais (TANENBAUM; STEEN, 2007).

2.3.1 Modelo Cliente-Servidor

Dentre os vários modelos de arquiteturas que se encaixam nos sistemas distribuídos, o mais famoso e proeminente é o cliente-servidor. Nele, os requerentes dos serviços são chamados de clientes, enquanto as tarefas e cargas de trabalho são de responsabilidade dos fornecedores de recursos, os servidores.

O clientes são marcados por fazerem solicitações aos servidores, consumirem recursos de rede e jamais se conectarem a outro cliente, ao passo que os servidores são reconhecidos por sempre esperarem uma requisição, pela capacidade de conexão a outros servidores, interação

com usuários finais e estruturação de um sistema (KUROSE; ROSS, 2013).

Dentre suas principais características estão a comunicação fácil e rápida, a divisão do processamento entre várias máquinas, o fácil compartilhamento de recursos e a redução da replicação de dados, visto que eles são armazenados nos servidores e não nos clientes. A maneira mais simples de implementar este modelo é através de um protocolo do tipo requisição/resposta (OLIVEIRA; FRAGA; MONTEZ, 2002).

Sua arquitetura é multicamadas, o que facilita o processo de programação distribuída. Neste formato, destaca-se o modelo de duas camadas (*2-tier*), mais simples, e o modelo de três camadas (*3-tier*), o mais utilizando atualmente. O primeiro padrão, também conhecido como *thick client*, envolve somente um computador (cliente) e um banco de dados (servidor) que se conectam pela rede. O cliente executa a codificação e a lógica de negócios e exibe a saída para o usuário. Seu uso é recomendado quando o cliente tem acesso direto ao banco de dados (OLUWATOSIN, 2014).

O segundo padrão arquitetural é composto pela camada de usuário, que abriga os clientes, responsáveis pela entrada de dados, manipulação e análise; camada de negócios, que provê os serviços e regras do sistema; e camada de dados, formada pelo servidor que cuida da organização, segurança, consistência e integridade das informações. Neste modelo, o cliente exige menos recursos e menos codificação, enquanto o servidor provê mais recursos. Envolve ainda um *middleware*, que nada mais é que um *software* executado em máquina separada, responsável por aplicar uma lógica que não está presente originalmente no sistema operacional. Este formato pode ainda ser expansível para n-camadas, a depender das necessidades do projetista (KAMBALYAL, 2010; RABELO, 1998).

Apesar de robusto e amplamente utilizado, o modelo cliente-servidor não está isento de falhas, como perdas no desempenho diante de uma grande número de pessoas conectadas. Existem ainda alguns desafios na sua implementação, como exigência de experiência qualificada, alto custo dos servidores e questões de segurança (DAVIS, 1996).

Ainda assim, suas vantagens são claras, como fácil manutenção, já que é possível substituir, reparar, atualizar e realocar um servidor sem que o sistema seja afetado pela mudança; dados armazenados nos servidores, o que garante segurança e melhor acesso a recursos; fácil atualização e administração dos dados; constantes atualizações e correções de falhas; uso simples e suporte para vários clientes com diferentes capacidades operacionais.

2.4 Pipes-and-Filters

A arquitetura de *software* é responsável por indicar como um conjunto de componentes computacionais se relaciona. Ela define os elementos estruturais que formam essa arquitetura e também incorpora informações sobre eles. Sendo assim, é o primeiro lugar onde a abstração do

sistema é conceituada (MAIER; EMERY; HILLIARD, 2001).

Para a solução dos problemas que surgem, é comum combinar vários modelos arquiteturais, assim como aplicar padrões que agilizam a solução. Um desses padrões é o modelo Pipes-and-Filters (BUSCHMANN et al., 1996; THALER, 2020; FRANCOIS, 2003). Ele é formado por elementos chamados filtros, que recebem um fluxo de dados em suas entradas, aplicam processamento a eles, conforme a lógica de um ou vários algoritmos, e os enviam para a saída utilizando canais de comunicação, chamados *pipes* (dutos) (BIJLSMA et al., 2011; AVGERIOU; ZDUN, 2005).

Os filtros são componentes independentes, ou seja, não se relacionam diretamente com seus pares, restringindo suas funcionalidades somente ao que é recebido nos dutos de entrada e no que aparece nos dutos de saída. Mas isso não impede que eles sejam alocados em diversas posições, de modo a obter um resultado de saída diferente. Para além disso, é possível adicionar ou omitir filtros e conectá-los a diversos dutos. Esta configuração proporciona novas sequências de processamento sem alterar as propriedades dos filtros.

No que diz respeito à geração e consumo de dados, podem ser classificados em **geradores (*source*)**, quando tem a responsabilidade de ser o filtro que cria o dado inicial; **consumidores (*sink*)**, que apenas consomem os dados recebidos após todo o processamento; e **híbridos**, quando desempenham ambas as funções.

Outras importantes características funcionais dos filtros são apresentadas, de forma breve, a seguir (THALER, 2020):

- Não devem fazer suposições sobre a identidade dos filtros vizinhos;
- Devem possuir escalonamento justo (a saída da rede não deve depender da ordem de execução de filtros individuais);
- Todos eles possuem portas de entrada e saída;
- A saída, geralmente, começa a ser consumida antes que a entrada tenha terminado de receber dados;
- Capacidade de enriquecer os dados de entrada (ex: adicionar detalhes e informações);
- Refinamento de dados (ex: filtra os dados desinteressantes ou redundantes);
- Transformação de dados de entrada (ex: transforma certas unidades de medidas em outras);
- Análise de dados (estatísticas, identificação, classificação e etc).

Os dutos são responsáveis somente por transferir dados entre os filtros. Eventualmente, pode ser necessário que armazenem e sincronizem a atividade destes dados, mas nunca aplicam processamento e nem alteram os valores das informações que trafegam por eles.

Destaca-se o fato deles serem objetos de primeira classe, o que indica que podem ser representados e manipulados de forma independente; a capacidade de implementar *buffer* limitado ou ilimitado; a restrição do fluxo de dados; a contenção de referências simultâneas a objetos compartilhados e o envio de dois processos distintos para um único *host*.

A Figura 4 ilustra esta arquitetura. Note que um filtro pode ter mais de um canal de comunicação, bem como o fluxo não precisa ser linear.

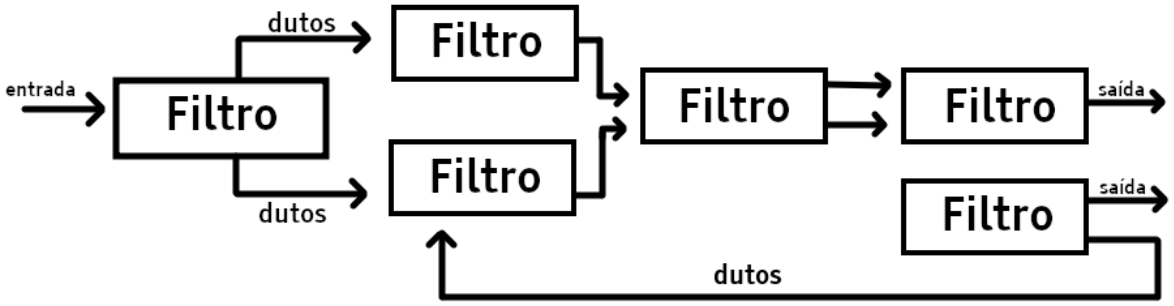


Figura 4 – Arquitetura básica do modelo Pipes-and-Filters.
Fonte: O autor
Adaptado de: (SHAW; GARLAN, 1996)

Este padrão é recomendado para dividir uma tarefa complexa em várias outras menores e mais fáceis de serem resolvidas, ou ainda para permitir a flexibilidade e reordenação das etapas de processamento, obter uma solução confiável que minimiza possíveis falhas, e quando as etapas de um projeto tem escalabilidade diferente.

Após toda essa elucidação, ficam claras suas vantagens, que abrangem desde uma fácil prototipação e potencial para paralelismo, até facilidade de entendimento, filtros independentes que podem ser tratados como caixas pretas (o que garante atributos de qualidade, como ocultação de informações, baixo acoplamento, capacidade de modificação e reutilização), e a construção adaptável a um processo.

2.5 Meios de Transmissão pelas Redes

Uma rede de transmissão é formada por um conjunto de sistemas que permitem a interconexão entre computadores e os mais diversos aparelhos eletrônicos. Para o envio de informações, são usados os meios físicos ou não-físicos. A presente Seção se encarrega de apresentar dois mecanismos de transmissão que se encaixam na primeira categoria e foram utilizados ao longo deste trabalho, que é o cabo de par trançado e o Wi-Fi. Ainda é apresentada a latência e o *jitter*, conceitos fundamentais que aparecem na troca de dados via rede.

Cabo de par trançado

Para a conexão direta entre dispositivos sem a necessidade de intervenção de outros aparelhos, como *hub*, *switch* ou roteador, são utilizados os cabos de par trançado. Eles possuem dois ou mais fios entrelaçados de forma espiral, onde cada par é fisicamente isolado e coberto por uma proteção de PVC. São constituídos por um condutor positivo, normalmente nas cores laranja, verde, azul ou marrom, e um negativo, usualmente de cor branca. O objetivo dessa composição é criar uma espiral virtual com capacitância e indutância suficientes para anular a indução de ruídos e a diafonia, enquanto consegue manter as propriedades elétricas constantes e em todo o comprimento do cabo (BELETTINI, 2015; TAROUCO, 1986; BOQUE; HAHN; ASTIAZARA, 2012).

É classificado em dois tipos diferentes, o UTP, não blindado e utilizado em redes onde não existe grande necessidade de isolamento, e STP, blindado e comum em ambientes com alta interferência eletromagnética. Um ponto importante deste último modelo é sua forma de transmissão utilizando pares balanceados, onde dentro de cada par, os fios enviam o mesmo sinal, mas com polaridade invertida (BELETTINI, 2015).

As principais vantagens para se usar um cabo de par trançado são decorrentes do baixo custo, alta flexibilidade, prevenção a ruídos elétricos e fácil instalação. Entretanto, como qualquer condutor convencional, ele sofre influências do meio externo e perde energia com o aumento da distância, além de somente permitir a conexão de dois pontos da rede. Mas ainda assim é uma excelente alternativa para conexões de curto e médio alcance (BOQUE; HAHN; ASTIAZARA, 2012).

Wi-Fi

A comunicação sem fio tem se tornado cada vez mais popular nos últimos anos, muito por conta da ampla gama de dispositivos que comportam este tipo de conexão, pela extensa cobertura, evolução contínua e uso de protocolos globais e bem definidos. As redes Wi-Fi[2] são o modelo dominante neste tipo de acesso. Elas recorrem a ondas de rádio bidirecionais para transmitir informações por meio de roteadores, *access points*, adaptadores sem fio, repetidores e sistemas embarcados. Para que um dispositivo tenha acesso a estes sinais, ele deve estar em determinado raio de ação, chamado *hotspot*, que atua a partir de duas frequências específicas, 2,4 GHz ou 5 GHz (JONES; LIU, 2007; SUN et al., 2014).

O Wi-Fi é descrito no padrão IEEE 802.11, que contempla uma diversificada família de protocolos, listados a seguir (RUFINO, 2014):

- IEEE 802.11a: apresenta como principal característica um significativo aumento de velocidade, alcançando até 54 Mbps, na faixa de 5 GHz. Permite ainda a conexão de 64 clientes

[2] <https://www.wi-fi.org/certification>

de forma simultânea e conta com 12 canais não sobrepostos, cobrindo uma área maior e mais densamente povoada;

- IEEE 802.11b: padrão para a frequência de 2,4 GHz, utiliza a técnica DSSS para diminuição de interferência, alcançando a capacidade teórica de até 11 Mbps;
- IEEE 802.11g: mais recente que os demais, também atua a 2,4 GHz de frequência, mas como permite a coexistência com equipamentos que seguem o padrão IEEE 802.11b, isso não é um problema. Incorpora várias características do modelo IEEE 802.11a, como a velocidade de cerca de 54 Mbps;
- IEEE 802.11i: mais focado em oferecer mecanismo de autenticação e privacidade na rede;
- IEEE 802.11n: tem foco principal no aumento da velocidade e da área de cobertura geográfica. Funciona em ambas as frequências disponíveis e permite compatibilidade retroativa com outros padrões;
- IEEE 802.11ac: apresenta velocidades que chegam até a 1,3 Gbps, utiliza frequência de 5 GHz com compatibilidade para 2,4 GHz, tem melhor qualidade de sinal e conexões mais estáveis;
- IEEE 802.1x: mesmo não sendo projetado para redes sem fio, tem características complementares a essas redes, já que permite autenticação de forma escalável e expansível, dois métodos já conhecidos. Dessa maneira, é possível promover um padrão único de autenticação, independentemente da tecnologia empregada.

Por conta de sua popularidade, o termo Wi-Fi é utilizado como sinônimo para qualquer WLAN e até mesmo para o protocolo IEEE 802.11, apesar de algumas propriedades dele não terem sido implementadas nesta rede sem fio. Quanto às suas vantagens, cita-se os preços acessíveis, a capacidade de ser utilizada em espaços onde o cabeamento é inviável, como áreas ao ar livre e edifícios históricos, a compatibilidade entre diferentes produtos e protocolos, maior segurança, qualidade de serviço, economia de energia e boa aplicabilidade em sistemas de áudio e vídeo (CAMPS-MUR; GARCIA-SAAVEDRA; SERRANO, 2013; ESCOBAR, 2015).

Latência e *Jitter*

A comunicação em rede sempre possuiu um atraso entre o envio de um pacote e o recebimento do mesmo, chamado comumente de latência. Esta propriedade consiste no tempo que leva para uma mensagem inteira enviada pelo *host* de origem alcançar o receptor e ter a confirmação de chegada. Ela é formada por quatro componentes: tempo de propagação (que mede o tempo necessário para um *bit* trafegar da origem até o destino), tempo de transmissão (tempo entre a saída do primeiro *bit* no emissor até a chegada do último no receptor), tempo de fila (tempo que cada dispositivo intermediário ou terminal retém a mensagem antes dela ser processada)

e retardo de processamento (tempo que um pacote leva para atravessar a rede até o destino, passando por roteadores e enlaces intermediários). Deve-se somar ainda o congestionamento de rede e os *firewalls* utilizados no processo. Na prática musical, acrescenta-se também o atraso do som e os diversos *buffers* envolvidos no processo, como o que existe na placa de captura de áudio (FOROUZAN, 2007).

Por natureza, ela pode gerar erros de base de tempo, perda de sequência dos dados e até perda parcial do conteúdo. Uma das principais fontes que contribuem para a ocorrência deste fenômeno, por questões óbvias, é a distância entre os dois pontos que querem se comunicar. Vale também o destaque para a quantidade de *links* que intermedeiam esta conexão, assim como suas taxas de transferência e erro (BARBOSA; CARDOSO, 2021).

Esse atraso na entrega de pacotes de dados pode ser variável por conta de inúmeros fatores, como a inserção de um novo nó no sistema ou o aumento no tráfego da rede. A esta variação na latência da-se o nome de *jitter*. Ele pode causar uma recepção não regular dos dados e representa um problema especialmente para aplicações que utilizam informações sensíveis ao tempo, como áudio e vídeo, por exemplo. Uma representação gráfica desta propriedade é observada na Figura 5 (FOROUZAN, 2007).

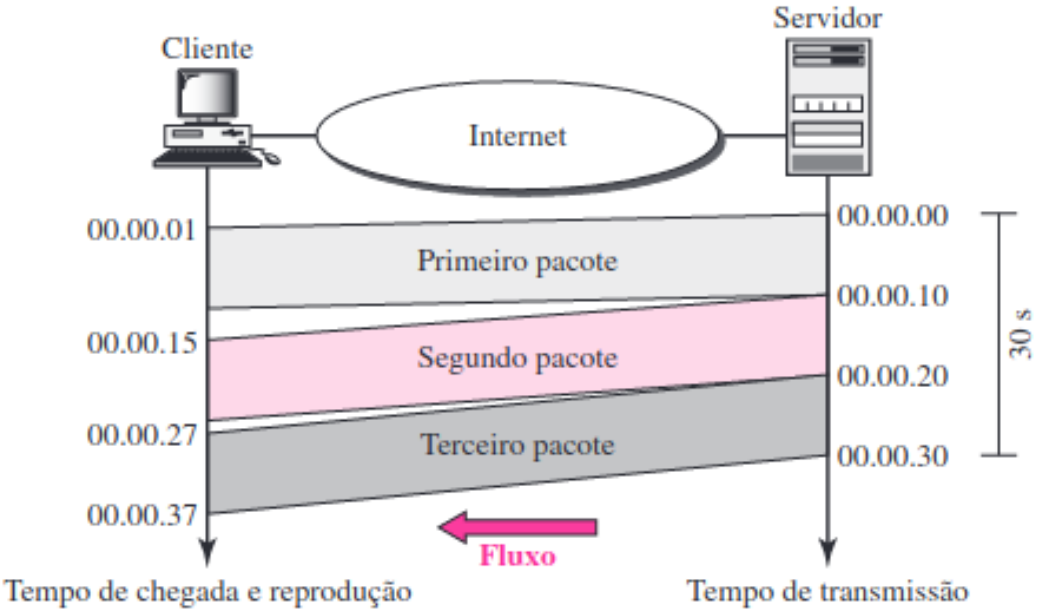


Figura 5 – Atraso entre pacotes que compõem o mesmo bloco de informação, caracterizando a ocorrência do *jitter*.

Fonte: (FOROUZAN, 2007)

O *jitter* é normalmente dividido em duas classes: aleatório e regular. O aleatório é uma consequência do ruído térmico, causado pelo fluxo de elétrons nos condutores; ruído de disparo, decorrente de buracos nos cabos causados pela corrente de polarização e por frequências de banda muito altas; e ruído de cintilação, decorrente de dados onde o espectro é inversamente

proporcional à frequência. Já o *jitter* regular é causado pela interferência sofrida nos equipamentos usados para comunicação. Além disso, ambos podem ser ocasionados por conta do meio físico escolhido para conexão à internet (indo desde par-trançado e cabo coaxial até fibra óptica e espectro de rádio terrestre), pela largura de banda utilizada e até mesmo pelo número de pessoas conectadas à rede (MOZHAIEV; KUCHUK; USATENKO, 2019).

3 TRABALHOS RELACIONADOS

Nesta Seção, encontra-se uma breve descrição dos modelos que vão servir de base de comparação ao longo deste trabalho, ressaltando seus protocolos, formatos de arquivos utilizados e algumas particularidades inerentes a cada um deles. O objetivo desta análise é identificar pontos em comum nos exemplares e colher informações que podem ser úteis na criação de cenários de IoMusT.

Primeiro modelo

A primeira ferramenta selecionada para análise propõe uma arquitetura de IoMusT enriquecida semanticamente (TURCHET et al., 2018). Ela ainda utiliza o modelo cliente-servidor para processamento de eventos e a linguagem SPARQL para consultas em uma base de dados, com o propósito de garantir que musicistas e membros da audiência cooperem entre si em uma rede sem fio local.

Os autores defendem que essa estruturação deva permitir uma interação oportuna e fracamente acoplada entre as entidades. Para alcançar tal objetivo, eles utilizaram um *middleware* orientado para a publicação e assinatura de recursos, responsável por habilitar a interoperabilidade e compartilhar informações da base de conhecimento.

As coisas musicais utilizadas neste modelo são protótipos feitos na placa de processamento Bela[1], que proporciona áudio com baixa latência. Esses objetos foram classificados de acordo com seu modo de funcionamento, recebendo três distinções: produtores, com função de atualizar o conteúdo da base de conhecimento, adicionando, removendo ou modificando informações; consumidores, que somente leem a base de dados e operam sob a lógica de solicitação-resposta; e agregadores, que apesar de não serem obrigatórios, podem aparecer atuando tanto como consumidores quanto produtores, pois reagem às mudanças no ambiente e também atualizam o banco de dados com novos conhecimentos.

O motor de áudio foi codificado na biblioteca libpd[2], que permite que *patches* de áudio feitos no Pure Data[3] possam ser executados nas placas Bela. Além disso, os dados eram transmitidos ao servidor por meio de um *script* em Python e a informação musical foi trocada no protocolo OSC. A rede sem fio foi implementada no protocolo IEEE 802.11ac em um roteador configurado no modo *access point* para redução da latência, além de ter suas configurações de segurança desabilitadas.

