Universidade Federal do Espírito Santo
Centro Tecnológico
Programa de Pós-Graduação em Engenharia Elétrica
Dissertação de Mestrado

Higor Araújo Fim Camporez

RoboMus: Compensação de Atraso Mecânico e Sincronização Rítmica em Robôs Musicais

Vitória, ES, Brasil
Maio 2020

Higor Araújo Fim Camporez

RoboMus: Compensação de Atraso Mecânico e Sincronização Rítmica em Robôs Musicais

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica.

Universidade Federal do Espírito Santo – UFES

Departamento de Engenharia Elétrica

Programa de Pós-Graduação

Orientador: Helder Roberto de Oliveira Rocha

Coorientador: Jair Adriano Lima Silva

Vitória, ES, Brasil

Maio 2020

Ficha catalográfica disponibilizada pelo Sistema Integrado de
Bibliotecas - SIBI/UFES e elaborada pelo autor

Camporez, Higor Araújo Fim, 1993-
C198r RoboMus : Compensação de Atraso Mecânico e Sincronização Rítmica em Robôs Musicais / Higor Araújo Fim Camporez. - 2020.
118 f. : il.

Orientador: Helder Roberto de Oliveira Rocha.
Coorientador: Jair Adriano Lima Silva.
Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Performances Musicais Robóticas. 2. Sincronização de robôs. 3. Ubimus. 4. Internet das Coisas Musicais. 5. Compensação de atrasos mecânicos. 6. Redes Neurais. I. Rocha, Helder Roberto de Oliveira. II. Silva, Jair Adriano Lima. III. Universidade Federal do Espírito Santo. Centro Tecnológico. IV. Título.

CDU: 621.3

Higor Araújo Fim Camporez

RoboMus: Compensação de Atraso Mecânico e Sincronização Rítmica em Robôs Musicais

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica.

Trabalho aprovado. Vitória, ES, Brasil, 28 de Maio de 2020

Prof. Dr. Helder Roberto de Oliveira Rocha
Orientador

Prof. Dr. Jair Adriano Lima Silva
Coorientador

Prof. Dr. Arnaldo Gomes Leal Júnior
Universidade Federal do Espírito Santo
Examinador Interno

Prof. Dr. Evandro Manara Miletto
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul
Examinador Externo

Vitória, ES, Brasil
Maio 2020

Este documento foi assinado digitalmente por [redacted] ROCHA
Para verificar o original visite: https://api.lepisma.ufes.br/arquivos-assinados/54966?tipoArquivo=O

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

PROTOCOLO DE ASSINATURA

O documento acima foi assinado digitalmente com senha eletrônica através do Protocolo Web, conforme Portaria UFES nº 1.269 de 30/08/2018, por
HELDER ROBERTO DE OLIVEIRA ROCHA - SIAPE 1860639
Departamento de Engenharia Elétrica - DEE/CT
Em 27/08/2020 às 13:24

Para verificar as assinaturas e visualizar o documento original acesse o link:
https://api.lepisma.ufes.br/arquivos-assinados/54949?tipoArquivo=O

Este documento foi assinado digitalmente por HELDER ROBERTO DE OLIVEIRA ROCHA
Para verificar o original visite: https://api.lepisma.ufes.br/arquivos-assinados/54949?tipoArquivo=O

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

PROTOCOLO DE ASSINATURA

O documento acima foi assinado digitalmente com senha eletrônica através do Protocolo Web, conforme Portaria UFES nº 1.269 de 30/08/2018, por
ARNALDO GOMES LEAL JUNIOR - SIAPE 3106424
Departamento de Engenharia Mecânica - DEM/CT
Em 27/08/2020 às 13:48

Para verificar as assinaturas e visualizar o documento original acesse o link:
https://api.lepisma.ufes.br/arquivos-assinados/54966?tipoArquivo=O

Este documento foi assinado digitalmente por ARNALDO GOMES LEAL JUNIOR
Para verificar o original visite: https://api.lepisma.ufes.br/arquivos-assinados/54966?tipoArquivo=O

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

PROTOCOLO DE ASSINATURA

O documento acima foi assinado digitalmente com senha eletrônica através do Protocolo Web, conforme Portaria UFES nº 1.269 de 30/08/2018, por
JAIR ADRIANO LIMA SILVA - SIAPE 1514954
Departamento de Engenharia Elétrica - DEE/CT
Em 27/08/2020 às 14:49

Para verificar as assinaturas e visualizar o documento original acesse o link:
https://api.lepisma.ufes.br/arquivos-assinados/54996?tipoArquivo=O

Este documento foi assinado digitalmente por JAIR ADRIANO LIMA SILVA
Para verificar o original visite: https://api.lepisma.ufes.br/arquivos-assinados/54996?tipoArquivo=O

Este trabalho é dedicado a todos os professores
que fizeram parte de todo o meu aprendizado.

Agradecimentos

Agradeço, primeiramente, a Deus por ter guiado meus esforços e não ter me deixado desistir no decorrer dessa árdua estrada, proporcionando-me forças para enfrentar e superar as dificuldades. Agradeço aos meus pais, Iramaia Inês Araújo e José Reinaldo Fim Camporez, por terem se empenhado na minha formação educacional e por terem me apoiado nessa jornada. Em especial, agradeço a minha namorada, Yasmim Sagrillo Pimassoni, por toda ajuda, todo apoio, toda paciência, toda compreensão, todos momentos juntos e muito mais.