A combinação única e inovadora de tecnologias propostas neste ambiente propõe trans-

[1] <https://bela.io/>

[2] <https://github.com/libpd>

[3] Também conhecido como Pd, é uma linguagem de programação e um ambiente gráfico para desenvolvimento e processamento de áudio em tempo real, desenvolvidos por Miller Puckette na década de 1990, voltados para criação e composição de música eletrônica, performances ao vivo, produção de efeitos sonoros, análise de áudio, controle de câmeras e sensores, interação com a web e trabalhos multimédias em geral (PUCKETTE, 1996).

formações em experiências artísticas, uma vez que atribui inteligência a elas, aumentando as capacidades musicais e colocando o público em um contato mais próximo com os artistas. Outras vantagens são os usos da web semântica, a separação entre parte lógica e os detalhes de implementação, os eventos e ações reutilizáveis, e claro, a interoperabilidade como um todo. A arquitetura foi validada por meio da implementação observada na Figura 6. Pode-se observar, por fim, seu potencial de integrar aplicações reais em concertos locais ou remotos.

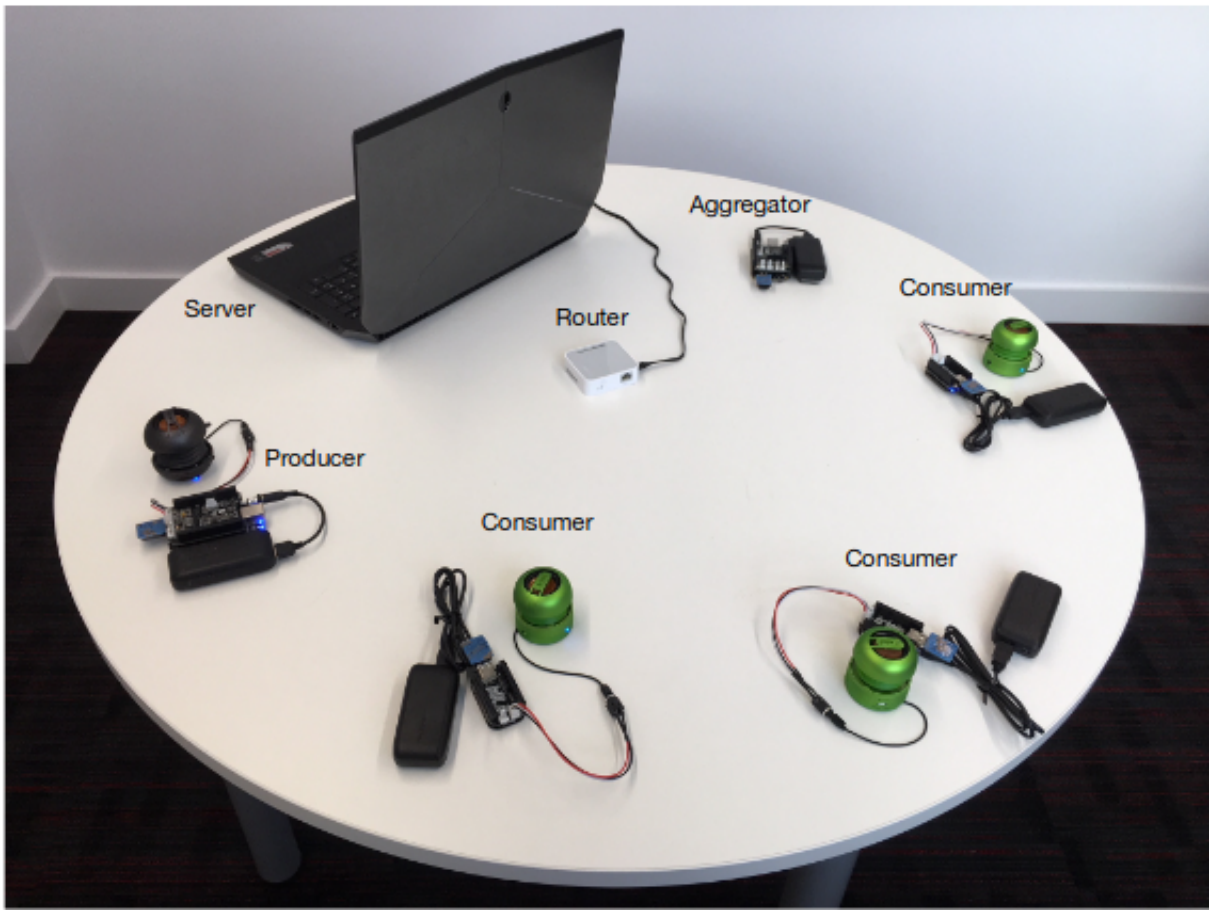


Figura 6 – Primeiro ambiente de IoMusT analisado.
Fonte: (TURCHET et al., 2018)

Segundo modelo

O segundo modelo apresenta uma sessão musical que utiliza conceitos de IoMusT para permitir a interação entre um musicista que utiliza um SMI, o Smart Mandolin (TURCHET, 2018), membros da audiência, que participam utilizando *smartphones*, e o repositório gratuito de áudio Freesound[4]. Foi idealizado para possibilitar novas formas de interação entre o intérprete e o instrumento, bem como o intérprete e o público (TURCHET; BARTHET, 2018).

As duas formas de interação passam por recuperar conteúdo no Freesound para acompanhamento musical. No caso do artista, ele pesquisa pelo conteúdo desejado em seu próprio

[4] <https://freesound.org/>

instrumento, enquanto que para o público, o conteúdo é gerado a partir de uma ação colaborativa.

O mecanismo de áudio foi novamente feito no Pure Data, dada a sua variedade de efeitos sonoros e estratégias de mapeamento para controlar os sons coletados pelos sensores e microfones presentes no SMI. O público, no que lhe diz respeito, conseguiu participar utilizando um aplicativo dedicado que abrange os dois casos de uso. Este aplicativo foi criado no TouchOSC[5] e é responsável por enviar mensagens OSC encapsuladas em pacotes UDP[6] para o bandolim e por realizar consultas no repositório.

As amostras baixadas no Freesound eram no formato MP3, com intuito de diminuir o tamanho do arquivo e economizar recursos de rede. Como o Pure Data lida apenas com dados no formato WAV, uma conversão foi necessária. Já a conectividade ocorreu de forma sem fio, através de um Wi-Fi IEEE 802.11ac.

O ponto forte do sistema é a grande variedade de sons que podem ser reproduzidos, o que favorece a criação de uma narrativa e auxilia na expressividade musical. Outro ponto de destaque é o pioneirismo ao combinar a IoMusT com a iniciativa Audio Commons[7] para criar novas formas de interação musical.

Terceiro modelo

Os rigorosos requisitos de confiabilidade e baixa latência exigidos em um ambiente de IoMusT estão fora do alcance da atual geração de redes de móveis. Em vista disso, o terceiro modelo propõe uma arquitetura de comunicação para prática musical baseada na tecnologia 5G (CENTENARO; CASARI; TURCHET, 2020). A vantagem de pensar o sistema dessa forma é que ele fornece entrega de pacotes a uma baixa latência e de forma altamente confiável. Este sistema combina uma rede de acesso de rádio e uma rede 5G, que transfere áudio e permite a mediação de serviços.

As coisas musicais propostas incluem *hardware* para entrada, saída e processamento de áudio, contando com o sistema operacional Elk Audio[8], e um módulo de comunicação 5G. Elas podem estar presentes em cenários de ensaio remoto, onde os musicistas não precisam estar no mesmo local físico, e em interações um-para-um entre os artistas e o servidor.

De todos os ambientes apresentados, este é o que está em estágio mais inicial. Por conta disso, os autores não se preocuparam com questões sobre formato do áudio, protocolos de informação musical e de comunicação em telefonia móvel. Mas alguns desafios e dilemas já foram apresentados, como a necessidade de uma rede 5G, que ainda carece de regulamentação e difusão em diversos países, falta de caracterização da cobertura do serviço, ausência de

[5] <https://hexler.net/touchosc>

[6] Definido pelo RFC 768, atua no envio de pacotes de um hospedeiro para outro. Entretanto, a entrega não é garantida, visto que eles podem se perder no caminho, chegar fora de ordem e não exigirem apresentação prévia entre as entidades que vão se comunicar, sendo caracterizado como não orientado para conexão.

[7] <https://www.audiocommons.org/>

[8] <https://elk.audio/>

percepção e inteligência da rede, e a pouca disponibilidade de coisas musicais habilitadas para esta tecnologia.

4 DESIGN DE UM AMBIENTE DE INTERNET DAS COISAS MUSICAIS

O *design* de um sistema consiste em apresentar sua arquitetura, módulos, interfaces, protocolos, dados, processo de concepção e tomadas de decisão que auxiliam na execução de determinada tarefa. Este Capítulo, portanto, é dedicado a apresentar estas informações a cerca do Sunflower, concebido para propor um novo modo de se pensar ambientes de IoMusT e indicar possíveis soluções para os problemas mais recorrentes a esta área. Ele consegue suportar inúmeros instrumentos e coisas musicais que trocam informação pela rede, proporcionando uma experiência musical passível de testes e de inferências de resultados.

4.1 Características desejáveis

Antes de apresentar as características desejáveis para este ambiente e seus dispositivos, cabe uma breve contextualização de como tais parâmetros foram definidos. A investigação nasce como uma continuação dos trabalhos anteriormente realizados pelo orientador desta pesquisa, como a Medusa (SCHIAVONI; QUEIROZ; IAZZETTA, 2011), um ambiente sonoro distribuído que permite que várias máquinas conectadas em uma rede local compartilhem vários fluxos de áudio e MIDI[1] e substituam *mixers* de *hardware* e também cabos de áudio multicanais especializados por comunicação de rede, e o Libmapper (MALLOCH; SINCLAIR; WANDERLEY, 2013), uma biblioteca de *software* de multiplataforma de código aberto, usada para declarar sinais de dados em uma rede compartilhada e permitir que conexões arbitrárias sejam feitas entre eles. A biblioteca cria um sistema de mapeamento distribuído, sem pontos centrais de falha, com potencial para estreita colaboração e fácil paralelização de síntese de mídia. O foco principal do seu desenvolvimento é fornecer ferramentas para criar e usar ferramentas para controle interativo de dados sonoros. Em comum, estes ambientes possuem uma tentativa de padronizar comunicações de cunho musical que ocorrem pela rede.

Somado a isso, existe o interesse do próprio autor em trabalhar com redes de computadores, Internet das Coisas e performances musicais pela rede, o que o levou a integrar o laboratório Arts Lab in Interfaces, Computers, and Everything Else (ALICE), formado por cientistas da computação e artistas, que contribuíram com ideias e sugestões para o ambiente aqui idealizado. Ainda, houve a participação do mesmo na peça “O Chaos das 5” (SCHIAVONI et al., 2019; ARAÚJO et al., 2019), um espetáculo audiovisual desenvolvido pelo Grupo Transdisciplinar de Pesquisa (GTRANS) da Universidade Federal de São João, focado em proporcionar uma experiência interativa para a audiência, valendo-se de uma combinação entre música, elementos visuais e gestuais. Os *insights* tirados desta apresentação serviram para clarificar alguns conceitos

[1] Protocolo criado em 1983 por um consórcio de fabricantes de instrumentos musicais eletrônicos para padronizar a comunicação entre eles, em vista da expansão deste mercado naquele período.

sobre a rede aplicada a um ambiente volátil e também para direcionar as tratativas sobre o comportamento dos espectadores.

Por fim, a experiência adquirida na criação do controlador MIDI Fliperama (VIEIRA; SCHIAVONI, 2020), um instrumento construído a baixo custo e de maneira sustentável, voltado para leigos na música e/ou tecnologia, também contribuíram para estabelecer algumas diretrizes a serem seguidas no desenvolvimento do Sunflower, em especial nas suas coisas musicais, que devem ser heterogêneas, transparentes e de fácil conexão.

4.1.1 Características desejáveis para o ambiente

Posto isso, a análise se volta para os modelos de ambientes de IoMusT. Como existem poucos exemplos, é difícil chegar a um consenso sobre quais características são desejáveis para eles. De tal maneira, o autor também se valeu de definições oferecidas pelas áreas que lhe permeiam, tais como performance musical em rede, arte interativa e música ubíqua (TURCHET et al., 2018).

A partir disso, constatou-se que uma boa estrutura de rede para apresentações artístico-musicais deve apresentar algumas características de cunho geral, como **baixa latência**, **interoperabilidade**, **escalabilidade**. Somado a isso, existem conceitos oriundos da performance musical pela rede, como fácil integração e participação dos usuários, ideias derivadas de ambientes de artes interativas, como integração com diferentes tipos de dados, e preceitos oriundos da música ubíqua, tais como capacidade de reconfiguração e implementação independente das ferramentas escolhidas pelos desenvolvedores.

Somado a isso, existem requisitos específicos para cada sistema. No caso do Sunflower, ele também deve ser qualificado para lidar com diferentes tipos de sinais, como áudio, vídeo e controle, expandir a capacidade de participação de não-musicistas e auxiliar no processo criativo. Por conta da dificuldade de atender a estes requisitos em um sistema real, o ambiente aqui apresentado foca em oferecer às três primeiras características citadas e em menor grau, se preocupa com os demais requisitos.

Outros dois atributos que devem ser considerados no Sunflower é o caráter **gerenciável** no momento da configuração e caráter **autônomo** na apresentação, onde eles conseguem determinar as próprias normas de conduta, sem imposições de outros elementos. Isso vai permitir que uma pessoa ou um grupo planeje e personalize a execução de serviços, definindo os parâmetros para as trocas de dados e interconectando-se a outros usuários e dispositivos, ao passo que apresentam uma política e gestão de conexão em comum.

Quanto ao perfil dos participantes neste ambiente, eles podem ser divididos em **usuários** e **administradores**. Para a primeira categoria, eles podem assumir três papéis distintos: **usuário final**, **usuário de sistema** ou **agente autônomo**. No primeiro caso, eles são marcados pela capacidade de interação com o sistema sem que haja a necessidade de conhecimento técnico

para tal, apresentando um número limitado de tarefas, enquanto no segundo caso o usuário tem um pouco mais de privilégio, conseguindo controlar elementos artísticos e de rede, além de fazer requisições mais específicas de acordo com suas necessidades, mas ainda sem todas as funcionalidades de um administrador. O terceiro e último caso trata dos elementos que determinam seu modo de comportamento e funcionam como sistemas isolados, sendo autogerenciáveis. Apesar de conseguirem se comunicar com outros objetos da rede, eles não podem controlá-los ou alterar suas propriedades.

O outro perfil diz respeito aos administradores, que controlam atividades e monitoram o uso dos recursos do ambiente. Suas tarefas básicas passam por obter informações da rede e tratá-las, além de diagnosticar e corrigir possíveis problemas. São chamados **administradores centralizados**, quando todo o gerenciamento é feito por uma única entidade, ou **administradores distribuídos**, quando o comando é realizado por diversas unidades em toda a rede.

Esta separação de privilégios fornece um pequeno grau de segurança ao sistema, porém funcional, indicando aqueles capazes de solicitarem ou responderem requisições, deixando os elementos presentes no ambiente livres para se comportarem de maneiras específicas e com capacidade de conexão com os dispositivos que são de seu interesse, ao passo que também oferece governabilidade e controle ao mesmo.

4.1.2 Características desejáveis para os dispositivos

Outro ponto importante a ser destacado é a composição física e o comportamento dos dispositivos presentes neste ambiente. Por se tratar de um campo com preocupações artísticas, os fatores estéticos, expressivos e ergonômicos também são relevantes (VIEIRA; ROCHA; SCHIAVONI, 2020). Com estes conceitos em mente, algumas características devem se fazer presentes, como:

- Heterogeneidade e Transparência
- Interface por linha de comando
- *Display* gráfico para troca de informações de controle
    - *Status* de conexão;
    - Número de identificação;
    - Protocolos suportados em cada objeto;
    - Número das portas utilizadas;
- Informações de entrada e saída
    - Formato de áudio e vídeo;
    - Profundidade de *bits*;

- Taxa de amostragem;

- Integração com *softwares* e *hardwares* legados
  - Integração por áudio;
  - Integração com dispositivos MIDI.

A **heterogeneidade** é possivelmente o aspecto mais sensível neste trabalho. Esta condição é um requisito de todo o sistema e indica que as coisas musicais devem possuir diferentes características e fabricantes. Para além disso, devem considerar as capacidades musicais e técnicas dos membros da audiência que vão utilizá-las, de forma que sejam acessíveis e manuseáveis por usuários com diferentes níveis de habilidade. Quanto às suas capacidades de conexão, elas devem ser adaptáveis as condições oferecidas pela rede. Isso está diretamente relacionado com a transparência, que se preocupa em oferecer recursos que contemplem todas essas necessidades.

Para garantir a **transparência** das coisas musicais para os usuários, é necessário que elas sejam configuráveis e isso pode ser feito tanto por um administrador quanto por um sistema inteligente. Caso a primeira opção esteja disponível, é interessante que ela ocorra através de **linhas de comando**. Essa abordagem é comum em ambientes computacionais, em especial para verificar alguns aspectos diretamente no sistema operacional e para reduzir o tempo e esforço de configuração de determinada tarefa. Apesar das suas vantagens, alguns contratempos podem aparecer, como exigência de maior grau de memorização, diversidade de comandos e alto nível de precisão, pois caso ocorra um erro de grafia ou entrada de comando inexistente, o trabalho precisará ser reiniciado. Ainda assim, ela aparece como uma boa opção para usuários gerais ao armazenar o histórico dos últimos comandos inseridos, agilizando e facilitando o uso da ferramenta; consumir menos memória RAM e ciclo de CPU; deixar o fluxo de trabalho mais dinâmico e simples; permitir que tarefas recorrentes sejam executadas sem tomar muito tempo e capacidade de funcionamento remoto (FELLMANN; KAVAKLI, 2007; UNWIN; HEIKE, 2000).

O ***display* gráfico** presta auxílio ao permitir a visualização das características operacionais e de rede de cada coisa musical, facilitando o processo de conexão e também prevenindo e corrigindo eventuais falhas (VIEIRA; SCHIAVONI, 2021a).

As **informações sobre as portas de entrada e saída** de cada objeto, bem como sobre seus diversos formatos de áudio e vídeo, mensagens de rede e protocolos usados para comunicação auxilia para que a heterogeneidade entre eles seja alcançada, permitindo detalhar seus aspectos técnicos e musicais e facilitando a conexão ao colocar em contato aqueles que têm algumas características em comum. Isso facilita tanto o gerenciamento manual de tais dispositivos quanto o gerenciamento automático.

A **integração com equipamentos legados** permite o uso de ferramentas que não foram criadas com a intenção de compor um ambiente de IoMusT, e que em muitas das vezes, podem estar obsoletas e/ou fora da linha de produção. Isso aumenta o interesse dos usuários pelo sistema,

visto que eles vão poder utilizar dispositivos com os quais já têm alguma familiaridade, além de não estarem suscetíveis a obsolescência programada. Para isso ser possível, são necessárias conversões entre diferentes formatos de áudio e também a comunicação por meio do protocolo MIDI. Isso implica em um sistema flexível, de forma que uma tecnologia nova ou antiga possa ser incluída independentemente de o mesmo ter sido desenvolvido para suportá-la.

Ademais, é importante garantir que estes dispositivos tenham características similares àqueles encontrados na área de IoT, como, por exemplo, que possam ser físicos ou lógicos; tenham mobilidade, apresentem conectividade sem fio, sejam configuráveis e manipuláveis de forma remota; possam ser personalizáveis, onde suas características são alteradas para melhor satisfazer o usuário, além de permitir integração com diferentes ferramentas; conseguir lidar com áudio em tempo real e apresentarem sensibilidade ao contexto.

4.2 Pensando em Pipes and Filters

Ao pensar em construir tal ambiente, algumas possibilidades de arquitetura de sistemas podem ser adotadas, como o modelo cliente-servidor ou o Pipes-and-Filters. Além de ser popular na engenharia de *software*, o modo de funcionamento deste último modelo também pode ser observado na música e no processamento de áudio (VIEIRA; SCHIAVONI, 2020). Um exemplo é o famoso sintetizador Moog (TROCCO; PINCH, 2004), constituído de módulos separados (semelhantes aos filtros), capazes de aplicarem modificações nos sons e enviá-los adiante pelo intermédio de cabos (que funcionam como os dutos). Outra equivalência está na forma como o sintetizador permite a reconfiguração destes módulos.

Uma segunda situação onde este comportamento está presente é no uso de pedais de efeitos e pedaleiras. Os fluxos de áudio oriundos de uma guitarra (função símile a de um filtro gerador) são transportados via cabos (dutos) até os pedais (filtros), que aplicam mudanças nas ondas sonoras e as enviam, novamente utilizando cabos, para os amplificadores e alto-falantes (filtros consumidores).

O mesmo padrão pode ser observado ainda em estúdios e em concertos, onde uma infinidade de microfones e instrumentos (filtros geradores) são ligados em mesas de som ou em um *patch bay* (filtros modificadores) que os interconectam com alto-falantes (filtros consumidores).

Esta arquitetura também se faz presente em linguagens de programação voltadas para a criação musical, como SuperCollider, ChucK, Mosaicode, FAUST e Pure Data, que recebem dados e processamentos em blocos de controle e os passam adiante utilizando dutos.

Frente a esta quantidade significativa de cenários com funcionamento congênere ao Pipes-and-Filters, somada à sua capacidade de reúso, substituição e evolução, propõe-se para o ambiente aqui idealizado, uma forma de funcionamento que siga estes mesmos moldes, apesar da implementação em rede ainda acontecer por meio do modelo cliente-servidor. Sendo assim, cada

coisa musical terá comportamento correlativo a um filtro, entidades independentes que recebem dados em suas entradas, os processam e os enviam para seus vizinhos, cujas propriedades não são previamente conhecidas. Neste contexto, o autor classifica-os em **filtros vertedouros**, capazes de gerar informação, **filtros sumidouros**, com funcionalidade única de consumir dados, **filtros conversores**, que transmutam um tipo de dado em outro e são úteis também para permitirem a integração de dispositivos e protocolos que não foram previstos pelo projetista do ambiente, expandindo ainda mais o seu leque de possibilidades para comunicação, e **filtros híbridos**, que desempenham duas ou mais dessas funções.

Os dutos, por sua vez, ficam responsáveis por somente transmitir dados, não aplicando nenhum tipo de processamento ou modificação a eles. São representados na figura dos cabos que transferem pulsos elétricos oriundos de instrumentos musicais e de microfones, e pelos meios de transmissão de dados na rede, como cabo de par trançado e Wi-Fi.

Ao trazer este modelo para a Internet das Coisas Musicais e classificar seus filtros de tal maneira, pode-se criar uma separação de papéis e supor o fluxo de informação no ambiente. Com isso, é possível definir que filtros sumidouros são implementados como **servidores** que irão ser conectados por filtros vertedouros, implementados como **clientes**. Já os filtros conversores e híbridos possuirão tanto servidores, em suas portas de entrada, quanto clientes, em suas portas de saída. Desta maneira, ao pensar em uma guitarra (filtro vertedouro de áudio), um pedal de efeito (filtro híbrido de áudio) e um amplificador (filtro sumidouro de áudio) podemos notar que a guitarra (cliente) poderá ser conectada tanto a entrada do pedal (servidor) quanto à entrada do amplificador (servidor).

Essa forma de pensar o ambiente garante que ele seja heterogêneo, já que diferentes dispositivos podem estar presentes e comunicar entre si, além de garantir facilidade de inclusão e/ou remoção de objetos e a possibilidade de classificação em sistemas independentes, que podem contribuir para a apresentação musical trabalhando de forma isolada ou agregando forças com outros elementos.

A Figura 7 ilustra o comportamento do ambiente. Note que os filtros são representados por diferentes cores a partir de seu modo de funcionamento, e que eles também apresentam diversos sentidos de comunicação.

Contudo, essa forma de estrutura se depara com três problemas decorrentes da diversidade de dados que trafegam por ela: i) o formato dos dados devem ser acordados entre os módulos; ii) possíveis sobrecargas podem ocorrer ao tentar padronizar dados; e iii) a incompatibilidade no formato dos dados pode dificultar o reúso de filtros.

Para solucionar este problema enquanto são mantidas as principais características e contribuições dos filtros e dutos, o Sunflower foi dividido em 4 camadas distintas e independentes, apresentadas em detalhes a seguir.

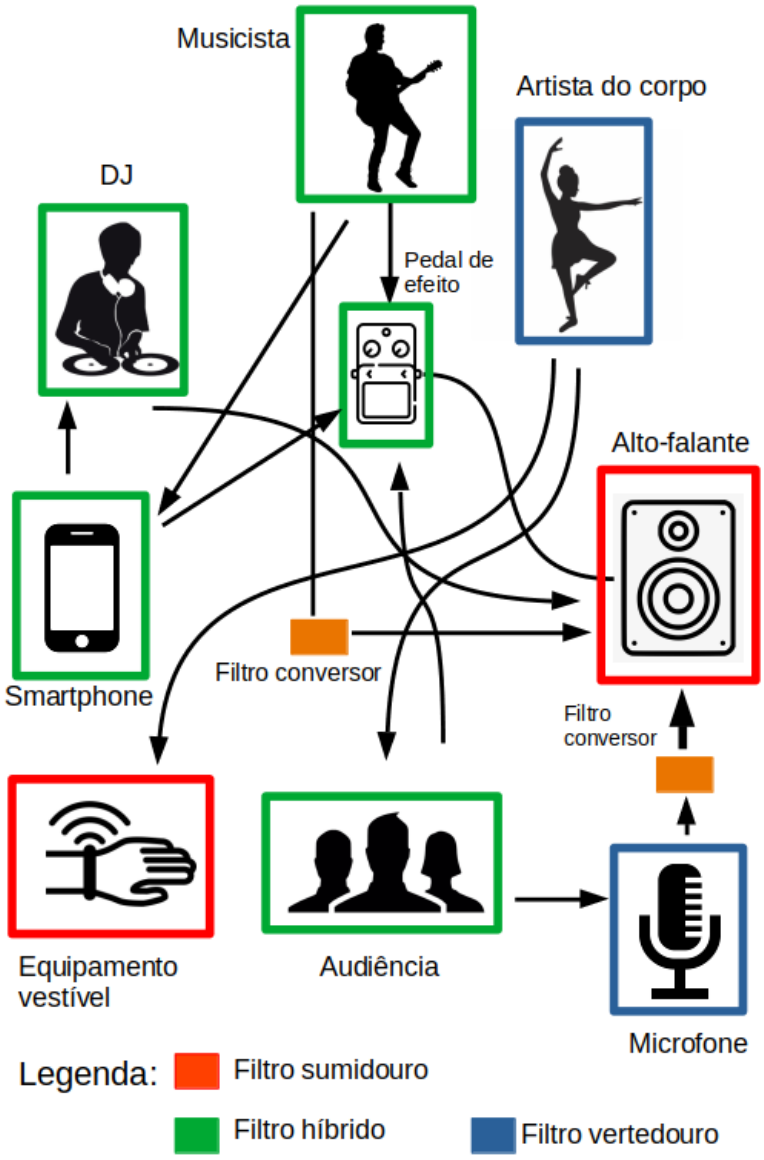


Figura 7 – Comportamento prático dos filtros.
Fonte: O autor

4.3 Pensando em camadas

Diversas ferramentas na área da ciência da computação utilizam arquitetura em camadas, como os sistemas operacionais e o modelo TCP/IP. Geralmente, essa estrutura é usada para isolar detalhes de implementação de uma camada para outra, fornecendo alguma capacidade de manutenção para o sistema. De tal maneira, os níveis no Sunflower também são independentes, embora não estejam dispostos de forma hierárquica nem precisem ser ordenados na forma em que são apresentados neste documento. As vantagens de se pensar o sistema dessa forma passam pelo seu caráter modularizado e pela sua facilidade de entendimento e implementação.

Cada camada é caracterizada conforme as coisas musicais, tipos de dados e protocolos presentes nela. Isso acontece porque cada dispositivo tem um circuito à parte, responsável por enviar as mais diversas informações, como áudio, MIDI, OSC, vídeo e afins. É importante

destacar que os dispositivos estão dispostos em suas categorias a partir dos tipos de dados que trocam na rede. Isso culminou numa divisão em **camada de áudio digital**, **camada gráfica**, **camada de controle** e **camada de gerenciamento**, onde cada uma delas fornece uma abstração sobre o trabalho necessário para realizar certas tarefas no ambiente. Elas serão explicadas em detalhes nas seções subsequentes.

Essa divisão baseada nos dados que circulam pelas camadas se faz necessária por conta da diversidade e quantidade de informação presente neste ambiente. Desse modo, tudo que é áudio circula na camada de áudio, tudo que é informação visual circula na camada gráfica e assim por diante. Isso irá refletir em outra propriedade deste sistema, que é ter um endereço IP *multicast* específico para cada tipo de informação, dividindo-os em canais. Este formato de arquitetura também permite a integração dos elementos oriundos do Pipes-and-Filters de forma não-automática. Sendo assim, abre-se espaço para pensar como devem se comportar as entradas e saídas desses dispositivos, o que ajuda a melhorar os conceitos de implementação deste trabalho. Ressalta-se ainda que apesar dessa divisão, uma camada pode trocar dados com outra e um aparelho pode estar presente em uma ou em várias delas. É importante destacar ainda que o autor se preocupou em utilizar conceitos e termos presentes na literatura, por isso a escolha do nome "camada", mas, na prática, elas se comportam como módulos, sem uma organização e troca de dados hierárquicas.

4.3.1 Camada de Áudio Digital

A camada de áudio digital, como o nome indica, é responsável por gerar dados sonoros para o ambiente, suportando áudio no padrão PCM, por exemplo. Uma importante característica desta camada é a capacidade de dividir o processamento entre seus componentes, podendo adicioná-los ou removê-los de forma que garanta a escalabilidade do sistema e a criação musical de forma compartilhada. Isso pode causar melhorias nas interações que acontecem em apresentações artísticas, em instalações, em gravações no estúdio, no *setup* de bandas, orquestras e afins.

Quanto aos papéis presentes nesta camada, os musicistas podem tanto enviar o áudio de seus instrumentos eletroacústicos para a rede quanto receber informações dessa maneira, além de terem novas formas de criação graças às coisas musicais. Atores, dançarinos ou qualquer outra pessoa que use o corpo para expressão artística, podem gerar sons a partir de determinados movimentos, enquanto o público ajuda na composição utilizando seus próprios dispositivos, enviando e recebendo dados dos demais elementos do ambiente. O engenheiro de som terá um papel similar ao que desempenha em apresentações tradicionais, com a diferença que agora estará munido de mais recursos para realizar o seu trabalho.

Já os exemplos de filtros presentes nesta camada são observados a seguir:

- **Filtros vertedouros**: instrumentos musicais, microfones, sintetizadores, metrônomos,

diapasões, reprodutores de arquivos sonoros e outros elementos que geram sinais de áudio;

- **Filtros sumidouros**: afinadores, caixas de som, gravadores, analisadores de espectro e outros elementos que apenas consomem sinais de áudio;
- **Filtros conversores**: realizam ajustes no formato de áudio, como mudança de taxa de amostragem, mesclagem de canais, mudança na profundidade de *bits*, tamanho de janela ou outras modificações no formato do áudio para aumentar a possibilidade de conexão entre dispositivos que possuem entradas e saídas incompatíveis (ex: recebe áudio a 44100 Hz e entrega a 48000 Hz);
- **Filtros híbridos**: efeitos de áudio, equalizadores, mesas de mixagem, conversores de sinais e outros elementos que recebam áudio e proporcionem alguma transformação no sinal e o entregue na saída.

4.3.2 Camada Gráfica

Outro aspecto que contribui para apresentações musicais são os elementos gráficos, que vão desde vídeos capturados em tempo real que destacam o comportamento dos musicistas ou elementos não observáveis a olho nu, até animações, figuras, efeitos especiais e informações que podem ser repassadas de forma visual, corroborando tanto para a estética do espetáculo quanto para uma maior participação do público. Para isso ocorrer, a imagem deve passar por um processo de codificação e descodificação, de forma que esses dados possam ser interpretados pelos sistemas computacionais. Nesta conjuntura, surge a camada gráfica do Sunflower, responsável por este procedimento e por garantir que o sistema seja capaz de lidar com este tipo de linguagem, incumbindo-se das representações visuais.

Esta camada também proporciona maior participação do público no espetáculo, pois além de ser outra funcionalidade passível de controle, recebendo inserções e modificações gráficas, mudanças nos padrões de cores, na resolução e afins, também pode oferecer informações, como partituras, letras das músicas e movimentos a serem seguidos.

Os musicistas podem se valer desta camada para receberem dados e para trocarem informações entre si e com o público. Para mais, utilizam estes recursos visuais para aumentarem a narrativa da obra. Os artistas do corpo, uma vez mais, podem criar cenas a partir de seus gestos e também podem exibir movimentos a serem mimetizados pelo público. A audiência, além de todas as possibilidades presentes na camada de áudio, conseguem ainda enviar imagens e vídeos a serem exibidos nas telas, mudarem o padrão de cores, a resolução, o formato desse material gráfico e assim por diante. Os engenheiros de som podem enviar informações aos musicistas através dessa camada, bem como organizá-las, assim como faz com o áudio.

São exemplos de filtros que atuam nesta camada:

- **Filtros vertedouros:** câmeras de vídeo, sintetizadores de imagens e outros elementos que geram sinais visuais;
- **Filtros sumidouros**: reprodutores e gravadores de vídeo, telas, projetores e demais elementos que apenas consomem estes dados;
- **Filtros conversores**: fazem ajustes no formato de vídeo, como mudança nos quadros por segundos, largura, altura, resolução de cores e outras modificações no formato, de modo a aumentar a possibilidade de conexão entre dispositivos que possuem entradas e saídas incompatíveis (ex: a entrada recebe um vídeo em cores, com resolução de 800x600 e entrega o conteúdo em preto e branco, a uma resolução de 320x480);
- **Filtros híbridos**: processadores e efeitos de vídeo, agregadores de sinais e mecanismos que causam alguma transformação no sinal.

4.3.3 Camada de Controle

A sincronização é um fator essencial para dispositivos que trocam dados pela rede, assim como para musicistas, que baseiam suas ações em eventos realizados em um intervalo de tempo. Sendo assim, cada coisa musical presente neste ambiente deve se comportar dessa maneira, além de conseguir enviar/receber dados e controle pela rede. Isso implica em um controle de forma remota, capaz de alterar propriedades como volume, frequência, BPM, etc.

Este procedimento de configuração poderia ser realizado através de mensagens de texto (como o TXT, por exemplo), XML, ou algum outro formato passível de encapsulamento de dados para o tráfego em rede sendo que o escolhido para desempenhar esta função foi o protocolo de controle da informação musical OSC. A grande vantagem do OSC é padronizar o envio de dados usando encapsulamento numérico, o que não é possível nos formatos textuais. Para exemplificar, imagine que um usuário queira mandar o valor de π pela rede. Ele irá gastar um *byte* por caractere no padrão ASCII, enquanto que no formato Unicode esses valores serão ainda maiores. Assim, o π com 11 casas decimais (3,14159265359) irá ocupar 13 *bytes* tanto em sua codificação em XML quanto em TXT. Já sua representação numérica usando o padrão *float* IEEE 754 vai ocupar 4 *bytes*, não importando a quantidade de casas decimais. Dessa forma, fica claro que a mensagem OSC terá um custo muito menor. Além disso, as mensagens OSC possuem um endereçamento hierárquico que permite criar espaços de nome com um custo também mais baixo, uma vez que o encapsulamento do XML depende de *tags* e atributos.

Entretanto, ao se usar este protocolo deve-se ter em mente alguns de seus empecilhos. Ao permitir que cada usuário projete o seu próprio registro em vez de lhe fornecer um conjunto predeterminado, ele pode ser aplicado em situações nunca imaginadas e ocasionar erros. Um fator relacionado ao espaço de nome é que ele é gerado como *string*, que apesar de ser mais eficiente para avaliação humana, gera um maior custo computacional (FRAIETTA, 2008). Algumas de suas características, como ser um protocolo sem estado, a baixa delegação de eventos e os poucos

tipos de mensagens disponíveis também atrapalham a produtividade e podem entrar em conflito com certas aplicações (ROSE; HAIGHTON; LIU, 2015).

O controle proveniente desta camada não precisa ser necessariamente oriundo de uma coisa musical criada especificamente para isso. Ele pode decorrer de outros dispositivos, como, por exemplo, um teclado musical que ao executar determinada sequência de notas altera alguma propriedade de vídeo, ou uma câmera que ao capturar certa cor dispara uma sequência na bateria.

Quanto ao papel dos musicistas, eles podem controlar seus equipamentos ou o de colegas pela rede, garantindo-lhes maior governabilidade neste quesito, além de uma expansão no seu potencial artístico. O público pode utilizar os conceitos aqui apresentados para interferir nos efeitos sonoros e gráficos, de modo que os artistas terão que se adequar a este novo ambiente, criando uma peça totalmente inédita. Esta forma de controle sobre a apresentação permite que até quem não está fisicamente presente no espetáculo possa participar e interferir nos seus rumos. O engenheiro de som será capaz de mixar e alterar os parâmetros sonoros e gráficos pela rede e inserir elementos recém-baixados da internet, ganhando uma importância crescente neste contexto.

Os exemplos de filtros nesta camada são:

- **Filtros vertedouros**: *sliders*, botões, *knobs*, controladores e outros elementos que geram sinais de controle;
- **Filtros sumidouros**: efeitos, mesas de mixagem, metrônomos, afinadores e outros dispositivos que utilizam controles para modificar sua configuração durante seu uso;
- **Filtros conversores**: fazem ajustes no nome de espaço, faixa de valores e outras modificações no formato das mensagens de controle, aumentando a possibilidade de conexão entre equipamentos que possuem entradas e saídas incompatíveis;
- **Filtros híbridos**: arpejadores, sequenciadores e outros dispositivos que recebem um sinal de controle, modificam o mesmo e o enviam para a saída.

4.3.4 Camada de Gerenciamento

Em apresentações musicais, é comum a presença de um técnico de som para prover suporte aos musicistas. De forma parecida, as redes de computadores necessitam de um administrador, responsável por configurar e manter esta infraestrutura e também fornecer suporte aos usuários. No Sunflower, os dois papéis são indissociáveis, sendo o administrador responsável tanto por garantir a usabilidade da rede quanto para assegurar que os aspectos sonoros e gráficos estejam nos requisitos esperados pelos artistas. Para isso, é necessário que as coisas musicais possam ser mapeadas e que suas características principais cheguem ao administrador. Entretanto, esta tarefa pode ser complexa, uma vez que muito dos dispositivos presentes nestes contextos

nem sequer apresentam uma interface gráfica, unidade de processamento ou alguma porta de entrada e saída de dados que forneça informações sobre os mesmos. Somado a este fato, existe a imprevisibilidade de eventos abertos ao público e os problemas decorrentes da heterogeneidade do ambiente, que agrega dispositivos com diferentes formatos de áudio, taxas de amostragem, profundidade de *bits*, resoluções de vídeos, sistemas de cor, protocolos de comunicação, controle, etc.

Uma maneira de lidar com estas questões é através de um **gerenciamento** dos objetos pela rede, onde o administrador consegue ver quem está presente e quais suas principais características, de modo a interligar somente aquelas que podem trocar dados entre si. Para isso ocorrer, cada coisa musical, ao se juntar à rede, deve informar seu número de identificação, portas e protocolos usados na comunicação e formato de áudio e/ou vídeo aceitos, de modo a permitir e facilitar a conexão entre os dispositivos. Quando se desconectam, também devem publicar uma mensagem informando isso, de maneira a permitir sua **configuração** antes de o mesmo ser efetivamente utilizado. Estas operações são feitas na quarta e última camada do Sunflower, a camada de gerenciamento.

As conexões conduzidas por esta camada se dão de forma arbitrária. Isso quer dizer que um filtro vertedouro só será capaz de se conectar com aqueles do tipo sumidouro ou híbrido. Caso ele tente se conectar com um filtro do mesmo tipo, nenhuma informação será repassada à rede, já que eles não conseguem executar dados, somente gerá-los. Se o usuário não tiver conhecimento prévio de quais elementos se comportam como sumidouros, ele pode tentar alcançá-los por tentativa e erro, enviando mensagens para a porta que ele deseja se conectar. Se ela não for apta a receber informações ou já estiver em uso, a conexão será negada. É nesta camada onde o ambiente é preparado para atuar nos moldes da arquitetura Pipes-and-Filters, devido ao fato dele só ser possível diante a existência de um administrador.

Nesta camada, destacam-se os papéis dos dispositivos, que podem ser classificados em duas categorias: **gerenciáveis** e **gerenciadores**. Os que se encontram no primeiro grupo são aqueles que recebem algum tipo de controle. Já os elementos na segunda categoria são responsáveis justamente por exercerem esse controle. Vale frisar que nem todos os usuários e objetos precisam ser gerenciadores, mas ao menos um participante deve se incumbir dessa função, de modo a tornar o ambiente utilizável. A vantagem de pensar o sistema dessa maneira consiste em uma maior versatilidade na gestão dos equipamentos e também permite maior interatividade por parte dos usuários, que de forma remota ou local, conseguem alterar as propriedades de tais aparelhos.