Agradeço ao NESCoM (Núcleo Espírito-Santense de Computação Musical) que em 2014, num clima natalino, na UFES campus São Mateus, onde eu ainda era um aluno de graduação, me aceitou para participar do projeto CAMPUS VIVO e me mostrou o mágico mundo da pesquisa. Ainda agradeço especialmente ao coordenador desse grupo de pesquisa, Prof. Dr. Leandro Lesqueves Costalonga, que foi coorientador externo nesta dissertação de mestrado e que desde então não tem medido esforços para me apoiar no desenvolvimento deste trabalho. Agradeço ao Leandro por todos os ensinamentos, direcionamentos, paciência, exemplos e muito mais. Agradeço também aos ensinamentos que perpassam os técnicos, que possibilitaram e continuam possibilitando torna-me um ser humano melhor. Inclusive acredito, fortemente, que o produto desta dissertação não está apenas no conteúdo escrito e desenvolvido, mas também na minha evolução como ser humano.

Agradeço a todos os professores do PPGEE (Programa de Pós-Graduação em Engenharia Elétrica), em especial aos meus orientadores Prof. Dr. Helder Roberto de Oliveira Rocha e Prof. Dr. Jair Adriano Lima Silva que me proporcionaram a oportunidade de desenvolver este mestrado, agradeço aos ensinamentos (dentro e fora da pesquisa), aos direcionamentos, a paciência e todo aprendizado.

Agradeço a todos os amigos do LabTel (Laboratório de Telecomunicações) da UFES pela ajuda em momentos que precisei, pelos momentos de descontração, pelos cafés, almoços com muita alegria ... e por tudo.

Agradeço a UFES, PPGEE e LabTel por me proporcionar desenvolver este trabalho, bem como a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) que apoiou financeiramente este trabalho - Código de Financiamento 001.

"happiness only real when shared"
Alexander Supertramp

Resumo

As atividades musicais como uma ferramenta de diversão, entretenimento, interação social, expressividade criativa e outros, se tornaram aliadas do ser humano no seu cotidiano. Os avanços tecnológicos têm propiciado a criação de ferramentas que apoiem as atividades voltadas à música, por exemplo, o computador. Com isso, a interseção entre a tecnologia e a música proporcionou a criação de áreas de pesquisa como ubimus (*ubiquitous music*), que faz uso de sistemas ubíquos com agentes humanos e recursos materiais (ferramentas de suporte a criatividade) para criação musical, e a área de IoMusT (*Internet of Musical Things*), que propicia aplicações de ubimus em infraestrutura de IoT (*Internet of Things*). Sendo que essas duas áreas englobam-se no desenvolvimento de instrumentos musicais robóticos, podendo estes serem controlados por humanos, através de interfaces de interação, ou autônomos.

Este trabalho visa desenvolver uma plataforma para performances musicais robóticas controladas por interfaces de interação, centralizando esforços no desenvolvimento de estratégias de sincronização, além de demonstrar o desenvolvimento de um bongô robótico (BongoBot). A plataforma faz uso do protocolo de comunicação musical *Open Sound Control* (OSC) para definir todos os formatos das mensagens de comunicação, onde o formato definido visa não limitar as futuras gerações de robôs, assim sendo, cada robô pode enviar suas ações específicas com os seus parâmetros de controle. Como o foco do trabalho encontra-se na sincronização dos robôs, algoritmos de macrossincronização são propostos utilizando SNTP (*Simple Network Time Protocol*) para sincronização de relógio e redes neurais pra treinamento e compensação dos atrasos mecânicos presente nos robôs. Para microssincronização implementou-se um módulo perceptual autônomo, onde utiliza-se o processamento de áudio em tempo real para que o robô possa "ouvir" e acompanhar ritmicamente uma performance.

Os testes para macrossincronização foram feitos com dois robôs musicais, as redes neurais apresentaram RMSE (*Root Mean Square Error*) de 0,82% e 1,2%. Também foi alcançado uma melhoria de até 55,33% no atraso médio da performance sem compensação do atraso mecânico e, ainda, todos os atrasos entre as batidas dos robôs ficaram abaixo do limiar da percepção sonora humana, estipulado em 20ms. A microssincronização (módulo perceptual autônomo) foi aplicada ao BongoBot para sincronização rítmica com um metrônomo, uma avaliação numérica mostrou que 80% dos casos ficaram com atraso médio abaixo de 20ms e uma avaliação perceptual, desenvolvida por pessoas através de um questionário *online*, demonstrou que 55% dos testes tiveram aceitação acima de 70%, ou seja, percentual de votos com respostas positivas para a pergunta: *você considera que os pulsos sonoros nesse áudio estão em sincronia?*.

Palavras-chave: Performances Musicais Robóticas. Sincronização de robôs. Ubimus. Internet das Coisas Musicais. Compensação de atrasos mecânicos. Sincronização Rítmica. Redes Neurais.

Abstract

Musical activities as a tool for fun, entertainment, social interaction, creative expressiveness, and others, have become allies of the human being in their daily lives. Technological advances have enabled the creation of tools that support musical activities, for example, the computer. The intersection between technology and music provided the creation of new research areas such as ubimus (*ubiquitous music*) that uses ubiquitous systems with human agents and material resources (tools to support creativity) for musical purposes. There is also the area named IoMusT (*Internet of Musical Things*) that provides ubimus applications in IoT (*Internet of Things*) infrastructure. These areas intersect with robotic musical instruments, which can be controlled by humans, through interaction interfaces, or operate in autonomous mode.