Quanto aos papéis dos artistas e do público, eles são mais limitados nesta camada do que nas demais, dado o fato que ela foi criada para facilitar o gerenciamento das coisas musicais e da rede. Sendo assim, das funções inicialmente pensadas, quem mais atua é o engenheiro de áudio, responsável por conectar e desconectar dispositivos e impedir que outros se comuniquem.

Por último, a camada deve permitir que suas configurações sejam salvas, proporcionando

uma configuração automática e evitando que toda vez que um mesmo grupo de artistas ou objetos se reúna, todo o arranjo tenha que ser feito a partir do zero.

4.4 Proposta de Protocolo

As camadas acima mencionadas são responsáveis por prover serviços à rede. Estes serviços indicam as operações que determinada camada está apta a realizar em nome de seus usuários, mas não informa como estas ações são implementadas. Neste sentido, é importante a existência de um protocolo que represente um conjunto de regras que controla o formato e o significado das mensagens que circulam pelo sistema. É por conta desses protocolos que, por exemplo, não se pode conectar um cabo Ethernet em uma guitarra e esperar que funcione, porque as normas que tratam dos meios físicos e elétricos são diferentes.

Protocolo de configuração

No Sunflower, cada entidade em cada camada utiliza de um protocolo de configuração para implementar suas definições de serviço. Esta regulamentação é simples e deve ser seguida pelos componentes, onde seus principais fundamentos são:

- Permitir que qualquer nó indique seu estado para o administrador;
- Permitir que este mesmo estado esteja sempre sincronizado com a rede;
- Fazer com que cada nó exiba suas principais características e portas aptas tanto para receber quanto para enviar dados;
- Conectar somente um par ou conjunto de nós que realmente suportam esta conexão;
- Informar que está sendo configurado ou desconectado da rede.

A partir dessas definições, o protocolo de configuração do Sunflower foi dividido em 5 etapas, apresentadas e conceituadas a seguir.

Passo 1: Descoberta

Nesta primeira etapa, o Sunflower irá se comportar como um protocolo de descoberta tradicional, procurando por outros nós na rede e estabelecendo contato entre eles. Para isso, cada coisa musical, ao se conectar na rede, deve informar através de uma mensagem OSC *multicast*, no IP 224.0.0.0 e porta 60000, seu número de identificação único, endereço IP e nome, conforme o padrão exibido na sequência.

- **Informando credenciais:** */hello/número_ID/endereço_IP/nome*

Essa mensagem precisa ser entregue a todos os dispositivos do ambiente, que por sua vez, respondem com uma mensagem no mesmo formato para também informar suas credenciais. Isso elimina a necessidade de intermédio do administrador para toda e qualquer troca de dados no sistema.

Na sequência, deve-se noticiar as portas de entrada, responsáveis por receberem dados. A mensagem a ser enviada apresenta o seguinte molde:

- **Informações de entrada:** */input/número_ID/tipo_de_porta/tipo_da_coisa_musical/ taxa _de _amostragem/ profundidade_de_bits/formato_arquivo_de_áudio/número_de_canais/ IP_do _remetente/ número_da_porta/descrição_da_porta/protocolo_de_rede /protocolo_de _informação _musical /nome*

As portas de saída devem alcançar nós capazes de receberem estes dados. Para isso, utilizam o padrão visto logo abaixo. Vale destacar que cada elemento difere de outro por conta de suas funcionalidades e/ou das especificidades incluídas por cada desenvolvedor, mas estas mensagens são abrangentes o suficiente para informar a todo o sistema as principais características de cada dispositivo presente nele, criando os parâmetros necessários para ocorrer alguma comunicação.

- **Informações de saída:** */output/número_ID/número_da_porta_remetente /tipo _de_porta _remetente /descrição_porta_remente/número_da_porta_destinatário/nome_destinatário*

Passo 2: Configuração individual dos dispositivos

A segunda etapa do protocolo permite que eventuais mudanças nas configurações de rede e/ou dados dos dispositivos possam ocorrer. Isso implica em, por exemplo, mudar o número da porta de entrada de um elemento que se comporta como filtro sumidouro para evitar conflitos, ou ainda, mudar a taxa de amostragem, espaço de nome da mensagem OSC ou até mesmo a resolução do vídeo para se adequar às capacidades de determinada coisa musical onde se pretende comunicar. Desse modo, todos os atributos informados na publicação de recursos podem/devem conter mensagens de configuração, que serão respondidas com comunicados de atualização do ambiente, indicando a situação atual de cada elemento. Tal resposta deve ser exibida até mesmo se o dispositivo não for modificado ou para indicar que tal configuração não é possível.

Estas mensagens podem adotar o seguinte padrão:

- **Alterando configuração de rede:** */network/número_ID/nome/número_IP/porta_atual /nova_porta/grupo_multicast_atual/novo_grupo_multicast*
- **Configurando dados de áudio:** */setup_audio/número_ID/nome/número_IP /taxa _de _amostragem /profundidade_de_bits/formato_arquivo_de_áudio/número_de_canais*

- **Configurando dados de vídeo:** */setup_video/número_ID/nome/número_IP/codec/ resolução/padrão_de_cores/ formato_arquivo_de_vídeo*

Também faz parte deste passo 2 as mensagens de criação de dispositivos lógicos, que nada mais são que uma replicação de uma coisa musical presente no sistema. Desta maneira, após determinado artefato receber esta mensagem, ele gera uma cópia modificável de si mesmo, permitindo que as características de áudio, vídeo (quando possível) e rede sejam modificadas para satisfazerem as necessidades dos usuários. Após confirmada a duplicação de um elemento, o sistema deve responder com uma atualização de estado, semelhante àquela enviada quando as configurações das coisas musicais são alteradas. A mensagem que cria esta cópia pode ser vista abaixo.

- **Criando dispositivos lógicos:** */create/número_ID/nome/número_IP/nova_coisa_musical/ especificações_de_áudio_ou_vídeo*

Caso os usuários desejem a exclusão de um dispositivo lógico, ele deve enviar uma mensagem no exemplar visto na sequência. Como resposta, novamente terá uma atualização de estado.

- **Excluindo dispositivos lógicos:** */delete/número_ID/nome/número_IP/nova_coisa_musical/ especificações_de_áudio_ou_vídeo*

Passo 3: Configuração e conexão do ambiente

Com os dispositivos configurados, chega-se na etapa de configuração e conexão do ambiente. As mensagens devem, portanto, informar as suas credenciais, porta por onde sairão os dados e a porta de entrada do dispositivo que recebe estas informações, conforme o padrão ilustrado.

- **Conectando entradas e saídas específicas:** */connect/número_ID/nome/número_IP /número_saída/número_porta_entrada*

Esta mensagem indica que uma coisa musical A utiliza uma porta de saída X para enviar dados para uma coisa musical B, que os recebe na porta de entrada Y. Caso a conexão seja confirmada, o sistema responde informando este novo *status*, caso contrário, ele indica que a interligação foi negada, algo que pode acontecer tanto pelo dispositivo já estar conectado ou por tal ação não ser possível.

Passo 4: Mudança de modo

As coisas musicais podem seguir outro método de funcionamento quando estão sendo utilizadas na prática, chamado **modo de performance**. Nele, tais dispositivos não aceitam mais configurações e alterações remotas, o que garante maior estabilidade técnica e artística aos usuários, ao passo que também facilita a gerência da rede, visto que ela terá de lidar com menos trocas de informações, garantindo maior segurança aos usuários e aos dados trocados. Além disso, pode omitir o Wi-Fi e/ou portas de rede dos objetos, de maneira a evitar que pessoas não autorizadas tentem acessá-lo.

Isso acarreta na necessidade da troca de um modo para outro, ou seja, passar de configuração para performance e vice-versa. Neste contexto, surge o quarto e penúltimo passo no protocolo do Sunflower, que trata justamente dessa mudança de comportamento das coisas musicais. As seguintes mensagens são as responsáveis por isso.

- **Configuração para Performance:** */setup/número_ID /nome/ número_IP/status_atual/ novo_status*
- **Performance para Configuração:** */performance/número_ID /nome/ número_IP/ status_atual/ novo_status*

Quando a primeira mensagem é enviada, ela altera todo o ambiente e não uma única coisa musical. Assim, os dispositivos recebem uma resposta que indica que eles agora estão imodificáveis, ou seja, não mais passíveis de controle. Vale frisar que estes elementos ficam somente inacessíveis para configuração, mas ainda conseguem enviar ou receber dados de seus vizinhos. Já na segunda mensagem, a resposta é similar de quando um objeto se conecta a primeira vez ao ambiente, somada por uma atualização de estado do mesmo.

Esta etapa do protocolo pode comprometer a dinamicidade do ambiente, ao restringir as possibilidades de interação dispositivo. Em contrapartida, ela garante maior determinismo ao musicista, garantindo-lhe a manutenção de certas propriedades, sejam elas referentes a rede ou aos dados que ele irá trocar. Destaca-se também que esta mudança de modo não é obrigatória, mas ela traz ao sistema mais possibilidades de configuração. A presença de um "super administrador" que consegue alterar aspectos no objeto mesmo quando ela está no modo performance, ajudaria a corrigir eventuais falhas e pode ser uma solução para esta questão.

Passo 5: Encerramento

Finalmente, a última etapa da proposta do protocolo trata da desconexão de uma coisa musical da rede. A mensagem que comunica isso é exibida a seguir.

- **Mensagem de encerramento:** */goodbye/número_ID/número_IP/nome/status*

É importante alertar que esta mensagem se difere daquela de desconexão, recebida quando a coisa musical comunica que não quer mais integrar o ambiente. Para o caso observado na quinta etapa, ocorre um encerramento definitivo, onde o dispositivo não é mais capaz de trocar dados. A Figura 8 resume todas as informações exibidas nesta Seção.

Protocolo de performance

Este protocolo é decorrente da mudança de modo de funcionamento na configuração dos dispositivos e é responsável por versar sobre os tipos de dados trocados nas camadas do Sunflower. Embora o formato de áudio seja PCM, ele pode adotar várias configurações em suas taxas de amostragem e profundidade de *bits*. O mesmo se repete na camada gráfica, que aceita os formatos AVI, QuickTime e MPEG, mas pode ter diversos padrões de cores e resoluções no produto final, e na camada de controle, que opera majoritariamente utilizando o protocolo OSC, mas consegue encapsular outras mensagens e protocolos em seus pacotes. Isso resulta em diversas possibilidades de implementação e ferramentas para cada uma das camadas, não sendo intenção dos autores limitar este comportamento. Alguns dos tipos de dados presentes nas camadas são sintetizados na Figura 9.

Enfatiza-se ainda uma divisão em canais para este ambiente. De tal maneira, o IP *multicast* 224.0.0.0 é usado para dados de controle, enquanto no endereço 239.255.255.255 navegam os dados de áudio e em 224.0.0.1 as informações de vídeo.

Figura 8 – Protocolo de comunicação do Sunflower.
Fonte: O Autor

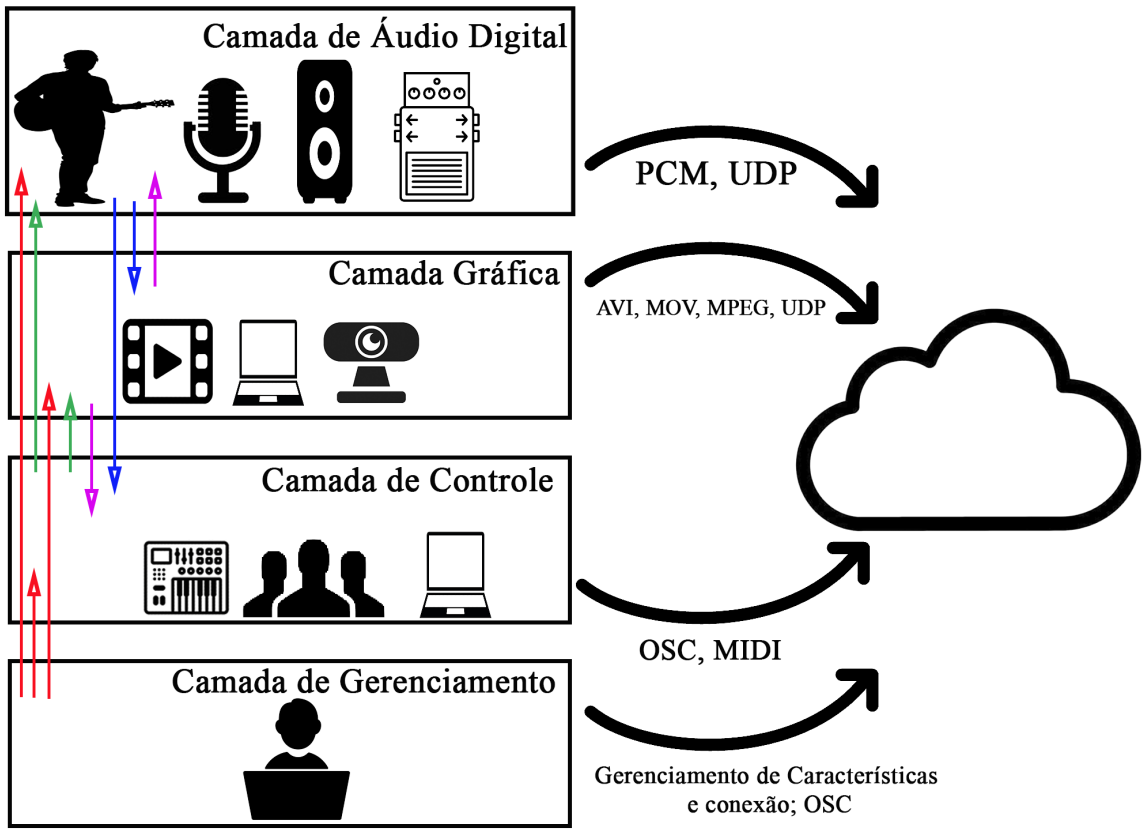


Figura 9 – Divisão em camadas do Sunflower.
Fonte: O Autor

5 IMPLEMENTAÇÃO E TESTES

Para validar o ambiente proposto, foram criados diversos protótipos de coisas musicais, de modo a simular instrumentos eletroacústicos, equipamentos comuns a apresentações artísticas e dispositivos com características próprias. Cada um deles tem um número de identificação único e consegue fornecer suas portas, dados de entrada e saída para os demais elementos presentes no ambiente. Os usuários podem ter acesso a eles no diretório oficial do Sunflower no GitHub[1], bem como modificá-los para que se encaixem em seus propósitos artísticos. Podem ainda utilizar coisas musicais desenvolvidas por eles próprios, desde que respeitem as condições propostas.

Esta prototipação contou ainda com elementos desenvolvidos em diferentes ambientes e linguagens de programação, debatidos na sequência. Ao fim de tudo, foram realizados testes de rede, utilizando a comunicação em *localhost*, cabo de par trançado e Wi-Fi, a fim de averiguar o comportamento do sistema.

Reforça-se que o Sunflower é o *design* do ambiente, ou seja, ele versa sobre sua arquitetura, módulos, interfaces, serviços e protocolos. Esta aplicação é apenas uma reafirmação dos conceitos apresentados.

5.1 Implementando o *design* do Sunflower e utilizando o Pure Data

Considerando a estrutura e os protocolos que regem o funcionamento do Sunflower, foram concebidas, de forma prática, suas quatro camadas. O sistema conta com 27 protótipos de coisas musicais, eles são: *player* de áudio, *drum machine*, *patch* que capta áudio de microfone e guitarra/baixo e o envia para a rede, alto-falante, diapasão, gravador, produtor e reprodutor de vídeo. Cada um deles também tem um ou vários *patches* de controle, a depender de suas particularidades. Como alguns comportam também *subpatches* (que atuam como dispositivos lógicos), o número final de coisas musicais no Sunflower é 37. Estes protótipos foram desenvolvidos na linguagem Pure Data (versão 0.50.2), escolhida por ser multiplataforma, aberta, de fácil programação e por estar presente em diversos sistemas que lidam com música pela rede ou em tempo real.

Quanto ao envio de dados para a rede, ele acontece por meio dos protocolos UDP e OSC. Isso foi possível graças ao uso de uma *external*[2] chamada **mrpeach**, capaz de encapsular pacotes com estes tipos de dados e enviá-los para a rede de forma simples e rápida.

Para garantir a interoperabilidade do sistema, cada coisa musical deve informar suas principais características funcionais ao se conectar, indicando qual a sua taxa de amostragem,

[1] <https://github.com/romulovieira-me/sunflower_iomust_environment>

[2] Função adicional para a linguagem desenvolvida pela comunidade, com funcionamento análogo às bibliotecas de uma linguagem tradicional.

profundidade de *bits*, número de canais usados para comunicação, portas usadas no processo e assim por diante. Um trecho de um *patch* da camada de áudio pode ser visto na Figura 10. Ele captura o áudio oriundo de um microfone externo conectado ao computador e o envia para a rede ao invés de um alto-falante. É passível de controle, de forma remota, em seus níveis de volume e reverberação.

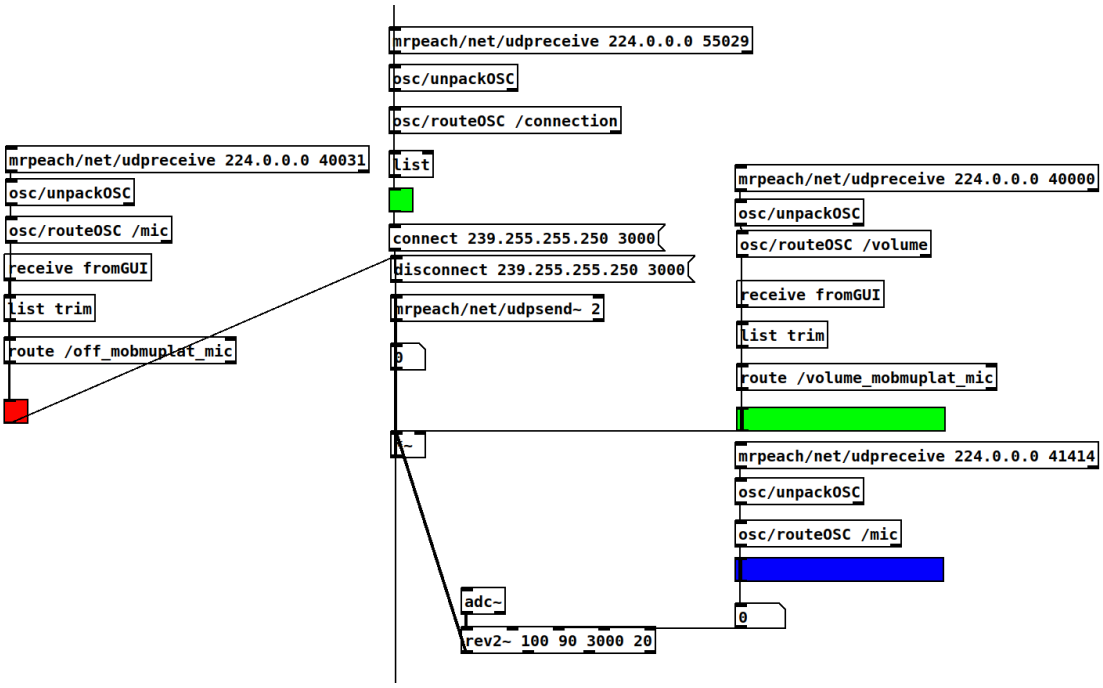


Figura 10 – Trecho responsável por capturar o áudio de um microfone e enviá-lo pela rede.
Fonte: O autor

Para assegurar a possibilidade de integração de instrumentos e *softwares* legado no ambiente, tal camada também consegue lidar com mensagens MIDI, desde que estejam empacotadas no formato citado. Este comportamento é observado na Figura 11.

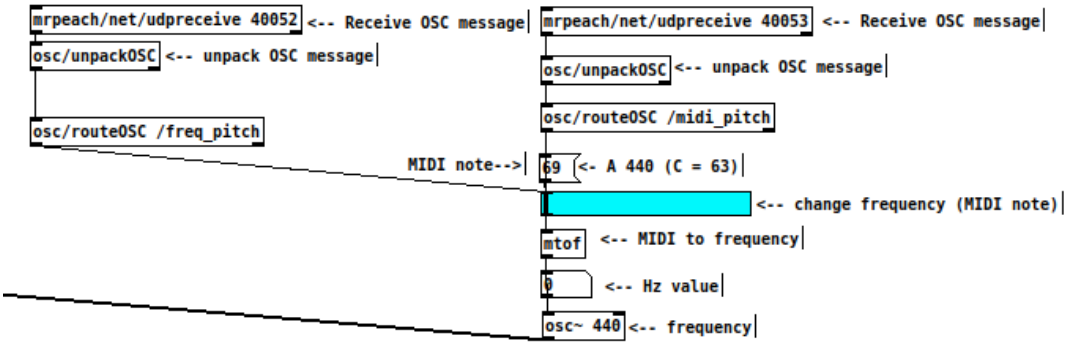


Figura 11 – Encapsulamento de informações MIDI em pacotes OSC.
Fonte: O Autor

Para a camada gráfica, os autores recorreram novamente às capacidades multimédias do Pure Data, usando o *external* **GEM** para capturar e reproduzir dados de vídeo oriundos de DVD *players*, *webcams*, *laptops* ou qualquer outro meio compatível, além de realizar síntese e

processamento de imagens em tempo real. A qualidade de imagem está mais relacionada com o equipamento que a registra, e a exemplo do áudio, também circulam pela rede por meio do protocolo UDP. Um exemplo de *patch* presente nesta camada é exibido na Figura 12.