This master's dissertation aims the development of a platform for networked robotic musical performances through interaction interfaces to control robots, centralizing efforts on synchronization strategies. In addition, this dissertation shows the robotic bongo development, so-called BongoBot. The platform uses the musical communication protocol *Open Sound Control* (OSC) to define all message formats. As the focus is on synchronization, macro-synchronization algorithms are proposed using SNTP for clocks synchronization and neural networks for training and compensation of the mechanical delays contained in robots. For micro-synchronization, an autonomous perceptual module was implemented, using real-time audio processing to allow the robot to "hear" the performance and synchronize rhythmically.

Tests for macro-synchronization were done with two robots, the neural networks presented an RMSE (root mean square error) error of 0.82% and 1.2%. This strategy also provided a reduction up to 55.33% in the average performance delay and kept all the beats delay of the robots below 20ms (threshold of human sound perception). Micro-synchronization (autonomous perceptual module) was applied to the BongoBot for rhythmic synchronization with a metronome. The numerical evaluation showed that 80% of the tests had an average delay below 20ms and the perceptual evaluation, developed by an online survey, described that 55% of the tests had above 70% of positive votes, that is, percentage of votes in which participants agreed with the synchronization, answering yes for the following question: *Do you consider that the sound pulses in this audio are in sync?*.

Keywords: Networked Musical Performance. Robots Synchronization. Ubimus. Internet of Musical Things. Mechanical Delay Compensation by Neural Network.

Lista de ilustrações

Figura 1 – Banda musical robótica The Three Sirens: (a) guitarra elétrica robótica com *slide* (Aglaopheme), (b) contra-baixo robótico (Peisinoe) e (c) bateria robótica (Thelxiepeia). 38
Figura 2 – Instrumentos musicais robóticos: (a) guitarra elétrica robótica, (b) guitarra elétrica com *slide* (LEMUR GuitarBot) e (c) dispositivo de slide para cada corda do GuitarBot. 39
Figura 3 – Robô Haile: (a) estrutura do robô, (b) performance colaborativa com o robô e (c) braço direito do robô 40
Figura 4 – Marimba robótico Shimon 41
Figura 5 – Exemplo de aplicação da STFT em um sinal. 46
Figura 6 – Filtros triangulares espaçados na frequência 47
Figura 7 – Esquemático para geração do espectrograma Mel 48
Figura 8 – Gráficos do OSE, da autocorrelação do OSE e aplicação da janela no gráfico da autocorrelação. 49
Figura 9 – (a) Gráfico do espectrograma de um sinal, representando as intensidades de frequência para cada *frame* por cores e (b) ilustração da matriz de similaridade, que representa níveis de similaridade entre *frames*. 52
Figura 10 – Nova matriz de similaridade gerada a partir da Figura 9 (b). 53
Figura 11 – Troca de mensagens entre um cliente e um servidor SNTP 55
Figura 12 – Modelo de um neurônio artificial 56
Figura 13 – Função de ativação Sigmóide com $\alpha = 1$ e função de ativação ReLu . . 57
Figura 14 – Estrutura de rede neural de múltiplas camadas 58
Figura 15 – Arquitetura da plataforma RoboMus 62
Figura 16 – Interface assistiva para controle musical chamada Olhar Musical 64
Figura 17 – Estrutura genérica para variáveis de entrada e saída para as redes neurais utilizadas no SSMM 74
Figura 18 – Diagrama do processo de microssincronização baseado em pulsos musicais 74
Figura 19 – Estruturas de controle para a plataforma RoboMus 75
Figura 20 – A esquerda encontra-se o modelo 3D do BongoBot e, à direita, a representação real 76
Figura 21 – (a) Circuito *driver* e (b) circuito amplificador 77
Figura 22 – Diagrama de blocos representando as estruturas do BongoBot 77
Figura 23 – Setup para estimação da diferença entre relógios 79
Figura 24 – Diagrama do *setup* experimental para estimar o atraso entre robôs bongô 80
Figura 25 – Diagrama do *setup* experimental para avaliação do atraso dos robôs após a aplicação dos algoritmos de microssincronização 81

Figura 26 – Atraso mecânico do robô BongoBot em relação ao tambor maior (*hembra*): (a) o perfil de atraso do solenoide central e (b) o perfil do solenoide lateral 84
Figura 27 – Atraso mecânico do robô BongoBot relacionado ao tambor menor (*macho*): (a) o perfil de atraso do solenoide central e (b) o perfil de atraso do solenoide lateral 84
Figura 28 – Atrasos entre os relógios de dois dispositivos Raspberry: (a) mostra o resultado de um teste promovendo a sincronização de relógio apenas no início e (b) demostra o resultado do atraso fazendo uma sincronização a cada 30 segundos 85
Figura 29 – Atrasos entre os dois robôs bongô: (a) apresenta os resultados para o teste 1 e (b) os resultado para o teste 2 86
Figura 30 – Estrutura da rede neural usada especificamente para os dois bongôs robóticos 87
Figura 31 – Dados de treinamento das redes neurais: (a) mostra os dados de *loss* para o robô 1 *hembra* e (b) ilustra os dados de *loss* para o robô 2 *macho* 87
Figura 32 – Dados de teste das redes neurais: (a) mostra os dados de reais e previstos do atraso mecânico do robô 1 *hembra* e (b) ilustra os dados de reais e previstos do atraso mecânico do robô 2 *macho* 88
Figura 33 – Atrasos entre os dois robôs bongô: (a) apresenta os resultados para o teste 3 e (b) os resultado para o teste 4 89
Figura 34 – Gráficos com os 10 primeiros segundos dos sinais de áudio e batidas detectadas: (a) representa os sinais proveniente do metrônomo e (b) os sinais do bongô robótico 91
Figura 35 – Gráficos com os 10 últimos segundos dos sinais de áudio e batidas detectadas: (a) representa os sinais do metrônomo e (b) os sinais do bongô robótico 93
Figura 36 – Atraso entre o metrônomo e o bongô robótico 93
Figura 37 – Gráfico mostrando a quantidade absoluta de votos sim e não para cada teste 95
Figura 38 – Gráfico mostrando o percentual de votos sim versus o atraso médio calculado 95
Figura 39 – Gráfico mostrando a quantidade absoluta de votos sim e não para cada teste considerando o grupo com experiência 1 ou 2 97
Figura 40 – Gráfico mostrando o percentual de votos sim, para o grupo com experiência 1 ou 2, pelo atraso médio calculado 97
Figura 41 – Gráfico mostrando a quantidade absoluta de votos sim e não para cada teste considerando o grupo com experiência 4 ou 5 99

Figura 42 – Gráfico mostrando o percentual de votos sim, para o grupo com experiência 4 ou 5, pelo atraso médio calculado 99
Figura 43 – Foto de uma *Lap Steel Guitar* 113
Figura 44 – Estrutura do *lap steel guitar* robótico 114
Figura 45 – Fotos do *lap steel guitar* robótico: (a) dispositivo de excitação das cordas, (b) dispositivo de excitação e abafamento das cordas e (c) foto do robô completo 115

Lista de quadros

Quadro 1 – Tipos de dados da plataforma RoboMus 66
Quadro 2 – Formato da mensagem de *handshake* de um robô 66
Quadro 3 – Formato da mensagem de *handshake* da interface 67
Quadro 4 – Formato da mensagem de resposta ao *handshake* 67
Quadro 5 – Formato da mensagem de desconexão de um robô ou interface 68
Quadro 6 – Formato da mensagem para informar às interfaces sobre a desconexão de um robô 68
Quadro 7 – Formato da mensagem para cancelar uma mensagem RoboMus 68
Quadro 8 – Formato da mensagem para requisitar informações dos robôs disponíveis 69
Quadro 9 – Formato da mensagem para repasse das informações de um robô . . . 69
Quadro 10 – Formato da mensagem para retorno do atraso (tempo gasto para executar uma ação) 69
Quadro 11 – Formato da mensagem para a ação de tocar uma nota 70
Quadro 12 – Formato da mensagem para a ação de tocar uma nota de um instrumento do tipo trasteado. 72
Quadro 13 – Formato da Mensagem tocar o tambor maior (*hembra*). 78
Quadro 14 – Formato da Mensagem tocar o tambor menor (*macho*). 78
Quadro 15 – Formato da mensagem de slide 115
Quadro 16 – Formato da mensagem para tocar cordas 116
Quadro 17 – Formato da mensagem para silenciar cordas 116

Lista de tabelas

Tabela 1 – Resultados dos testes para avaliar o atraso mecânico entre os robôs . . 89
Tabela 2 – Comparação dos resultados após aplicação da compensação dos atrasos mecânicos por redes neurais 89
Tabela 3 – Resultados experimentais dos testes de microssincronização por pulsos musicais. 91
Tabela 4 – Resultados experimentais dos testes de microssincronização por pulsos musicais considerando apenas os 10 primeiros segundos. 92
Tabela 5 – Resultados perceptuais e calculados para os testes de microssincronização por pulsos musicais. 94
Tabela 6 – Tabela demostrando a quantidade absoluta e percentual dos teste de acordo com o percentual de aceitação pelo público. 96
Tabela 7 – Resultados perceptuais para os testes de microssincronização por pulsos musicais levando em consideração o grupo com experiência 1 ou 2. . . . 96
Tabela 8 – Tabela demostrando a quantidade absoluta e percentual dos teste de acordo com o percentual de aceitação pelo público do grupo com experiência 1 ou 2. 98
Tabela 9 – Resultados perceptuais dos testes de microssincronização executados pelo grupo com experiência 4 ou 5. 98
Tabela 10 – Tabela demostrando a quantidade absoluta e percentual dos teste de acordo com o percentual de aceitação pelo público do grupo com experiência 4 ou 5. 100
Tabela 11 – Comparação entre os resultados do grupo geral, do grupo com experiência 1 ou 2 e com experiência 4 ou 5. 100

Lista de abreviaturas e siglas

BPM	Batidas Por Minuto
CC	Corrente Contínua
DORT	Distúrbios Osteomusculares Relacionados ao Trabalho
FT	*Fourier Transform*
GPIO	*General-Purpose Input/Output*
IHR	Interação Humano-Robô
IHC	Interação Homem-Computador
IoMuT	*Internet of Musical Things*
IoT	*Internet of Things*
LER	Lesões por Esforço Repetitivo
MIDI	*Musical Instrument Digital Interface*
NTP	*Network Time Protocol*
OSE	*Onset Strength Envelope*
OSC	*Open Sound Control*
PC	*Personal Computer*
PWM	*Pulse Width Modulation*
ReLU	*Rectified Linear Unit*
RMSE	*Root Mean Square Error*
RTT	*Round Trip Time*
SNTP	*Simple Network Time Protocol*
SSMM	Servidor de Sincronização de Mensagens Musicais
STFT	*Short-Term Fourier transform*
Ubicomp	*ubiquitous computing*
Ubimus	*Ubiquitous Music*
USB	*Universal Serial Bus*