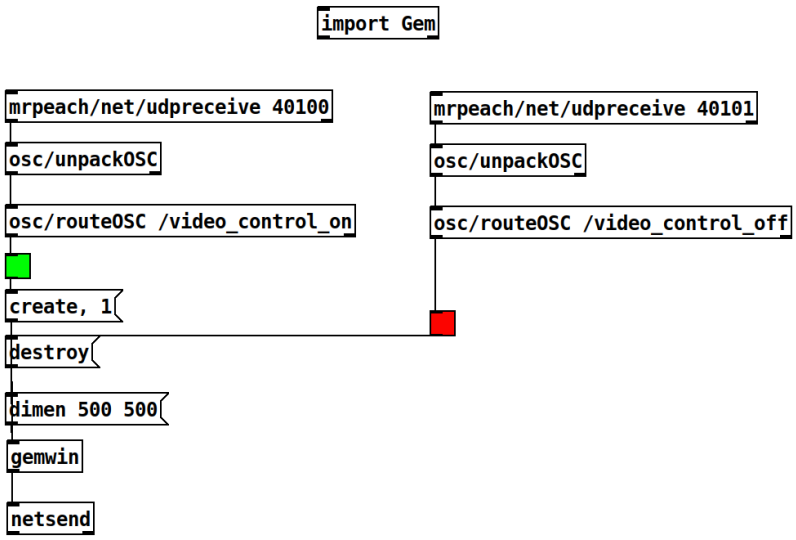


Figura 12 – *Patch* responsável pelo vídeo no Sunflower.
Fonte: O autor

O **gemwin** é o objeto que cria a janela onde os vídeos aparecem. É responsável também por controlar os quadros por segundo em cada amostra visual, descartando aqueles que demoram a ser renderizados. Já o **dimen** permite que o usuário defina a resolução da imagem de vídeo. Se for uma resolução muito pequena, haverá menos *pixels* para computar e, portanto, será mais veloz, ao passo que também haverá perda na qualidade da imagem. Se a resolução escolhida for muito alta, o oposto acontece, com computação lenta dos dados e imagem de melhor qualidade. O padrão é utilizar 25 quadros por segundo, mas vídeos com mais de 15 quadros por segundo já possuem a ilusão de movimento garantida (GAINSFORD, 1993).

5.2 Utilizando o Processing

Para expandir ainda mais as possibilidades artísticas e interativas deste sistema, foram elaboradas quatro artes na linguagem de programação Processing para completar a camada gráfica. Esta linguagem foi criada por Ben Fry e Casey Reas em 2001, e é, originalmente, uma modificação e simplificação da linguagem Java, removendo os aspectos originais que exigem um conhecimento mais profundo sobre programação. Dessa forma, ela se torna leve e de fácil aprendizagem, ideal para trabalhos visuais e artes eletrônicas (REAS; FRY, 2010).

No projeto em questão, foi utilizada na versão 3.5.4. É passível de alterações nas cores, velocidades e direções das artes criadas graças a biblioteca **oscP5**, que permite o recebimento

de mensagens OSC responsáveis por essas modificações, o que proporciona aos usuários novas formas de interação e controle no ambiente. O resultado obtido é exposto na Figura 13.

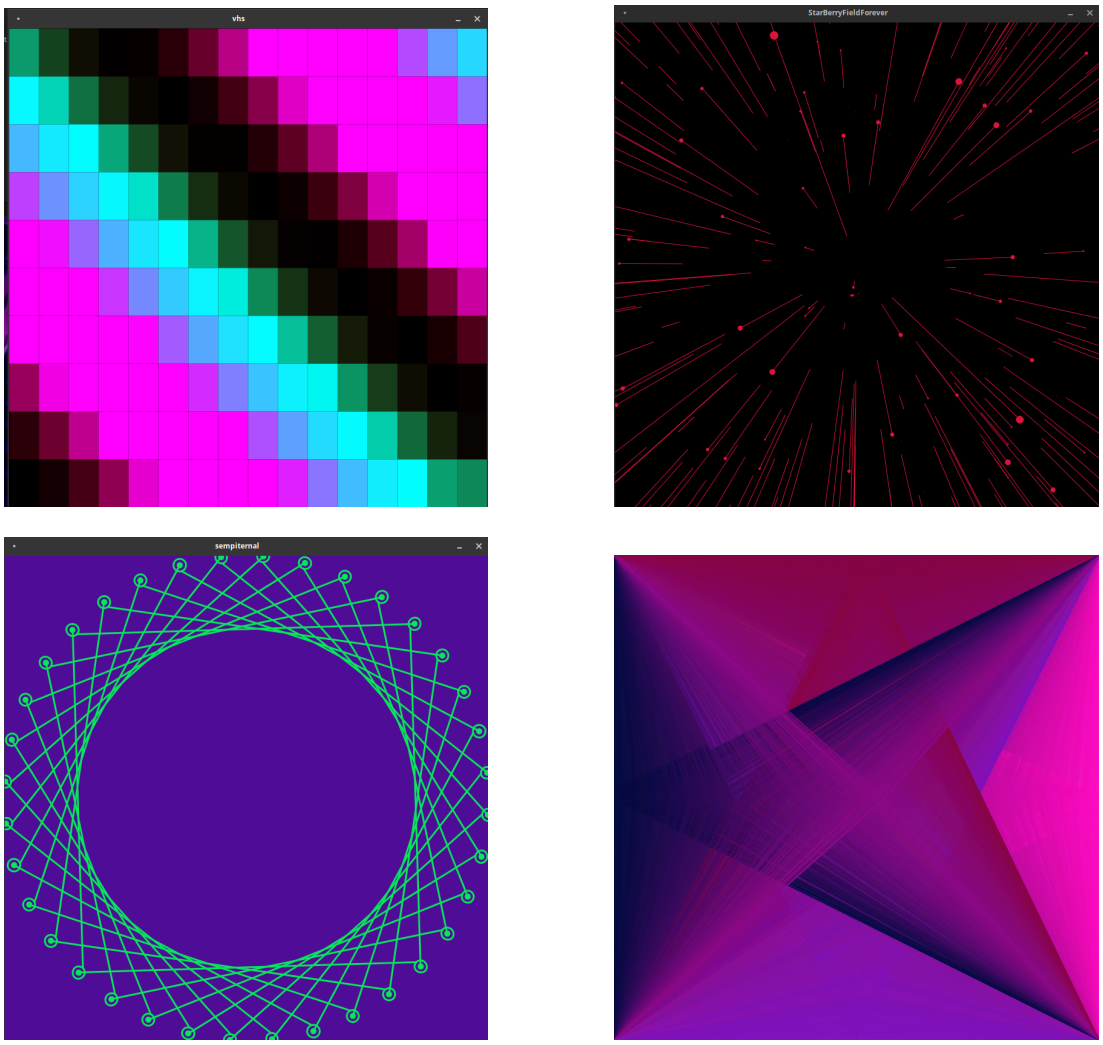


Figura 13 – Artes presentes na camada gráfica.
Fonte: O autor

5.3 Utilizando Python

Na camada de controle, cada coisa musical pode ser gerenciada pela rede. Isso significa que elas podem ser ligadas ou desligadas de forma remota, bem como ter volume, frequência, BPM e afins alterados da mesma forma. Para cumprir e administrar estes e outros requisitos, foi criada uma CLI.

Ela foi desenvolvida na linguagem Python (versão 3.8.10) e contou com os módulos **argparse**, para criação das opções na linha de comando; **pythonosc**, responsável por receber e desempacotar as mensagens OSC que chegavam da rede; **re**, que permitiu o uso da técnica de

expressões regulares, responsável por averiguar se as requisições de desconexão correspondiam a alguma coisa musical presente na rede; e **socket**, também utilizado para receber e enviar dados.

Do ponto de vista funcional, a CLI apresenta informações sobre o endereço IP e número de identificação de cada coisa musical, bem como seus dados de entrada e a saída. Como mostra a Figura 14, ela é composta por quatro comandos: -i, capaz de listar individualmente as portas e dados de entrada das coisas musicais; -o, responsável por exibir as portas e os parâmetros dos dados de saída; -c, encarregado de conectar o par IP:porta, de forma que elimina a necessidade de fazer isso no próprio dispositivo; e finalmente, o comando -d, que desfaz as conexões.

O número de identificação, as portas, o endereço IP e demais dados que o administrador precisa para conectar os dispositivos, são fornecidos no momento em que eles aparecem para a rede. Quando estes mesmos dispositivos são desconectados, suas informações são removidas das listas.

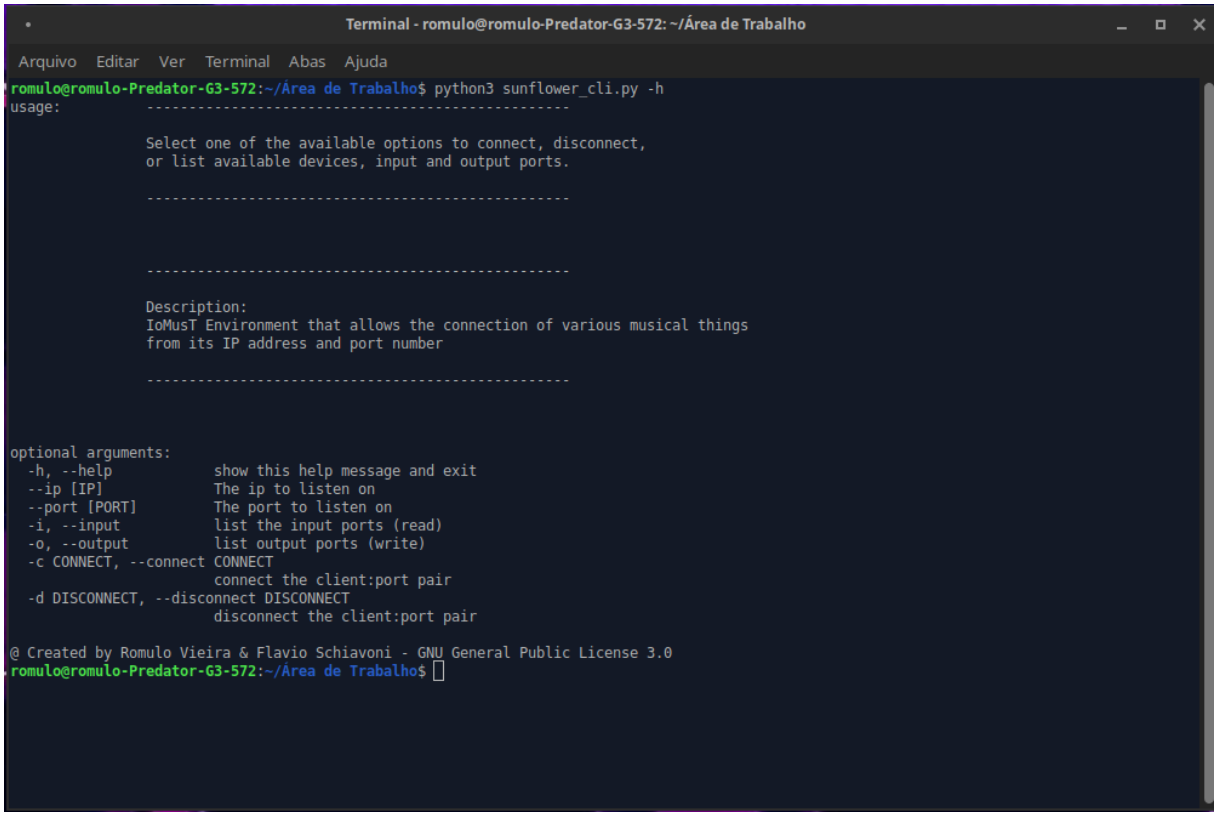


Figura 14 – CLI do Sunflower e suas principais funcionalidades.
Fonte: O Autor

A CLI é executada diretamente em um terminal. Como ela também atua como um servidor OSC para receber as mensagens oriundas das coisas musicais, caso não seja inserido nenhum comando, ela exibe uma lista tipificando o objeto (com número de identificação, endereço IP e nome), uma segunda com suas configurações de entrada, e uma terceira lista com informações sobre as saídas das coisas musicais. É importante dizer que a CLI deve ser executada antes dos aparelhos se conectarem a ela. Caso contrário, mesmo que estejam corretamente conectados a

rede, suas informações não serão exibidas, o que pode causar problemas no gerenciamento e na funcionalidade do sistema.

Para fins de organização, recomenda-se que sejam abertos quatro terminais. Um para receber as informações gerais; outro com o comando "-i", para receber informações de entrada; um com o comando "-o", para os dados de saída; e um último para conectar ou desconectar as coisas musicais. Dessa forma, garante-se que os servidores sempre estarão abertos e atualizados para receber informações das coisas musicais.

Esta interface de comando só aceita uma conexão ou desconexão por vez, onde elas são feitas seguindo o padrão IP:porta (ex: 224.0.0.0:3000). Isso evita problemas ao tentar utilizar um endereço que já está ocupado.

A quarta e última camada fica por conta do gerenciamento do sistema. Por conseguinte, ela contém informações das funcionalidades que o administrador pode utilizar para realizar seu trabalho e consegue receber mensagens nos formatos explicitados na Seção 4.4. A Figura 15 ilustra seu funcionamento prático.

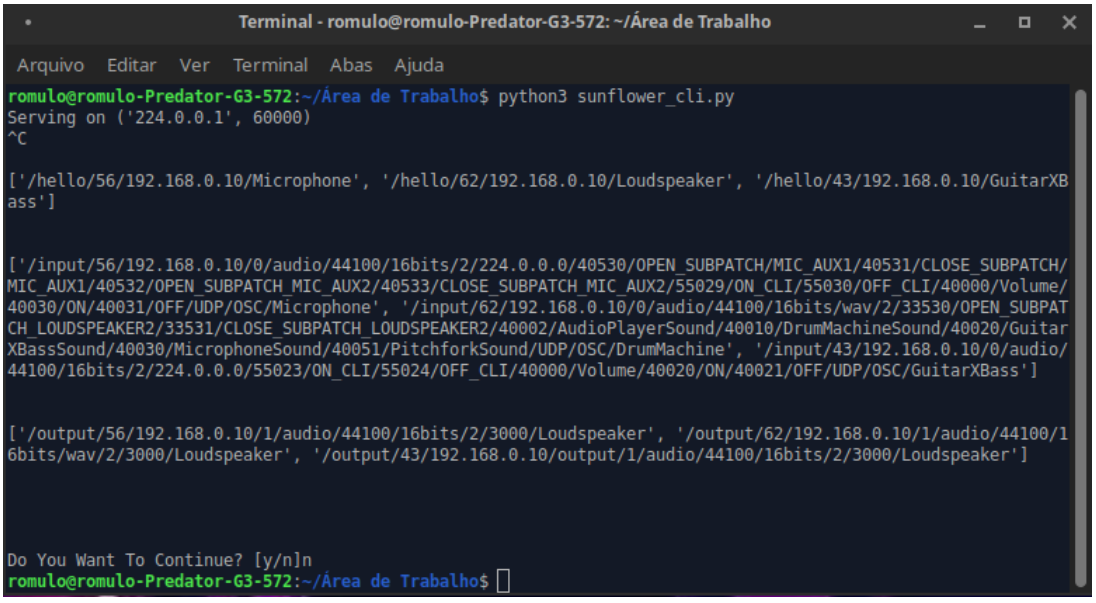


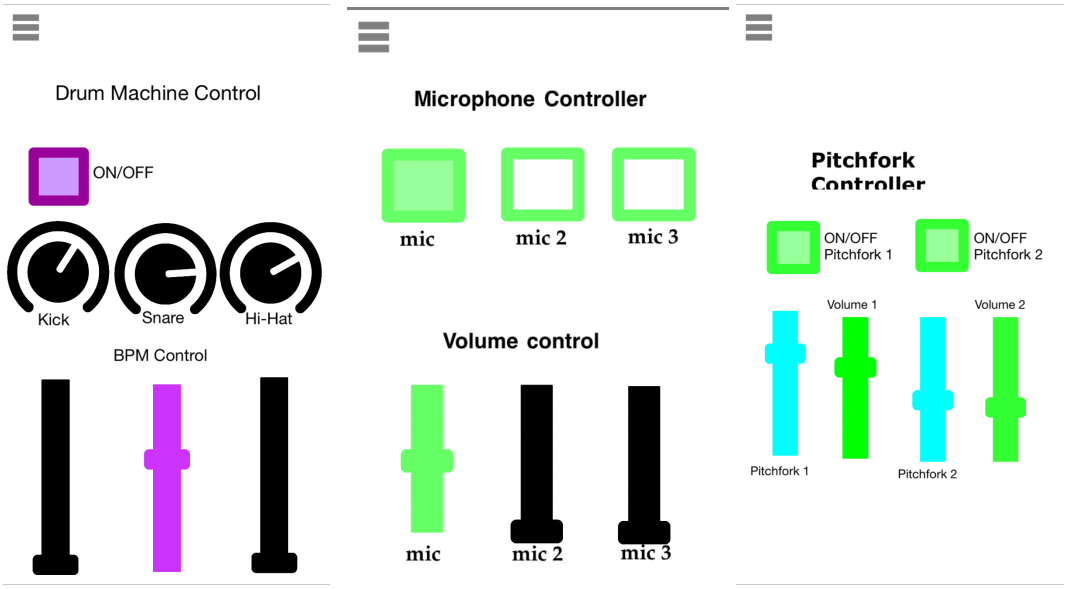
Figura 15 – CLI em operação, contendo mensagens que serão interpretadas e utilizadas pelo administrador para conectar ou desconectar determinadas coisas musicais.
Fonte: O Autor

5.4 Utilizando o MobMuPlat

Finalmente, três coisas musicais (*drum machine*, microfone e diapasão) foram transpostas para *smartphones*, para serem executadas no aplicativo MobMuPlat[3], em sua versão 1.84. Ele é um aplicativo *standalone* para iOS e Android, capaz de hospedar e executar códigos do Pure Data. Pode ainda fazer síntese sonora, receber dados MIDI e OSC via rede, consultar

[3] <https://danieliglesia.com/mobmuplat/>

e definir características de *hardware*, exibir imagens e gráficos vetoriais, receber entrada de *joystick*/*gamepad* e muito mais. Desse modo, os usuários ganham uma nova plataforma para interagir com o ambiente, ao passo que o Sunflower passa a aceitar um dispositivo de uso fácil e geral, permitindo a integração de mais pessoas. Essas novas ferramentas são vistas na Figura 16.



(a) *Drum Machine* **(b) Microfone** **(c) Diapasão**

Figura 16 – Elementos criados para o MobMuPlat.
Fonte: O Autor

5.5 Testes práticos

Antes de entrar na parte dos testes propriamente ditos, é importante fazer uma breve constatação sobre o funcionamento do Pure Data e como isso gera latência ao sistema. O áudio no Pd é processado em blocos de amostras. Esses blocos nada mais são que um grupo de *bits* que transmitem alguma informação. Neste caso, informações musicais que podem atingir os tamanhos de 64, 128, 256, 512, 1024 e 2048 amostras por bloco. Uma vez formados e enviados, eles passam por todo o processo de serem copiados em espaço de *kernel* e retornados ao espaço de usuário, em um tempo de processamento que depende diretamente do tamanho desses agrupamentos de informações. Quando tal processamento ocorre na mesma placa de som, o tempo gasto será sempre o mesmo, mas quando ocorre em duas ou mais máquinas, ele será dependente do tamanho do bloco que estas máquinas podem aceitar. Este tempo de processamento é exatamente a latência do Pure Data. As amostras de áudio utilizadas no ambiente aqui apresentado têm taxa de amostragem de 44100 Hz, culminando em um atraso de 1,45 ms.