Lista de símbolos

$\Delta t_{Relógio}$	Diferença de tempo entre o relógios cliente e do servidor SNTP
t_1	Momento em que o cliente envia a requisição ao servidor SNTP
t_2	Momento em que a requisição do cliente chega ao servidor SNTP
t_3	Momento em que o servidor SNTP envia a resposta ao cliente
t_4	Momento em que a resposta da requisição feita chega ao cliente
t_{RTT}	Tempo de ida e volta das mensagens SNTP
Δ_m	Fator de deslocamento da média
$max()$	Função que retorna o máximo valor de um vetor
$mean()$	Função que retorna a média de um vetor
n	n-ésima amostra de um sinal de áudio
$post_avg$	Intervalo superior para determinar o valor médio
$post_max$	Intervalo superior para determinar o máximo valor
pre_avg	Intervalo inferior para determinar o valor médio
pre_max	Intervalo inferior para determinar o máximo valor
$previous_n$	Amostra onde ocorreu o último pico
$wait$	Intervalo mínimo entre dois picos consecutivos
x	Sinal digital de áudio
X_{STFT}	Sinal no domínio da frequência
d	Deslocamento da janela
k	Ponto da transformada de frequência
g	Representa a função da janela
L	Tamanho do sinal da janela
N	Tamanho do sinal da janela

f	Frequência em Hz
b	Frequência em escala Mel
m	Número da banda Mel
Fs	Taxa de amostragem
M	Número total de bandas
f_l	Menor frequência em Hz
f_h	Maior frequência em Hz
t	Representa o tempo
τ	Deslocamento temporal
τ_0	Tempo central da janela
σ_τ	Parâmetro de controla da curva de ponderação
$\{t_i\}$	Sequência de pulsos
N_b	Número de pulsos
α	Fator de balanceamento entre os termos
Δt	Intervalo de tempo
τ_p	Intervalo de tempo ideal
G	Compassos completos
P	Compassos incompletos
SCS	Nível de similaridade entre dois compassos completos
SIS	Nível de similaridade entre dois compassos incompletos
Bar	Tamanho do compasso completo
r	Tamanho do compasso incompleto

Sumário

1 INTRODUÇÃO . . . 31
1.1 Motivação . . . 32
1.2 Objetivos Gerais . . . 33
1.3 Objetivos Específicos . . . 33
1.4 Metodologia . . . 34
1.5 Estrutura da Dissertação . . . 34
1.6 Principais Contribuições . . . 35

2 REFERENCIAIS TEÓRICOS . . . 37
2.1 Robôs Musicais . . . 37
2.2 Ubimus . . . 40
2.3 Internet of Musical Things . . . 42
2.4 Processamento Digital de Sinais de Áudio . . . 44
2.4.1 Detecção de Picos em Sinais . . . 44
2.4.2 Transformada de Fourier de Curto Termo . . . 45
2.4.3 Espectrograma Mel . . . 45
2.4.4 Sistema de Rastreamento de Batidas . . . 47
2.4.4.1 Onset Strength Envelope . . . 48
2.4.4.2 BPM Global . . . 48
2.4.4.3 Formulação do Rastreador de Batidas . . . 50
2.4.5 Extração da Assinatura de Tempo . . . 50
2.5 Open Sound Control . . . 53
2.6 Simple Network Time Protocol . . . 54
2.7 Redes Neurais Artificiais . . . 56
2.7.1 Modelo de um Neurônio Artificial . . . 56
2.7.2 Funções de Ativação . . . 57
2.7.3 Estrutura de Redes de Múltiplas Camadas Alimentadas Diretamente . . . 57
2.8 Resumo do Capítulo . . . 59

3 ROBOMUS . . . 61
3.1 A plataforma RoboMus . . . 61
3.2 Servidor de Sincronização de Mensagens Musicais . . . 61
3.3 Interação Humano-Robô . . . 62
3.4 Comunicação RoboMus . . . 64
3.4.1 Atributos dos Robôs . . . 64
3.4.2 Mensagem de Apresentação . . . 66

3.4.3 Mensagem de Desconexão . . . 67
3.4.4 Mensagem para Cancelamento de uma Ação . . . 68
3.4.5 Mensagem para Repasse de Informações dos Robôs . . . 68
3.4.6 Mensagem de Atraso . . . 69
3.4.7 Mensagem para Execução de Nota . . . 70
3.4.7.1 Formato de Mensagens para a Família de Instrumentos Trasteados de Cordas . . . 70
3.5 Sincronização entre Robôs . . . 72
3.5.1 Macrossincronização . . . 72
3.5.2 Microssincronização . . . 73
3.6 Estrutura de Controle dos Robôs RoboMus . . . 75
3.7 Protótipo Bongô RoboMus . . . 76
3.7.1 Formato das Mensagens do BongoBot . . . 77
3.8 Metodologia para Avaliação da Macrossincronização entre Relógios 78
3.9 Metodologia para Avaliação Macrossincronização entre Robôs . . . 79
3.10 Metodologia para Avaliação da Microssincronização . . . 79
3.10.1 Avaliação Numérica . . . 80
3.10.2 Avaliação Perceptual . . . 81
3.11 Resumo do Capítulo . . . 81