Observe que este comportamento é normal em sistemas de áudio e que não está sendo analisado o tempo de processamento da rede, somente o processamento dos blocos que carregam informações. Possíveis soluções para sanar este problema envolvem um relógio global, que sincroniza os cristais da placa de som, fazendo com que elas enviem informações simultaneamente,

ou unidades de processamento local, pois ao colocar mais informação de processamento dentro de um único bloco, garante-se que eles serão executados no mesmo instante. Mas como estes tópicos fogem do escopo do trabalho, eles não serão debatidos em maiores detalhes.

5.5.1 O ambiente de testes

Tratando-se do desempenho da rede e entrando na seara dos testes, a latência e o *jitter* são indicadores de grande importância. Isso acontece porque são os parâmetros que indicam o tempo gasto para as informações se deslocarem de um ponto A até um ponto B. Como o Sunflower é um sistema de troca de informações em tempo real, estes valores ganham ainda mais relevância.

O primeiro elemento pode ser calculado como a diferença média entre o tempo relativo do som e o tempo esperado, somado ao tempo de latência mínima do Pure Data, que como foi mostrado, é de 1,45 ms. A equação 5.1 exibe esta definição em termos matemáticos (SCHIAVONI; QUEIROZ; WANDERLEY, 2013).

$$latencia(\Delta t) = \frac{1}{n}\sum_{i=1}^{n}(t(i) - esperado_t(i) + min_t) \tag{5.1}$$

Quanto ao segundo elemento, ele foi medido a partir do desvio-padrão da latência, como exibido na equação 5.2. Para um melhor desempenho, o ideal é que ele esteja abaixo de 20 ms. Caso exceda 30 ms, o impacto passa a ser perceptível pelos usuários (SCHIAVONI; QUEIROZ; WANDERLEY, 2013).

$$jitter = \frac{1}{n}\sum_{i=1}^{n}|t(i) - \Delta t| \tag{5.2}$$

Para a medição de valores foi utilizado o *software* Wireshark, capaz de analisar todo o tráfego da rede e organizá-lo conforme as especificações de cada máquina envolvida na comunicação. A escolha desta ferramenta se deu pelo fato de ser grátis, segura, e apresentar uma interface intuitiva para análise de dados.

Os testes foram realizados em um *laptop* Acer Predator Helios 300 (computador A) e um *laptop* Dell Inspiron 14 3442 3000 series (computador B). O computador A utilizou sistema operacional Linux Ubuntu Studio 20.04, enquanto o computador B fez uso de um Linux Ubuntu Studio 20.04 LTS. Tal sistema operacional foi escolhido por sua integração fácil e direta com sistemas de áudio. Também foi utilizado um microfone condensador Behringer C1, um violão eletroacústico de 6 cordas e uma interface de áudio Behringer U-Phoria UMC202HD, que permitiu a conexão destes dispositivos no computador.

Vale destacar que, em um primeiro momento, os valores da latência foram obtidos de forma individual, ou seja, cada dispositivo presente no ambiente teve seu valor medido e

registrado. Posteriormente, foram aplicadas as equações 5.1 e 5.2 para encontrar a latência e o *jitter* geral de cada meio de transmissão. Quanto a estes meios, foram utilizados três deles: *localhost*, cabo de par trançado e Wi-Fi.

5.5.2 Testes em Localhost

No desenvolvimento de sistemas e aplicativos, é importante descobrir se eles realmente funcionam como esperado após acessarem à internet. Para tal, o *localhost* é muito utilizado. Este termo se refere ao computador onde o programa está sendo executado, funcionando de forma análoga a um servidor virtual. Ele obrigatoriamente usa o endereço IP, na versão 4, de número 127.0.0.1 e recebe uma máscara de sub-rede no endereço 255.0.0.1. Desse modo, se qualquer roteador, *switch* ou *gateway* público receber pacotes de dados com o *localhost* como destino, as informações não serão registradas. Isso faz com que seja possível estimular uma conexão, ao passo que também evita desvio da rede, com a conexão permanecendo completamente dentro do próprio sistema.

Para acessá-lo nos *patches* do Pure Data, independentemente da camada, basta indicar a palavra reservada “localhost"ou o IP padrão na mensagem que inicia a conexão. Ainda é possível estabelecer esta interligação utilizando o *hostname* ou o IP de rede local da máquina em questão, mas estas formas não são recomendas por motivos de segurança e de otimização da rede.

Ao executar os testes dessa maneira, o computador atua como *loopback*, o que indica que o processamento começa e termina na mesma máquina. Além disso, o relógio que registra a data e hora dos pacotes será o mesmo tanto para quem envia quanto para quem recebe os dados. A latência observada é a mesma do sistema, independentemente do atraso de rede. Isso pode ser entendido como o tempo que se leva para empacotar o dado de áudio, fragmentá-lo, copiá-lo para o espaço de *kernel*, copiar de voltar para o espaço de usuário e finalmente desempacotar o dado.

A Tabela 1 exibe os tempos que cada coisa musical levou em todo este processo. Por conta de nenhum outro meio conseguir executar esta tarefa de maneira mais veloz, os tempos obtidos nos testes em *localhost* passaram a ser considerados os tempos ideais nos outros experimentos.

5.5.3 Testes em cabo de Par trançado

Os testes em cabo de par trançado aconteceram utilizando um modelo sem blindagem e com um conector RJ-45, macho para os segmentos de par trançado e fêmea para as placas de rede. Para que às duas máquinas se comuniquem, foi preciso criar uma rede local e atribuir um endereço IP para ambas. Por não ser o foco deste trabalho e por esse procedimento diferir em cada sistema, mais detalhes não serão oferecidos. Os valores medidos utilizando este método são expostos na Tabela 2.

Tabela 1 – Medições em *localhost*.

Patch	**Tempo**
Player de áudio	0,012 ms
Controle do *Player* de áudio - Abrir arquivo	0,049 ms
Controle do *Player* de áudio - Tocar/Pausar	0,001 ms
Controle do *Player* de áudio - Recomeçar trilha	0,031 ms
Controle do *Player* de áudio - Abrir *subpatch*	0,003 ms
Controle do *Player* de áudio - Ligar trilha	0,016 ms
Drum Machine	0,013 ms
Controle do *Drum Machine* - BPM	0,109 ms
Controle do *Drum Machine* - Volume	0,057 ms
Controle do *Drum Machine* - Ligar/Desligar	0,027 ms
Controle do *Drum Machine* - Padrão	0,068 ms
Guitarra/Baixo	0,182 ms
Controle da Guitarra/Baixo - Ligar/Desligar	0,012 ms
Alto-falante	Não se aplica (0 ms)
Controle do Alto-falante - Abrir/Fechar	0,003 ms
Microfone	0,041 ms
Controle do Microfone	0,001 ms
Diapasão	0,045 ms
Controle do Diapasão - Frequência	0,105 ms
Controle do Diapasão - MIDI	0,031 ms
Controle do Diapasão - Ligar/Desligar	0,014 ms
Gravador	Não se aplica (0 ms)
Controle do Gravador - Ligar/Desligar	0,066 ms
Emissor de Vídeo	0,040 ms
Receptor de Vídeo	Não se aplica (0 ms)
Controle de Vídeo - Começar/Parar	0,028 ms
Controle de Volume	0,457 ms

Fonte: O Autor.

5.5.4 Testes no Wi-Fi

Finalmente, para verificar o desempenho do Sunflower em uma rede sem fio, foi utilizado um Wi-Fi no padrão IEEE 802.11n, com um limite de transmissão teórico de 100 a 600 Mbps, utilizando uma internet com largura de banda de até 100 Mbps e um roteador TP-Link AC 1200 Archer C5. Os valores obtidos a partir disso são mostrados na Tabela 3.

Tabela 2 – Medições no cabo de par trançado.

Patch	**Tempo esperado**	**Tempo real (latência)**
Player de áudio	0,012 ms	0,896 ms
Controle do *Player* de áudio - Abrir arquivo	0,049 ms	2,724 ms
Controle do *Player* de áudio - Tocar/Pausar	0,001 ms	2,572 ms
Controle do *Player* de áudio - Recomeçar trilha	0,031 ms	1,805 ms
Controle do *Player* de áudio - Abrir *subpatch*	0,003 ms	0,052 ms
Controle do *Player* de áudio - Ligar trilha	0,016 ms	0,135 ms
Drum Machine	0,013 ms	0,613 ms
Controle do *Drum Machine* - BPM	0,109 ms	0,028 ms
Controle do *Drum Machine* - Volume	0,057 ms	0,166 ms
Controle do *Drum Machine* - Ligar/Desligar	0,027 ms	1,488 ms
Controle do *Drum Machine* - Padrão	0,068 ms	0,130 ms
Guitarra/Baixo	0,182 ms	1,228 ms
Controle da Guitarra/Baixo - Ligar/Desligar	0,012 ms	0,499 ms
Alto-falante	Não se aplica (0 ms)	Não se aplica (0 ms)
Controle do Alto-falante - Abrir/Fechar	0,003 ms	0,061 ms
Microfone	0,041 ms	1,120 ms
Controle do Microfone	0,001 ms	0,243 ms
Diapasão	0,045 ms	0,074 ms
Controle do Diapasão - Frequência	0,105 ms	0,199 ms
Controle do Diapasão - MIDI	0,031 ms	0,104 ms
Controle do Diapasão - Ligar/Desligar	0,014 ms	0,539 ms
Gravador	Não se aplica (0 ms)	Não se aplica (0 ms)
Controle do Gravador - Ligar/Desligar	0,066 ms	0,743 ms
Emissor de vídeo	0,040 ms	0,605 ms
Receptor de vídeo	Não se aplica (0 ms)	Não se aplica (0 ms)
Controle de vídeo - Começar/Parar	0,028 ms	0,921 ms
Controle de Volume	0,457 ms	1,556 ms

Fonte: O Autor.

Tabela 3 – Medições no Wi-Fi.

Patch	**Tempo esperado**	**Tempo real (latência)**
Player de áudio	0,012 ms	4,300 ms
Controle do *Player* de áudio - Abrir arquivo	0,049 ms	1,342 ms
Controle do *Player* de áudio - Tocar/Pausar	0,001 ms	1,877 ms
Controle do *Player* de áudio - Recomeçar trilha	0,031 ms	4,910 ms
Controle do *Player* de áudio - Abrir *subpatch*	0,003 ms	2,899 ms
Controle do *Player* de áudio - Ligar trilha	0,016 ms	1,964 ms
Drum Machine	0,013 ms	4,208 ms
Controle do *Drum Machine* - BPM	0,109 ms	1,541 ms
Controle do *Drum Machine* - Volume	0,057 ms	2,506 ms
Controle do *Drum Machine* - Ligar/Desligar	0,027 ms	2,157 ms
Controle do *Drum Machine* - Padrão	0,068 ms	2,656 ms
Guitarra/Baixo	0,182 ms	4,832 ms
Controle da Guitarra/Baixo - Ligar/Desligar	0,012 ms	6,153 ms
Alto-falante	Não se aplica (0 ms)	Não se aplica (0 ms)
Controle do Alto-falante - Abrir/Fechar	0,003 ms	3,368 ms
Microfone	0,041 ms	4,542 ms
Controle do Microfone	0,001 ms	2,669 ms
Diapasão	0,045 ms	4,795 ms
Controle do Diapasão - Frequência	0,105 ms	1,902 ms
Controle do Diapasão - MIDI	0,031 ms	0,977 ms
Controle do Diapasão - Ligar/Desligar	0,014 ms	3,379 ms
Gravador	Não se aplica (0 ms)	Não se aplica (0 ms)
Controle do Gravador - Ligar/Desligar	0,066 ms	6,313 ms
Emissor de vídeo	0,040 ms	3,062 ms
Receptor de vídeo	Não se aplica (0 ms)	Não se aplica (0 ms)
Controle de vídeo - Começar/Parar	0,028 ms	6,757 ms
Controle de Volume	0,457 ms	3,385 ms

Fonte: O Autor.

6 DISCUSSÃO E RESULTADOS

Este Capítulo traz considerações sobre a implementação do Sunflower. Seu uso prático serviu como prova para o conceito de funcionamento inicialmente planejado e também ajudou na detecção de possíveis falhas e aspectos que devem ser melhorados no desenvolvimento da ferramenta. O Capítulo também é responsável por abordar os resultados matemáticos e analisar se os objetivos pretendidos foram alcançados, além de propor uma análise comparativa entre o ambiente aqui desenvolvido e outros propostos na área de IoMusT (apresentados no Capítulo 4), de modo a identificar os pontos de contato e as particularidades de cada um.

6.1 Resultados dos testes

Desde a programação do ambiente e testes primários realizados em *localhost*, observou-se que o computador A apresentava melhor desempenho. Isso confirmou o fato já esperado que a configuração física da máquina (*hardwares*) influencia diretamente na transmissão de áudio em rede.

No que diz respeito a latência e ao *jitter*, elas foram iguais as do próprio sistema, independentemente do atraso de rede, por motivos já debatidos neste texto, mas em termos práticos, o áudio saiu quase de forma instantânea no alto-falante. Já os dados de vídeo, por consumirem mais recursos do computador e da rede, foram um pouco mais lentos. Os dados de controle OSC, que por serem pequenos em quantidades de *bytes* e enviados de forma espaçada, também foram praticamente instantâneos.

Utilizando o cabo de par trançado, pode-se observar uma latência geral, ou seja, englobando os valores das camadas de áudio, vídeo e controle, na casa de 2,083 ms, enquanto o *jitter* foi de 0,761 ms. Para o Wi-Fi, a latência foi de 4,453 ms e o *jitter* de 1,843 ms, confirmando que a conexão cabeada é melhor que aquela sem fio. Mas ainda assim, os valores ficaram nos limites considerados ideais e não comprometerem a troca de dados nem a prática musical.

Um ponto importante a ser destacado nos testes, em especial aqueles sem fio, é que ao aumentar a quantidade de dispositivos na rede, cresce também a sua latência. Ainda nesse sentido, destaca-se que os testes apresentam limites de usuários de acordo com seus meios de transmissão pela rede. A implementação em um cenário minimizado serviu para balizar esta comunicação e exibir os conceitos do Sunflower aplicados na prática, mas caso ela ocorresse em um cenário real, a infraestrutura de rede precisaria ser melhorada.

Quanto a percepção do som, o ouvido humano é capaz de detectar dois sons simultâneos desde que eles não estejam separados por mais de 20 ms, o que também significa que em uma performance bilateral, este limiar não deve superar os 40 ms de atraso (BARBOSA; CARDOSO, 2021). Destaca-se ainda que estes tempos são variáveis, pois dependem de algumas características do som (timbre e altura, por exemplo), do estilo da música e de outras categorias de *feedback*,

como estímulos visuais ou físicos. A latência de 40 ms é abrangente o suficiente para ser considerado no cálculo. Apesar de neste caso a latência estar abaixo dos valores que causam perdas perceptíveis, ela é variável e imprevisível, gerando erros de base de tempo, perda de sequência e até mesmo de conteúdo. Uma abordagem para lidar com este problema é aceitar esta condição como um elemento natural na criação de música em rede.

6.2 Sobre as características desejáveis

Das características inicialmente desejadas, todas foram alcançadas. O sistema mostrou-se heterogêneo ao comportar diversas coisas musicais, incluindo dispositivos capazes de gerarem e consumirem informações gráficas, além de apresentar a circulação de uma grande variedade de dados.

A interface de gerenciamento, apesar de simples e contar somente com informação textual, consegue exibir as principais características de cada coisa musical, assim como identificá-las e reportar os respectivos estados de conexão, fazendo com que elas mostrem ao sistema todas as suas configurações de rede. A mesma exibe informações sobre as entradas e saídas, cumprindo outro requisito inicialmente pretendido.

A integração com *softwares* e *hardwares* legados é possível graças a capacidade do sistema de encapsular informações MIDI no protocolo OSC enviá-las pela rede.

O funcionamento similar à arquitetura Pipes-and-Filters também ficou evidente, já que as coisas musicais não tinham conhecimento prévio dos outros elementos conectados à rede, cabendo-lhes somente o processamento de dados e respectivo envio para a saída. A categorização conforme o funcionamento também pôde ser observada, já que alguns filtros apenas emitiam informações, como o microfone, outros somente as recebiam, como o alto-falante, e alguns desempenhavam ambas as funções, como o gravador ou até mesmo a *drum machine*.

O cumprimento de todos esses requisitos abre caminho para que o sistema apresente uso lúdico e intuitivo, permitindo que múltiplos usuários, com objetivos e níveis de habilidade diferentes, possam usá-los sem maiores problemas.

Dos pontos passíveis de melhoria está a tela de gerenciamento, que pode evoluir para uma interface gráfica, mais intuitiva. Uma possibilidade que não foi aventada para este trabalho, mas que seria de grande valia, é pensar nos filtros conversores como *hubs* de áudio, vídeo e de mensagens OSC. Dessa forma, eles atuarão como intermediários entre outros elementos que não são configuráveis, mas que querem se comunicar, recebendo diferentes dados e adequando-os para as configurações aceitas por ambos. Assim, eles conseguem conectar elementos com diferentes intervalos ou espaços de nome em mensagens OSC, além daqueles com distintos formatos, taxas de amostragem e profundidade de *bits* para amostras de áudio.

Outro ponto que pode ser melhorado é o comportamento das camadas, que pode ser

pensado como um barramento de *software*, capaz de reunir todas as comunicações em um único canal virtual compartilhado, garantindo que os processos conectados a este barramento possam comunicar entre si. Este modo de pensar o sistema simplifica a funcionalidade existente nas camadas e facilita o compartilhamento de informações, além de garantir modularidade a elas e separação de privilégios.

Com o advento da Internet das Coisas Musicais, surgiu também uma infinidade de dispositivos, com os mais diversos recursos de interação e conectividade, permitindo que usuários e artistas presentes nestes sistemas extrapolassem estas capacidades. Consequentemente, isso ocasionou o aumento na complexidade e probabilidade de erros nesses equipamentos, especialmente por conta da incompatibilidade existente entre eles. Este *design*, portanto, apresenta a vantagem de facilitar a comunicação entre diferentes entidades, além de promover o fluxo criativo e permitir que a rede seja escalável. Sua capacidade de enviar e/ou acessar os dados como entidades completas garante maior flexibilidade da arquitetura, uma vez que os dados podem mudar com o passar do tempo. A reusabilidade e legibilidade por parte dos humanos também é uma característica presente neste protocolo, tornando-o fácil de aprender, especialmente por quem já apresenta algum conhecimento em linguagens de programação e sistemas computacionais. Por fim, a simplicidade das estruturas dos dados garante a dinamicidade da rede, permitindo que ela seja rearranjada de inúmeras formas.

Tópicos sobre segurança da rede e dos dispositivos não foram abordados neste trabalho, uma vez que estas configurações podem atrapalhar o desempenho musical ou impedir que determinados *softwares* sejam executados. Aspectos hápticos também não foram contemplados, dada a estrutura dos dispositivos utilizados. Já a concorrência de acesso é resolvida de forma simples, onde o primeiro elemento que solicitou a conexão tem preferência. Na hipótese da coisa musical que irá receber esta conexão conseguir lidar com mais de um fluxo de uma única vez, eles podem coexistir.

Destaca-se, por fim, que apesar do protocolo proposto estar presente na camada de aplicação do modelo TCP/IP, ele também pode ser implementado nos demais níveis, tornando o sistema semelhante a uma rede de malha, onde os dispositivos utilizam apenas um ponto de acesso para se comunicarem. Isso muda as perspectivas sobre um empecilho em sistemas reais, que é o uso da largura de banda. Ainda assim, a ideia de funcionamento análogo à arquitetura Pipes-and-Filters é mantida, predizendo a existência de elementos vertedouros, sumidouros e híbridos, além da necessidade de conhecimento sobre as portas de entrada e saída, assegurando que as cinco etapas do protocolo sejam atendidas.

Como o Sunflower (logo na Figura 17) é uma proposta para ambientes IoMusT que permite que diferentes tipos de dados coexistam no sistema e que as coisas musicais sejam interoperáveis e configuradas de forma remota, ele não deve ser tratado como algo monolítico e sim como uma solução aberta, capaz de receber contribuições de outros usuários e desenvolvedores, bem como ser incorporada em seus próprios projetos. Alguns aspectos não são

abordados neste trabalho, como uma trativa de exceções caso alguma coisa musical, parâmetro de rede ou configuração de protocolo não esteja no padrão esperado, e ausência de propostas de uso de protocolos tradicionais para a comunicação em rede, focando em apresentar novas funcionalidades e maneiras de se pensar este ambiente. Não houve também um debate sobre os problemas de padronização na área de Internet das Coisas e como esses desafios refletem no Sunflower. Por conta do trabalho aqui desenvolvido focar no aspecto artístico desse tipo de comunicação, ele se limitou a debater somente as instigações presentes na área de Internet das Coisas Musicais, sob a óptica proposta pelos trabalhos de Turchet e colaboradores (TURCHET et al., 2018).

Um vídeo onde o protocolo é utilizado para viabilizar uma performance artística[1] e outro demonstrando como configurá-lo[2] estão disponíveis em seu canal oficial no YouTube.

Figura 17 – Logo do Sunflower, uma proposta de *design* para ambientes de Internet das Coisas Musicais.

Fonte: O Autor

6.3 Reflexões sobre os protótipos desenvolvidos

Apesar de os motores de áudio e vídeo terem sido elaborados no Pure Data e no Processing, este ambiente não deve se limitar a tais ferramentas e deve conseguir integrar dispositivos

[1] <https://www.youtube.com/watch?v=bWfVCGAda_c&ab_channel=SunflowerIoMusT>
[2] <https://www.youtube.com/watch?v=lGW5eobrYwc&ab_channel=SunflowerIoMusT>

feitos a partir das mais diversas técnicas e conceitos. As possibilidades que surgem a partir disso exercem um contraponto aos exemplares de coisas musicais que aparecem ao longo deste texto, como a Sensus Smart Guitar e o Smart Mandolin, que apesar de serem inovadores e únicos, acabam por utilizar técnicas específicas, e em muita das vezes, tecnologias não acessíveis para o público geral. A partir disso, é importante pensar em instrumentos, equipamentos de áudio e afins que sigam um protocolo padrão, permitam conexão sem fio à rede e abram diversas possibilidades para interação e criação artística. Este comportamento pode ser observado em artefatos que adotam o protocolo Bluetooth, por exemplo, o que possibilita a troca de dados entre dispositivos heterogêneos, diferentemente dos antigos aparelhos de áudio que são monolíticos e pouco adaptáveis aos demais elementos presentes no ambiente. Estas questões foram consideradas no desenvolvimento dos protótipos. Por último, eles se mostraram escaláveis e mutáveis. Toda essa discussão, resultou na seguinte classificação dos elementos criados:

- **Instrumentos musicais:** consiste nos aparelhos que simulam instrumentos tradicionais, sejam eles acústicos, elétricos, de sopro, percussão ou digitais. O exemplo mais proeminente para esta categoria é o protótipo da *drum machine*;
- **Equipamentos de áudio:** formada por equipamentos que convertem ondas sonoras acústicas em sinais elétricos e/ou que apresentam capacidade de amplificá-los, modificá-los ou armazená-los em algum nível. Ex: microfone, alto-falante, gravador, *patch* que envia áudio do violão para a rede e reprodutor de áudio;
- **Equipamentos musicais simbólicos:** se anteriormente foram apresentados equipamentos responsáveis pela reprodução e armazenamento de áudio, aqui são classificados aqueles com capacidade de manipular informações musicais simbólicas. Estas ferramentas não produzem som por si mesmas, mas conseguem manipulá-lo digitalmente. O exemplo é o diapasão que opera sob o protocolo MIDI;
- **Unidades de efeitos:** inclui elementos capazes de alterar o som original das fontes de áudio, como o controlador de volume e de reverberação;
- **Ferramentas de auxílio à performance artística:** são utensílios que não produzem nem alteram o som, mas que são de grande valia para um ecossistema musical/artístico. Eles são: gerador e reprodutor de vídeo e demais elementos gráficos.

Além dessa categorização básica, o Sunflower também permite ambientes individuais. Isso quer dizer que cada artista pode ter a sua própria ilha de criação e os seus próprios dispositivos autônomos, desde que seja respeitado o *design* proposto. Isso é facilitado graças ao protocolo de performance. Assim, é possível que ele interaja apenas com o administrador, não permitindo que ninguém altere e manipule seus dados. Esse modo de funcionamento é inerente a um sistema distribuído.

Ao fim de tudo, o sistema tem modo de operação similar àquele observado na arquitetura publicador/assinante (*publish/subscriber*), onde os dispositivos publicam suas características e funcionalidades na rede e somente aqueles com interesse e capacidade de se conectar podem fazê-lo. Nesse sentido, o envio de mensagens é *multicast*, justamente para garantir essas características e evitar redundâncias e aumento de tráfego na rede.

6.4 Comparações com os trabalhos relacionados

A partir dos três modelos apresentados no Capítulo 3 e de todas as informações a respeito do Sunflower, verificou-se que a maioria deles recorre ao Pure Data como motor de áudio e aos protocolos OSC e UDP para envio de informações e para permitir a interoperabilidade entre dispositivos. Tratando especificamente do primeiro modelo, seus clientes também podem ser classificados como publicador/assinante, uma vez que os dados são repassados ao servidor através de mensagens OSC que serão mapeadas para as solicitações que podem lidar com os recursos contidos nessas mensagens. Outras semelhanças deste modelo com o Sunflower são: uso de *script* em Python para transmitir informações, proposta de arquitetura para o ambiente e separação entre o modelo lógico dos dados do modelo de implementação do sistema.

No segundo modelo apresentado, é possível observar uma prevalência de escolha da arquitetura cliente-servidor e uma divisão das coisas musicais de acordo com seu modo de funcionamento, assim como no Sunflower. A terceira proposta de ambiente, por ser a mais recente, não se preocupa ainda com protocolos de rede nem com protocolos que transferem informação musical, fazendo com este ambiente seja o mais diferente de todos que são expostos neste trabalho.

Apesar das inúmeras similaridades, as peculiaridades de cada um também se fazem claras. O sistema aventado neste trabalho tem um modo de funcionamento inspirado no Pipes-and-Filters, é dividido em camadas e mostra uma preocupação maior com as características do som, como taxa de amostragem e profundidade de *bits*, além de permitir a integração com ferramentas legados utilizando o protocolo MIDI.

O modelo 1 permite a integração com objetos reais, enquanto o 2 estabelece uma relação entre IoMusT e Audio Commons, ao passo que o 3 não somente indica um ambiente baseado em redes móveis como já vislumbra o uso da tecnologia 5G na prática musical.

Esta breve análise está sintetizada na Tabela 4 e consegue resumir algumas características e ferramentas que estão se consolidando em ambientes de Internet das Coisas Musicais, ao mesmo tempo que permite que cada rede tenha sua própria estrutura e objetivos artísticos, com múltiplas especificidades, permitindo sempre a adesão de novas tecnologias. Por conta disso, cada cenário tem uma combinação única e inovadora, fornecendo as bases para transformar as experiências musicais.

Tabela 4 – Síntese das arquiteturas de Internet das Coisas Musicais.

Característica	Modelo 1	Modelo 2	Modelo 3	Sunflower
Motor de áudio	Pure Data	Pure Data	Elk Audio OS	Pure Data
Protocolo de rede	UDP	UDP	Não especificado	UDP
Protocolo de troca de informação musical	OSC	OSC	Não especificado	OSC e MIDI
Tipo de dados sonoros	WAV	WAV e MP3	MP3	PCM
Tipo de dados de vídeo	Não se aplica	Não se aplica	Não se aplica	AVI, QuickTime, MPEG

Fonte: O Autor.

6.5 Possíveis cenários de uso

Após estas elucidações acerca do funcionamento do Sunflower, é possível imaginar alguns cenários de uso para este sistema, onde ele irá corroborar de forma ativa com o processo criativo, como uma *jam session* que combina instrumentos tradicionais e coisas musicais, gravações em um estúdio inteligente e até mesmo apresentações ao vivo. Isso proporciona novas maneiras de interação entre os diferentes elementos artísticos que estão presentes neste ambiente, além de permitir novas formas de composição, aprendizado e gravação de música.

A presente Seção, portanto, detalha estes cenários hipotéticos. Eles não visam representar as necessidades ou desejos reais dos usuários, pois este tópico ainda precisa ser investigado, mas sim proporcionar uma discussão sobre o potencial do Sunflower em contextos artísticos.

6.5.1 Cenário 1: Uma *jam session* que combina elementos musicais tradicionais e coisas musicais

Nesta conjuntura, tanto os instrumentos eletroacústicos quanto as coisas musicais podem trocar dados pela rede. Eles podem estar plugados em alto-falantes ou em *patches* conectados à internet, enquanto musicistas e usuários manipulam suas propriedades e geram sinais sonoros a partir de sistemas computacionais.

A camada gráfica pode se beneficiar de vídeos oriundos das câmeras de *smartphones* ou *laptops*, que serão posteriormente enviados para SmarTVs ou telas que vão exibi-los para os demais participantes. Ainda nesta camada, imagens e animações podem ser usadas para garantir uma maior experiência audiovisual aos envolvidos. O controle dos demais dispositivos, como disparar uma sequência de notas em um sintetizador ou alterar o tempo de uma bateria também pode ser realizado através de um equipamento de vídeo, que também atuarão como equipamentos gerenciadores.

Outros usuários podem participar exercendo controle sobre volume, gravação e efeitos nos instrumentos, inclusive se não estiverem no mesmo local onde acontece esta sessão, ou utilizando o MobMuPlat em seus *smartphones*. Devem ser capazes ainda de alterar os padrões de cores das imagens, resolução, e formato dos vídeos e das artes em Processing.

A camada de gerenciamento fica a cargo de um engenheiro/técnico de som, responsável

por manipular as conexões via interface de comando, permitindo ou proibindo que certos instrumentos e/ou usuários se comuniquem, além de escolher quais instrumentos e trilhas de áudio serão enviados para o público geral.

As coisas musicais podem capturar os movimentos dos usuários e usá-los para controlar certos parâmetros e gerar som. Ainda, devem ter potencial de receber informações e estímulos dos outros elementos presentes na rede

6.5.2 Cenário 2: Gravação em estúdio inteligente

Um segundo cenário de uso que se vislumbra é um estúdio inteligente que utiliza conceitos de IoMusT para gravar artistas solos, duplas, bandas e até mesmo orquestras com os mais variados instrumentos. Para tal, a interface de gravação deve adequar sua quantidade de canais conforme cada caso. Além disso, deve conseguir armazenar configurações e efeitos da preferência de cada musicista. O processo de gravação, no que lhe concerne, deve ser possível mesmo se as pessoas envolvidas não estiverem no mesmo local físico.

A camada gráfica pode fornecer informações técnicas da rede, indicando quais objetos estão conectados, ao passo que também corrobora com a gravação, exibindo a letra da música, partitura e outras informações que auxiliem esta etapa. Os efeitos e *plugins* utilizados podem ser obtidos através da internet e serem reconfigurados e controlados de maneira remota, enquanto o técnico de som pode combinar trilhas pré-gravadas com a execução ao vivo. Outros parâmetros musicais podem ser corrigidos automaticamente por *scripts* ou sistemas de inteligência artificial. Tudo isso será realizado na camada de controle.

A camada de gerenciamento pode, novamente, utilizar a CLI para conectar os instrumentos, pedais de efeitos e interfaces de áudio, bem como controlar as informações gráficas que serão exibidas.

6.5.3 Cenário 3: Apresentações ao vivo

A exemplo dos dois cenários citados, este terceiro, que trata de apresentações ao vivo, também está repleto de possibilidades. Os musicistas podem trocar dados pela rede utilizando coisas musicais semelhantes as que foram apresentadas nesse trabalho, enquanto os membros da audiência podem corroborar com a apresentação através de aplicações executadas no MobMuPlat, criando sons ou controlando parâmetros presentes no ecossistema.

A camada gráfica pode ser usada para passar informações ao público, exibir vídeos e animações que engrandecem a narrativa que se pretende criar e, novamente, apresentar algum auxílio aos musicistas, como exibir letras, lista de músicas e partituras.

O controle dos efeitos musicais, luzes e telões podem ser feitos por membros da audiência ou técnicos especializados. Da mesma forma, esta camada pode substituir os meios tradicionais de mixagem do som por sistemas digitais passíveis de controle pela rede.

A camada de gerenciamento, a partir de sua tela de comandos por linhas, pode restringir o acesso a certas funcionalidades e lidar com permissões, sejam elas referentes à prática musical ou ao acesso à rede. Em certos momentos, esta camada se mistura com a camada de controle.

As coisas musicais podem receber estímulos sonoros e vibrarem de acordo com cada canção, da mesma forma que apresentam potencial para aplicar efeitos únicos e individuais para cada som recebido. Ainda neste sentido, alteram-se as cores em um canhão de luz e disparam-se máquinas de fumaça diante certas incitações.

Os musicistas e dançarinos devem conseguir coletar dados relativos aos sentimentos do público, alterando a sequência de músicas, coreografias, efeitos visuais e cores usadas na apresentação de modo a melhor se adequar a eles.

A audiência remota também deve estar habilitada a participar do espetáculo, recebendo imagens e áudio em tempo real, além de conseguir controlar estes parâmetros.

7 CONCLUSÕES

A área de Internet das Coisas Musicais proporciona novas perspectivas para a prática musical, mediando a interação de musicistas com seus pares ou de musicistas com membros da audiência por meio de diferentes recursos computacionais. Os concertos ao vivo, gravações em estúdio e aprendizado de música tendem a se beneficiar dessa nova tendência. Apesar dos pontos positivos, alguns problemas são evidentes, como aqueles inerentes às questões sociais, econômicas e artísticas. Do ponto de vista tecnológico, existem desafios referentes à falta de segurança e privacidade nestes ambientes, e principalmente, a dificuldade para lidar com heterogeneidade dos dispositivos. O Sunflower surge justamente para atacar esta questão, propondo um *design* com funcionamento inspirado na arquitetura Pipes-and-Filters, onde cada elemento consegue estabelecer comunicação com seus vizinhos sem a necessidade de conhecimento prévio do endereço de IP, além de não enfrentarem restrições de comunicação por conta de suas características físicas. Existe ainda a divisão em camadas, que garante ao sistema reusabilidade, fácil manutenção e expansão. Com ambos os conceitos trabalhando juntos, diferentes tipos de dados, protocolos e funcionalidades conseguem cooperar. Quanto ao protocolo de funcionamento, ele consiste em cinco etapas que contemplam às necessidades de comunicação em uma rede, além de proporcionar uma mudança funcional entre modo de configuração e modo de performance. Cabe destacar que as ferramentas aqui apresentadas cumprem bem o seu papel, mas elas não são obrigatórias na elaboração de um ambiente similar. O ponto forte do trabalho, portanto, reside nas especificações propostas em sua estrutura funcional.

O desenvolvimento desta plataforma mostrou-se uma atividade complexa, pois requer conhecimentos em diversas áreas da computação, como redes de computadores, processamento de sinais, engenharia de *software* e *design* de som, além de conceitos oriundos da música. O público-alvo deste trabalho é formado por engenheiros e técnicos de som, musicistas, professores e alunos de música, cientistas que têm interesse de pesquisa na área e membros da audiência que queiram participar de forma mais significativa em um espetáculo. O Sunflower também visa contemplar o processo de criação artístico-musical e deve ser capaz de lidar com eventuais problemas de rede.

Findadas as discussões arquiteturais, as implementações práticas da ferramenta foram abordadas. Para isso, realizaram-se três testes diferentes, um em *localhost*, outro com conexão cabeada e um último em meio sem fio, utilizando Wi-Fi. Como era de se esperar, a conexão cabeada apresentou melhores resultados que a conexão sem fio. O *hardware* das máquinas utilizadas, mesmo que de forma não decisiva, também interferiram no ambiente.

Estes testes foram realizados com o intuito de i) verificar a latência e o *jitter* na troca de informação musical pela rede e ii) aferir a aplicabilidade dos conceitos aqui propostos. Como as duas propriedades do primeiro tópico são inerentes à comunicação em rede, elas não podem ser eliminadas, apenas atenuadas. Em todos os casos, os valores não interferiram na prática

musical e ofereceram resultados que, apesar de típicos desta aplicação e desta configuração de rede, podem auxiliar outros musicistas/desenvolvedores. O segundo tópico foi alcançado por meio da publicação de recursos, da visão do ambiente e das ferramentas utilizadas, muito por conta de serem livres e multiplataforma.

A ferramenta se mostrou promissora ao atender as necessidades musicais e de rede. Mesmo assim, o autor realizou uma comparação com outros ambientes para destacar quais aspectos são recorrentes, como uso dos protocolos UDP e OSC, bem como as individualidades de cada um que podem servir de inspiração para a elaboração de outros modelos. Dessa forma, não se pretende encerrar a discussão sobre quais formatos de áudios e protocolos devem ser usados na comunicação de IoMusT, da mesma forma que o autor não reivindica para si a autoridade de dizer o que deve ou não ser feito ao planejar um ambiente nestes moldes.

O desenvolvimento deste trabalho ao longo do mestrado foi realizado em paralelo com algumas disciplinas ofertadas pelo PPGCC. Dentre as que tiveram impactos mais significativos na concepção e realização desta tarefa, estão Redes de Computadores e Computação Móvel e Ubíqua, por abordarem os conceitos e estruturas básicas envolvidas neste tipo de comunicação, além de me aproximarem da pesquisa sobre Internet das Coisas Musicais. E apesar de ter sido cursada em estágio avançado de desenvolvimento do Sunflower, os conteúdos de Tópicos Especiais em Sistemas de Informação — Computação Musical ofereceram importantes alicerces que sustentam este trabalho, além de indicarem os próximos passos a serem seguidos.

As cadeiras de Teoria da Computação e Tópicos Especiais em Sistemas de Informação — Engenharia de Software forneceram as bases para o código da CLI, principalmente o conceito de expressões regulares, utilizado para analisar os objetos presentes nas listas e posteriormente removê-los, afora a contribuição para o modo de funcionamento do Sunflower, inspirado em duas arquiteturas de *software*. A última disciplina a ser mencionada é a de Aprendizado de Máquina, que embora não tenha tido uma consequência muito significativa neste trabalho, abre caminho para pesquisas futuras que podem correlacionar estes dois assuntos.

A realização dos dois estágios em docência também foram de grande valia. O primeiro deles, em Introdução à Robótica, permitiu o trabalho com dispositivos e conceitos que se aproximam da eletrônica. Esta experiência foi útil no momento de se pensar o comportamento e a estrutura das coisas musicais. O segundo deles, na disciplina de Computação na Arquitetura, permitiu um trabalho com linguagens usadas ao longo deste texto, como Pure Data e Processing, além de proporcionar debates e discussões sobre o papel da arte na sociedade, arte interativa, espetáculos mediados por tecnologia e tanto outros conceitos que foram, em algum grau, utilizados nesta tarefa.

Além do trabalho aqui apresentado, no decorrer do mestrado tive a oportunidade de participar de 16 eventos acadêmicos, apresentando trabalhos em 12 deles, listados a seguir:

- Congresso Internacional de Arte, Ciência e Tecnologia;

- Congresso Internacional de Direito e Inteligência Artificial;
- International Conference on New Interfaces for Musical Expression;
- Simpósio Brasileiro de Games e Entretenimento Digital;
- Simpósio Brasileiro sobre Fatores Humanos em Sistemas Computacionais;
- X Brazilian Symposium on Computing Systems Engineering;
- X Ubiquitous Music Workshop;
- XI Ubiquitous Music Workshop;
- 17º Brazilian Symposium on Computer Music;
- 18º Brazilian Symposium on Computer Music;
- International Conference on Audio Mostly;
- Workshop de Computação Teórica e Aplicada.

Além disso, tive uma produção artística e cultural (III Festival de Teatro Comunitário de Mariana-MG) e participação como ouvinte em outros três eventos (Festival Internacional de Música em Casa, II Congresso Internacional Online entre Arte, Cultura e Educação, e Seminário Internacional A(r)tivismos Urbanos – (sobre)vivendo em tempos de urgências).

Das contribuições para a área e divulgação científica, estão a publicação de 14 artigos completos:

- Everyday Use of the Internet of Musical Things: Intersections with Ubiquitous Music (VIEIRA; BARTHET; SCHIAVONI, 2020);
- The things of the Internet of Musical Things: defining the difficulties to standardize the behavior of these devices (VIEIRA; GONÇALVES; SCHIAVONI, 2020));
- Sunflower: an environment for standardized communication of IoMusT (VIEIRA; SCHIAVONI, 2021b);
- Managing an IoMusT environment and devices (VIEIRA; SCHIAVONI, 2021a);
- Can Pipe-and-Filters architecture help creativity in Music? (VIEIRA; SCHIAVONI, 2020);
- Fliperama: An affordable Arduino based MIDI Controller (VIEIRA; SCHIAVONI, 2020);
- Current research on the use of HCI in decision-making to build digital musical instrument: A survey (VIEIRA; ROCHA; SCHIAVONI, 2020);

- As principais vantagens e os desafios do emprego da metodoogia STEAM no ensino de música na educação básica (VIEIRA; LUNHANI; SCHIAVONI, 2021);
- Automatic classification of instruments from supervised methods of machine learning (VIEIRA et al., 2021);
- Uma aproximação entre participação do público em espetáculo artístico e jogos: racionalidade e improviso (VIEIRA et al., 2020);
- Using game theory to support the audience participation in digital performances (ARAÚJO et al., 2021);
- A technical approach of the audience participation in the performance “O Chaos das 5" (ARAÚJO et al., 2019);
- O Chaos das 5 (SCHIAVONI et al., 2019))
- Por uma Arte Digital Povera (VIEIRA; JUNIOR; SCHIAVONI, 2021).