4 RESULTADOS EXPERIMENTAIS . . . 83
4.1 Caracterização do Atraso Mecânico do BongoBot . . . 83
4.2 Macrossincronização entre Relógios . . . 83
4.3 Macrossincronização entre Robôs Musicais . . . 85
4.4 Microssincronização por Pulsos Musicais . . . 90
4.4.1 Avaliação Numérica . . . 90
4.4.2 Avaliação Perceptual . . . 93
4.4.2.1 Avaliação Perceptual para o Grupo com Experiência 1 ou 2 . . . 94
4.4.2.2 Avaliação Perceptual para o Grupo com Experiência 4 ou 5 . . . 96
4.5 Resumo do Capítulo . . . 99

5 CONCLUSÕES . . . 101

REFERÊNCIAS . . . 105

APÊNDICES 111

APÊNDICE A – PROTÓTIPO LAP STEEL GUITAR ROBOMUS . 113
A.1 Formato das Mensagens . . . 114

1 Introdução

Os seres humanos encontraram na música uma maneira de se expressar de forma criativa (DELIÈGE; WIGGINS, 2006). Os avanços dos dispositivos tecnológicos, principalmente o surgimento do computador, têm imergido os mesmos em várias áreas, inclusive na música, onde em meados de 1950 o computador passou a ser usado por músicos como ferramenta de criação musical. Essa nova forma de criação musical, unindo dispositivos tecnológicos e música, também despertou interesse dos pesquisadores criando áreas de pesquisa como ubimus (*ubiquitous music*) (KELLER; LAZZARINI; PIMENTA, 2014) e IoMusT (*Internet of Musical Things*) (TURCHET et al., 2018).

Ubimus engloba conceitos de computação ubíqua descrito por Weiser (1991) que possui objetivo no estudo da onipresença dos sistemas tecnológicos visando tornar a interação dos seres humanos com esses dispositivos menos perceptível e mais natural. Assim, ubimus descreve o uso de sistemas ubíquos com agentes humanos e recursos materiais (ferramentas de suporte a criatividade) para criação musical, onde relacionam-se os conceitos de portabilidade, mobilidade, conectividade e disponibilidade (incluindo para não músicos) (PIMENTA et al., 2015). Além disso, surge também a área de pesquisa conhecida como IoMusT (*Internet of Musical Things*) (KELLER; LAZZARINI, 2017; TURCHET et al., 2018), que propicia aplicações de ubimus em infraestrutura de IoT (*Internet of Things*) (ATZORI; IERA; MORABITO, 2010). De acordo com Turchet et al. (2018), IoMusT é descrita como o uso da eletrônica por meio de sensores, atuadores, processamento de dados e conectividade de rede para coleta e troca de dados visando um propósito musical, permitindo assim, a interconectividade entre artistas e/ou a audiência. O desenvolvimento de artefatos, ou "coisas" musicais, para suportar performances de ubimus e IoMusT relaciona-se diretamente com a área de IHC (Interação Humano-Computador), assim, Costalonga, Miletto e Pimenta (2019) apresentam uma discussão sobre o desenvolvimento desses artefatos levando em consideração a musicalidade, uma vez que essa é uma característica humana.

O amplo desenvolvimento dos dispositivos eletrônicos, principalmente em termos de automação, propiciou a criação de robôs para inúmeras funções, inclusive para atividades musicais. Assim, data-se que em 1863 o inventor francês Fourneaux desenvolveu o primeiro piano automático utilizando atuadores pneumáticos chamado de "Pianista" (KAPUR, 2005). Com o passar do tempo, as pesquisas com músicos robóticos foram evoluindo e tornando os robôs cada vez mais inteligentes, podendo estes serem controlados por interfaces de interação ou interagir de forma autônoma em performance solo ou conjunta (BRETAN; WEINBERG, 2016). Além disso, o foco também se ampliou a outras áreas, unindo a área de criação sonora com a de interação gestual, o que levou a criação de robôs antropomórficos

com aparência e gestos análogos aos dos seres humanos (HOFFMAN; WEINBERG, 2010; BRETAN; WEINBERG, 2014). As performances robóticas não possuem nativamente expressividade musical como as performances humanas, e de certa forma são criticadas por esse motivo. No entanto, os parâmetros de expressividade musical como força, velocidade e precisão podem ser modelados a partir das performances humanas e reproduzido pelos robôs, tornando as performances mais próximas de uma apresentação humana (COSTALONGA; PIMENTA; MIRANDA, 2019).

Esses robôs musicais aplicados a ubimus e IoMusT, principalmente os controlados por interfaces de interação, apresentam oportunidades para o desenvolvimento de novas ferramentas e recursos materiais para performances, bem como para proporcionar avanços sobre as barreiras físicas impostas pelos instrumentos acústicos que, em geral, demanda-se horas de treinamento para domínio de um instrumento. Para músicos profissionais estima-se que para se tornar proficiente em um instrumento musical seja necessário em torno de 10.000 horas de prática regular (ERICSSON; KRAMPE; TESCH-RÖMER, 1993). Porém, esse alto esforço respetivo para se tornar especialista em um instrumento pode causar LER (Lesões por Esforço Repetitivo) ou DORT (Distúrbios Osteomusculares Relacionados ao Trabalho), onde estima-se que 75% dos músicos possuem alguma dor proveniente dessas doenças (CASSAPIAN; PELLENZ, 2010). Além disso, a robótica vem se fortalecendo cada vez mais como ferramenta de ensino voltada a programação, como descrito por Zanetti e Bonacin (2014), assim, os robôs musicais apresentam-se como aliados no ensino da robótica, programação, música e outros.

De acordo com o último censo demográfico do Brasil (IBGE, 2010), cerca de 45,6 milhões de pessoas (23,9% da população em 2010) declararam ter pelo menos um tipo de deficiência, seja ela visual, auditiva, motora ou mental/intelectual. Onde as pessoas que se declararam com deficiência motora estão divididas em 8,8 milhões com alguma dificuldade, 3,6 milhões com grande dificuldade e 0,7 milhão com incapacidade. Ubimus possui o conceito de disponibilidade, que também pode ser interpretado em termos de inclusão, onde visa-se o uso de interfaces musicais assistivas por pessoas portadoras de deficiência. Assim, essas interfaces podem controlar os robôs e possibilitar o bem estar dessas pessoas, bem como serem aplicadas em práticas de musicoterapia (GANZA, 1988).

1.1 Motivação

Os instrumentos musicais robóticos podem ser importantes aliados às práticas de ubimus e IoMusT proporcionando expressividade por meio de performances musicais, cativando a criatividade e o aprendizado musical do usuário, bem como a interação social entre usuários. Além disso, poupando horas de treinamento para domínio dos movimentos biomecânicos de se tocar um instrumento e diminuindo a chance do desenvolvimento de

doenças relacionadas a esforços repetitivos, uma vez que as interfaces de controle podem ser projetadas considerando parâmetros voltados a ergonomia.

Com os robôs musicais é possível que essas pessoas possam se expressar musicalmente e ter momentos de lazer para melhorar seu bem estar, além de poderem incrementar processos de reabilitação cognitiva, levando o participante a estimular regiões do cérebro responsáveis pela criatividade musical, e aplicações de musicoterapia.

Já com um viés educacional o ato de projetar, construir e integrar sistemas complexos no desenvolvimento de instrumentos musicais robóticos, motivam e facilitam o processo de resolução de problemas, além de potencializar outras habilidades dos alunos (CAMPOREZ et al., 2015; NETTO et al., 2015). Assim, os robôs, por serem interdisciplinares, podem ser excelentes aliados no ensino de robótica, automação, programação e música.

1.2 Objetivos Gerais

Essa dissertação tem como objetivo geral contribuir para o desenvolvimento de uma plataforma que permita performances musicais robóticas em tempo real. A plataforma, chamada de RoboMus, possui uma arquitetura aberta de hardware e software que possibilita a construção de robôs musicais controlados por humanos através de interfaces de controle. Toda comunicação da plataforma se da por meio de formatos pré-estabelecidos e escalonáveis de mensagens musicais. Além disso, é desejável uma configuração de múltiplos robôs em uma mesma performance musical, assim, estratégias de sincronização são aplicadas por meio de um servidor centralizador de macrossincronização de mensagens musicais e algoritmos de microssincronização embarcados nos próprios robôs.

1.3 Objetivos Específicos

Para os objetivos específicos deste trabalho elenca-se os seguintes tópicos:

- especificar os formatos das mensagens dos robôs com o centralizador de mensagens de modo que não restrinja as futuras gerações de robôs e que permita a configuração de múltiplos robôs;
- definir os formatos básicos das mensagens entre as interfaces de interação e o servidor centralizador;
- elaborar o centralizador de mensagens, o Servidor de Sincronização de Mensagens Musicais (SSMM);
- construir robôs percussivos para teste da plataforma;

- desenvolver estratégia de macrossincronização para sincronizar os relógios dos robôs com o SSMM;

- fornecer estratégia de macrossincronização para compensação do atraso presente nos atuadores mecânicos dos robôs por meio de redes neurais;

- criar algoritmos embarcados nos robôs para sincronização rítmica, em tempo real, com outros robôs ou músicos humanos.

1.4 Metodologia

A construção de uma plataforma para performances musicais robóticas (RoboMus), bem como a prototipagem de instrumentos musicais robóticos, demanda conhecimento sobre o funcionamento de vários algoritmos de sincronização, equipamentos tecnológicos, componentes eletrônicos e etc. Assim, estudar os componentes de forma separada possibilita entender o funcionamento do sistema como um todo. Portanto, esse estudo apresenta uma natureza exploratória.

O desenvolvimento da RoboMus incia-se na escolha do protocolo de comunicação de mensagens musicais, para então definir formatos de mensagens da plataforma. Assim, visa-se a implementação do SSMM e de protótipos robóticos para teste, e posteriormente a aplicação dos algoritmos e realização de testes entre esse dois componentes da RoboMus, onde o principal foco é na macrossincronização. O SSMM aprende o perfil de atraso mecânico do robô por meio de redes neurais e pode compensar tais atrasos iniciando a execução do robô com antecipação. Além disso, a microssincronização é executada por um módulo perceptual embarcado no robô, onde o mesmo "ouve" a performance e se sincroniza ritmicamente, de forma autônoma em tempo real, com o contexto.

Este trabalho apresenta uma avaliação dos atrasos entre os robôs de foma numérica, por meio de algoritmos, e de forma perceptual, desenvolvida por seres humanos. Assim, entende-se que a natureza da pesquisa é quantitativa.

1.5 Estrutura da Dissertação

Esta dissertação está estruturalmente divida em 5 capítulos. O Capítulo 2 apresenta os conceitos necessários para entendimento do desenvolvimento deste trabalho. No Capítulo 3 é apresentado a metodologia do trabalho, ou seja, todo o desenvolvimento da plataforma RoboMus. Os resultados experimentais são apresentados no Capítulo 4, e, por fim, as conclusões e trabalhos futuros são apresentados no Capítulo 5.