Ainda, foi publicado um artigo curto (Desafios da Internet das Coisas Musicais (ALMEIDA et al., 2021)) e um resumo (Controlador MIDI baseado na Plataforma Arduino (VIEIRA; VITORINO, 2019)).

7.1 Trabalhos Futuros

Visando a evolução deste sistema, como trabalhos futuros, pretende-se:

- Implementar pelo menos um dos cenários propostos ao longo do texto;
- Criar uma interface gráfica para o sistema;
- Atribuir inteligência as coisas musicais, de modo a coletar informações sobre o ambiente, os usuários e as próprias coisas musicais, para fornecer aos administradores maiores detalhes sobre estes elementos;
- Criar dispositivos que funcionem como *hubs* de áudio ou OSC (valendo-se da inteligência das coisas musicais, proposta em tópico anterior) para permitir que objetos com diferentes propriedades possam se comunicar, aumentando a heterogeneidade do ambiente;
- Pensar em maneiras de se utilizar um relógio global ou uma unidade de processamento local para reduzir a latência no áudio;
- Debater algoritmos e estratégias de codificação e decodificação para os vídeos e demais elementos gráficos que compõem o sistema;

- Realizar testes com público e em um ambiente não-controlado para analisar as questões que surgem quando diferentes usuários, com os mais diversos interesses artísticos e níveis de habilidade, utilizam o Sunflower.

REFERÊNCIAS

ALEXANDRAKI, C. et al. Towards the implementation of a generic platform for networked music performance: The diamouses approach. 08 2008.

ALMEIDA, A. et al. Desafios da internet das coisas musicais. In: **Proceedings of the 18th Brazilian Symposium on Computer Music**. Recife - PE - Brasil: Sociedade Brasileira de Computação, 2021. p. 252–255.

ARAÚJO, J. a. T. et al. A technical approach of the audience participation in the performance 'o chaos das 5'. In: SCHIAVONI, F. et al. (Ed.). **Proceedings of the 17th Brazilian Symposium on Computer Music**. São João del-Rei - MG - Brazil: Sociedade Brasileira de Computação, 2019. p. 28–34.

ARAÚJO, J. T. et al. Using game theory to support the audience participation in digital performances. **Per Musi**, n. 40, p. 1–13, Jun. 2021. Disponível em: <https://periodicos.ufmg.br/index.php/permusi/article/view/32675>.

ASHTON, K. That 'internet of things' thing. **RFID Journal**, RFID Journal, v. 15, p. 1, 1999.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Computer Networks**, p. 2787–2805, 10 2010.

AVGERIOU, P.; ZDUN, U. Architectural patterns revisited - a pattern language. In: **EuroPLoP' 2005, Tenth European Conference on Pattern Languages of Programs**. [S.l.: s.n.], 2005. v. 81, p. 431–470.

BARBOSA, A.; CARDOSO, J. Ritmo musical adaptável à latência de rede no sistema public sound objects. 05 2021.

BELETTINI, C. T. Bachelor's Thesis, **Estudo de viabilidade da utilização da tecnologia Power Line Communication - PLC em Redes Locais em Comparativo com Cabo de Par Trançado**. 2015.

BIJLSMA, A. et al. **Software Architecture**. [S.l.]: Free Technology Academy, 2011. ISBN N/A.

BOQUE, V.; HAHN, R. M.; ASTIAZARA, M. V. Cabo de par trançado e cabo coaxial. Unpublished. 2012.

BURGESS, R. J. **History of Music Production**. Oxford, UK: Oxford University Press, 2014.

BURGOYNE, J. A.; FUJINAGA, I.; DOWNIE, J. S. **Music information retrieval**. 1st. ed. [S.l.]: John Wiley & Sons, 2016.

BURNS, B. **Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services**. [S.l.]: O'Reilly, 2018.

BUSCHMANN, F. et al. **A System of Patterns - Pattern-Oriented Software Architecture**. [S.l.]: John Wiley & Sons, 1996. 476 p. ISBN 978-0471958697.

CALEGARIO, F. **Designing Digital Musical Instruments Using Probatio: A Physical Prototyping Toolkit**. Berlin, Alemanha: Springer, 2019.

CAMPS-MUR, D.; GARCIA-SAAVEDRA, A.; SERRANO, P. Device-to-device communications with wi-fi direct: overview and experimentation. **IEEE Wireless Communications Magazine**, v. 20, 06 2013.

CENTENARO, M.; CASARI, P.; TURCHET, L. Towards a 5g communication architecture for the internet of musical things. In: . [S.l.: s.n.], 2020. p. 38–45.

CORRÊA, S.; CERQUEIRA, R. Computação autônoma: Conceitos, infra-estruturas e soluções em sistemas distribuídos. 01 2009.

COULOURIS, G. et al. **Sistemas Distribuídos: Conceitos e Projetos**. [S.l.: s.n.], 2013. 1064 p. ISBN 978-8582600535.

DAVIS, P. T. **Securing Client/Server Computer Networks**. [S.l.: s.n.], 1996. 589 p. ISBN 978-0070158412.

EMMERICH, W. **Engineering Distributed Objects**. [S.l.]: Wiley, 2000.

ESCOBAR, E. **How Does Wi-Fi Work?** 2015. <https://www.scientificamerican.com/article/how-does-wi-fi-work/>.

FELLMANN, T.; KAVAKLI, M. A command line interface versus a graphical user interface in coding vr systems. **Proceedings of the 2nd IASTED International Conference on Human-Computer Interaction, HCI 2007**, p. 142–147, 03 2007.

FORNY, L. Arte e interação: Nos caminhos da arte interativa? **Razón y palabra, ISSN 1605-4806, Nº. 53, 2006**, p. http://www.razonypalabra.org.mx/anteriores/n53/lforny.html, 11 2006.

FOROUZAN, B. **Comunicação de Dados e Redes de Computadores**. 4a. ed. [S.l.]: AMGH, 2007.

FRAIETTA, A. Open sound control: Constraints and limitations. In: . [S.l.: s.n.], 2008.

FRANCOIS, A. Software architecture for computer vision: Beyond pipes and filters. 08 2003.

FUENTES, F. **Sistemas Distribuidos**. Cuajimalpa de Morelos, México: Universidad Autónoma Metropolitana, Cuajimalpa, 2015.

GABRIEL, E. et al. Distributed computing in a heterogeneous computing environment. In: ALEXANDROV, V.; DONGARRA, J. (Ed.). **Recent Advances in Parallel Virtual Machine and Message Passing Interface**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. p. 180–187. ISBN 978-3-540-49705-9.

GAINSFORD, P. **Pure Data: FLOSS Manual**. 3. ed. The address of the publisher, 1993. An optional note.

GAYE, L. et al. Mobile music technology: Report on an emerging community. In: **New Interfaces for Musical Expression**. [S.l.: s.n.], 2006. p. 22–25.

GERSHENFELD, N.; KRIKORIAN, R.; COHEN, D. The internet of things. **Scientific American**, v. 291, p. 76–81, 11 2004.

GU, X. et al. Nmp - a new networked music performance system. In: . [S.l.: s.n.], 2004. p. 176 – 185. ISBN 0-7803-8798-8.

HALLER, S. The things in the internet of things. **Bern University**, 01 2010.

JONES, K.; LIU, L. What where wi: An analysis of millions of wi-fi access points. In: **2007 IEEE International Conference on Portable Information Devices**. [S.l.: s.n.], 2007. p. 1–4.

JORDÀ, S. **Digital Lutherie : Crafting musical computers for new musics' performance and improvisation**. Tese (Doutorado) — Universitat Pompeu Fabra, 01 2005.

KAMBALYAL, C. **3-Tier Architecture**. 2010.

KELLER, D.; LAZZARINI, V.; PIMENTA, M. **Ubiquitous Music**. [S.l.: s.n.], 2014. ISBN 978-3-319-11151-3.

KELLER, D.; SCHIAVONI, F.; LAZZARINI, V. Ubiquitous music: Perspectives and challenges. **Journal of New Music Research**, v. 48, p. 1–7, 09 2019.

KUROSE, J.; ROSS, K. **Redes de Computadores e a Internet: Uma Abordagem Top-Down**. [S.l.: s.n.], 2013. 656 p. ISBN 978-8581436777.

LAZZARO, J.; WAWRZYNEK, J. A case for network musical performance. 06 2001.

MAIER, M.; EMERY, D.; HILLIARD, R. Software architecture: introducing ieee standard 1471. **Computer**, v. 34, p. 107 – 109, 05 2001.

MALLOCH, J.; SINCLAIR, S.; WANDERLEY, M. Libmapper (a library for connecting things). In: **CHI 2013 Extended Abstracts**. [S.l.: s.n.], 2013.

MARSAN, C. **The Internet of Things: An Overview**. [S.l.]: Internet Society, 2015.

MATTERN, F.; FLOERKEMEIER, C. From the internet of computers to the internet of things. In: . [S.l.: s.n.], 2010. v. 33, p. 242–259. ISBN 978-3-642-17225-0.

MEURISSE, T. et al. Simulations of modal active control applied to the self-sustained oscillations of the clarinet. In: . [S.l.: s.n.], 2013. v. 100.

MEURISSE, T. et al. An active mute for the trombone. **The Journal of the Acoustical Society of America**, v. 138, 12 2015.

MILLARD, A. **America on Record: A History of Recorded Sound**. Cambridge, UK: Cambridge University Press, 2005.

MIRANDA, E.; WANDERLEY, M. M. **New Digital Musical Instruments: Control And Interaction Beyond the Keyboard**. Middleton, Wisconsin, EUA: A-R Editions, 2006.

MONTEIRO, A. C. **Criação e performance musical no contexto dos instrumentos musicais digitais**. Dissertação (Mestrado) — Universidade Estadual de Campinas, Brasil, 2012.

MOOG, R. The musician: Alive and well in the world of electronics. In: **The Biology of Music Making: Proceedings of the 1984 Denver Conference**. St Louis, MO, EUA: MMB Music, Inc, 1988. p. 214–220.

MOZHAIEV, M.; KUCHUK, N.; USATENKO, M. The method of jitter determining in the telecomunication network of a computer system on a special software platform. **Innovative Technologies and Scientific Solutions for Industries**, p. 134–140, 12 2019.

NARDIM, T. **Allan Kaprow, performance e colaboração : estrategias para abraçar a vida como potencia criativa**. 1-152 p. Tese (Doutorado) — Universidade de Campinas - UNICAMP, 08 2009.

OLIVEIRA, R. de; FRAGA, J.; MONTEZ, C. Programação em sistemas distribuídos. p. 1–49, 2002.

OLUWATOSIN, H. S. Client-server model. p. 1–5, 02 2014.

PARGMAN, T. C.; ROSSITTO, C.; BARKHUUS, L. Understanding audience participation in an interactive theater performance. In: . [S.l.: s.n.], 2014.

PATRICIO, E. **Instrumentos Musicais Digitais - Uma abordagem composicional**. Dissertação (Mestrado) — UFPR, Brasil, 2010.

PRESSING, J. Cybernetic issues in interactive performance systems. **Computer Music Journal**, v. 14, n. 1, p. 12–25, 1990.

PUCKETTE, M. Pure data. In: . [S.l.: s.n.], 1996. p. 37–41.

RABELO, N. S. **Utilização de Bancos de Dados e Arquitetura Cliente Servidor em Ambientes Comerciais**. Dissertação (Mestrado) — Centro Universitário do Triângulo, Uberlândia, Brazil, 1998.

REAS, C.; FRY, B. **Getting Started with Processing: A Quick, Hands-on Introduction**. 1st. ed. [S.l.]: O'Reilly Media, 2010.

ROCHA, G. L.; ARAÚJO, J. T.; SCHIAVONI, F. L. Ha dou ken music: Different mappings to play music with joysticks. In: **Proceedings of the Int. Conf. on NIME**. Porto Alegre, Brazil: UFRGS, 2019. p. 77–78. ISSN 2220-4806.

ROGERS, Y.; SHARP, H.; PREECE, J. **Interaction Design: Beyond Human-Computer Interaction**. 3rd. ed. [S.l.]: John Wiley & Sons, 2011.

ROSE, B.; HAIGHTON, T.; LIU, D. Open sound control. Unpublished. 2015.

RUFINO, N. M. de O. **Segurança em Redes sem fio: Aprenda a Proteger Suas Informações em Ambientes Wi-Fi e Bluetooth**. 4a. ed. [S.l.]: Novatec Editora, 2014.

SANTANA, I. **Design de Sistemas Distribuídos — Introdução a Design de Sistemas e Sistemas Distribuídos**. 2019. <https://medium.com/xp-inc/design-de-sistemas-distribu%C3%ADdos-introdu%C3%A7%C3%A3o-a-design-de-sistemas-e-sistemas-distribu%C3%ADdos-3b5fd43f9d86>.

SCHIAVONI, F.; QUEIROZ, M.; IAZZETTA, F. Medusa -a distributed sound environment. In: . [S.l.: s.n.], 2011.

SCHIAVONI, F.; QUEIROZ, M.; WANDERLEY, M. Network music with medusa: A comparison of tempo alignment in existing midi apis. In: . [S.l.: s.n.], 2013.

SCHIAVONI, F. L. et al. O chaos das 5. In: SCHIAVONI, F. et al. (Ed.). **Proceedings of the 17th Brazilian Symposium on Computer Music**. São João del-Rei - MG - Brazil: Sociedade Brasileira de Computação, 2019. p. 227–228.

SHAW, M.; GARLAN, D. **Software Architecture: Perspectives on an Emerging Discipline**. Hoboken, NJ, EUA: Prentice Hall, 1996.

SMUS, B. **Web Audio API: Advanced Sound for Games and Interactive Apps**. [S.l.: s.n.], 2013. ISBN 978-1449332686.

SUN, W. et al. Wi-fi could be much more. **IEEE Communications Magazine**, v. 52, n. 11, p. 22–29, 2014.

TANENBAUM, A.; STEEN, M. V. **Sistemas Distribuídos: Princípios e Paradigmas**. [S.l.]: Pearson Universidades, 2007. 416 p. ISBN 978-8576051428.

TAROUCO, L. M. **Redes de Computadores Locais e de Longa Distância**. [S.l.]: McGraw-Hill, 1986.

THALER, J. **System Architectures**. [S.l.]: FH Vorarlberg University of Applied Sciences, 2020.

TROCCO, F.; PINCH, T. **Analog Days: The Invention and Impact of the Moog Synthesizer**. [S.l.]: Harvard University Press, 2004. 368 p. ISBN 0674016173.

TURCHET, L. Smart mandolin: Autobiographical design, implementation, use cases, and lessons learned. In: **Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion**. New York, NY, USA: Association for Computing Machinery, 2018. (AM'18). ISBN 9781450366090. Disponível em: <https://doi.org/10.1145/3243274.3243280>.

TURCHET, L. Smart musical instruments: Vision, design principles, and future directions. **IEEE Access**, v. 7, p. 8944–8963, 01 2019.

TURCHET, L. et al. The internet of musical things ontology. **SSRN Electronic Journal**, 01 2020.

TURCHET, L.; BARTHET, M. Jamming with a smart mandolin and freesound-based accompaniment. In: . [S.l.: s.n.], 2018.

TURCHET, L.; BENINCASO, M.; FISCHIONE, C. Examples of use cases with smart instruments. In: . New York, NY, USA: Association for Computing Machinery, 2017. (AM '17). ISBN 9781450353731. Disponível em: <https://doi.org/10.1145/3123514.3123553>.

TURCHET, L. et al. Internet of musical things: Vision and challenges. **IEEE Access**, v. 6, p. 61994–62017, 09 2018.

TURCHET, L.; MCPHERSON, A.; BARTHET, M. Co-design of a smart cajón. **Journal of the Audio Engineering Society**, v. 66, 02 2018.

TURCHET, L.; MCPHERSON, A.; FISCHIONE, C. Smart instruments: Towards an ecosystem of interoperable devices connecting performers and audiences. In: **Proceedings of Sound and Music Computing Conference**. [S.l.: s.n.], 2016.

TURCHET, L. et al. Towards a semantic architecture for the internet of musical things. In: **IEEE Open Innovations Association**. [S.l.: s.n.], 2018.

TURCHET, L.; WEST, T.; WANDERLEY, M. Touching the audience: musical haptic wearables for augmented and participatory live music performances. **Personal and Ubiquitous Computing**, 03 2020.

UNWIN, A.; HEIKE, H. Gui and command-line - conflict or synergy? 01 2000.

VIEIRA, R. et al. Automatic classification of instruments from supervised methods ofmachine learning. In: **Proceedings of the 18th Brazilian Symposium on Computer Music**. Recife - PE - Brasil: Sociedade Brasileira de Computação, 2021. p. 28–34.

VIEIRA, R.; BARTHET, M.; SCHIAVONI, F. L. Everyday use of the internet of musical things. In: **Proceedings of the Workshop on Ubiquitous Music 2020**. Porto Seguro, BA, Brasil: Zenodo, 2020. p. 60–71. Disponível em: <https://doi.org/10.5281/zenodo.4247759>.

VIEIRA, R.; GONÇALVES, L.; SCHIAVONI, F. The things of the internet of musical things: defining the difficulties to standardize the behavior of these devices. In: **2020 X Brazilian Symposium on Computing Systems Engineering (SBESC)**. [S.l.: s.n.], 2020. p. 1–7.

VIEIRA, R.; JUNIOR, I.; SCHIAVONI, F. Por uma arte digital povera. In: **Proceedings of the 6th Congresso Internacional de Arte, Ciência e Tecnologia**. [S.l.: s.n.], 2021.

VIEIRA, R. et al. Uma aproximação entre participação do público em espetáculo artístico e jogos: racionalidade e improviso. In: **Proceedings of SBGames 2020**. Recife, PE, Brasil: SBC, 2020. p. 116–124. Disponível em: <https://www.sbgames.org/proceedings2020/ArtesDesignFull/209608.pdf>.

VIEIRA, R.; LUNHANI, G.; SCHIAVONI, F. L. As principais vantagens e os desafios do emprego da metodoogia steam no ensino de música na educação básica. In: **Proceedings of the Workshop on Ubiquitous Music 2021**. Porto, Portugal: [s.n.], 2021. p. 1–13. Disponível em: <https://dei.fe.up.pt/ubimus/>.

VIEIRA, R.; ROCHA, G.; SCHIAVONI, F. Current research on the use of hci in decision-making to build digital musical instruments: A survey. In: **Proceedings of the 19th Brazilian Symposium on Human Factors in Computing Systems**. New York, NY, USA: Association for Computing Machinery, 2020. (IHC '20). ISBN 9781450381727. Disponível em: <https://doi.org/10.1145/3424953.3426646>.

VIEIRA, R.; SCHIAVONI, F. L. Can pipe-and-filters architecture help creativity in music? In: **Proceedings of the Workshop on Ubiquitous Music 2020**. Porto Seguro, BA, Brasil: Zenodo, 2020. p. 109–120. Disponível em: <https://doi.org/10.5281/zenodo.4247691>.

VIEIRA, R.; SCHIAVONI, F. L. Managing an iomust environment and devices. In: **Proceedings of the Workshop on Ubiquitous Music 2021**. Porto, Portugal: [s.n.], 2021. p. 1–13. Disponível em: <https://dei.fe.up.pt/ubimus/>.

VIEIRA, R.; SCHIAVONI, F. L. Sunflower: an environment for standardized communication of iomust. In: **Proceedings of the Audio Mostly 2021**. [S.l.: s.n.], 2021.

VIEIRA, R.; VITORINO, A. Controlador midi baseado na plataforma arduino. In: **Proceedings of the Workshop de Computação Teórica e Aplicada**. [S.l.: s.n.], 2019.

VIEIRA, R. A.; SCHIAVONI, F. L. Fliperama: An affordable arduino based midi controller. In: MICHON, R.; SCHROEDER, F. (Ed.). **Proceedings of the International Conference on New Interfaces for Musical Expression**. Birmingham, UK: Birmingham City University, 2020. p. 375–379. ISSN 2220-4806. Disponível em: <https://www.nime.org/proceedings/2020/nime2020_paper73.pdf>.

WEINBERG, G. The aesthetics, history and future challenges of interconnected music networks. 01 2002.

WRIGHT, M. Open sound control: An enabling technology for musical networking. **Org. Sound**, v. 10, p. 193–200, 12 2005.