1.6 Principais Contribuições

Por meio dos estudos desenvolvidos neste trabalho que descreve sobre sincronização em uma plataforma para performances musicais robóticas, foi possível desenvolver formatos de mensagens de comunicação entre robôs e interfaces, construir instrumentos musicais robóticos e elaborar algoritmos de sincronização entre robôs. Essa investigação científica permitiu a produção dos seguintes artigos em conferencias/*workshops*, estes são:

- CAMPOREZ, H. A. F. et al. Olhar Musical: Uma Proposta de Interface para Expressividade Musical Voltada a Indivíduos com Deficiência Motora. In: *Anais do VIII Workshop de Música Ubíqua (UbiMus)*. São João del Rei - MG: UFSJ, 2018. p. 76–85 (publicado);
- CAMPOREZ, H. A. F. et al. RoboMus: Robotic Musicians Synchronization. In: *Anais do IX Workshop de Música Ubíqua (UbiMus)*. Porto Seguro - BA: 2020 (aceito);
- CAMPOREZ, H. A. F. et al. Features Extraction and Segmentation for an Assistive Musical Interface. In: *Anais do IX Workshop de Música Ubíqua (UbiMus)*. Porto Seguro - BA: 2020 (aceito).

Além desses, tem-se as seguintes produções publicadas em revista, jornal e capítulo de livro:

- CAMPOREZ, H. A. F. et al. RoboMus: Uma Plataforma para Performances Musicais Robóticas. In: KELLER, D.; LIMA, M. H. D. (Ed.). *Aplicações em Música Ubíqua*. 1. ed. São Paulo, Brasil: ANPPOM, 2018. p. 58–93;
- CAMPOREZ, H. A. F. et al. Interface Computacional para Controle Musical Utilizando os Movimentos dos Olhos. *Revista Vórtex*, Curitiba, v. 6, n. 2, p. 1–17, 2018.
- CAMPOREZ, H. A. F. et al. Internet das Coisas Musicais Aplicada a Instrumentos Musicais Robóticos . *Journal of Digital Media & Interaction*, Portugal: Universidade de Aveiro, v. 3, n. 5, p. 89-101, 2020;

E, por fim, o trabalho *RoboMus: Robotic Musicians Synchronization* foi selecionado para uma edição especial em música ubíqua do *MIT Press Journals Computer Music Journal*.

2 Referenciais Teóricos

2.1 Robôs Musicais

Com o avanço da tecnologia e, principalmente, o da computação e dos componentes eletromecânicos, tornou-se possível a construção de instrumentos musicais robóticos (KAPUR, 2005). A área de pesquisa em robôs musicais possui uma vasta conexão com outras áreas como computação, engenharias, música, psicologia, Interação Humano-Robô (IHR), ubimus e IoMusT, tanto na construção física dos robôs quanto no desenvolvimento de algoritmos para performances musicais, controle e expressividade (BRETAN; WEINBERG, 2016). A abordagem de construção dos robôs abrange vários tipos como os antropomórficos, que buscam um *design* estético semelhante ao ser humano e, em geral, disponibilizam interação gestual e sonora, bem como robôs não antropomórficos que possuem um foco maior na criação sonora (BRETAN; WEINBERG, 2016). Além disso, os robôs podem ser categorizados como instrumentos de corda, de percussão, de sopro e etc, como descrito por Bretan e Weinberg (2016) e por Kapur (2005).

Baginsky (199-?b) desenvolveu uma banda de rock robótica autônoma controlada por inteligencia artificial chamada The Three Sirens, essa banda era composta principalmente por 3 instrumentos musicais robóticos utilizando uma guitarra elétrica com *slide* (Aglaopheme), um contra-baixo (Peisinoe) e uma bateria (Thelxiepeia). Aglaopheme (Figura 1 (a)) é uma guitarra com 6 cordas possuindo o dispositivo de mão direita (guitarrista destro) composto por 6 solenoides para abafamento das cordas e também para bater e gerar sons, além disso há uma palheta conectada a um motor linear para tocar as cordas e um solenoide para movimentar a palheta e evitar que ela encoste em uma corda não desejada. Para o dispositivo de mão esquerda foi usado um motor CC (Corrente Contínua) para movimentar o *slide* ao longo do braço da guitarra, onde utilizou-se um solenoide para encostar ou desencostar o *slide* nas cordas. Essa guitarra com *slide* possui ainda dois grandes solenoides para controlar o movimento da guitarra como um todo, como se fossem duas pernas, além disso esse movimento aciona um pedal de efeito *wah-wah*[1]. Peisinoe (Figura 1 (b)) é um contra-baixo de apenas uma corda tocado por um arco motorizado, onde a mudança de nota foi proporcionada por um forte solenoide usado para o controle da tensão da corda, ou seja, mudando a afinação da mesma. A percussão é comandada pelo Thelxiepeia (Figura 1 (c)) utilizando um rototon, nesse instrumento a mudança de nota é feita rotacionando o mesmo. Assim, um motor foi usado para gira-lo e uma baqueta automatizada foi usada para bater no instrumento.

[1] Pedal de efeito que visa alterar o formato de onda do instrumento buscando modificar sua estética sonora, esse efeito atenua algumas frequências de acordo como é pressionado.

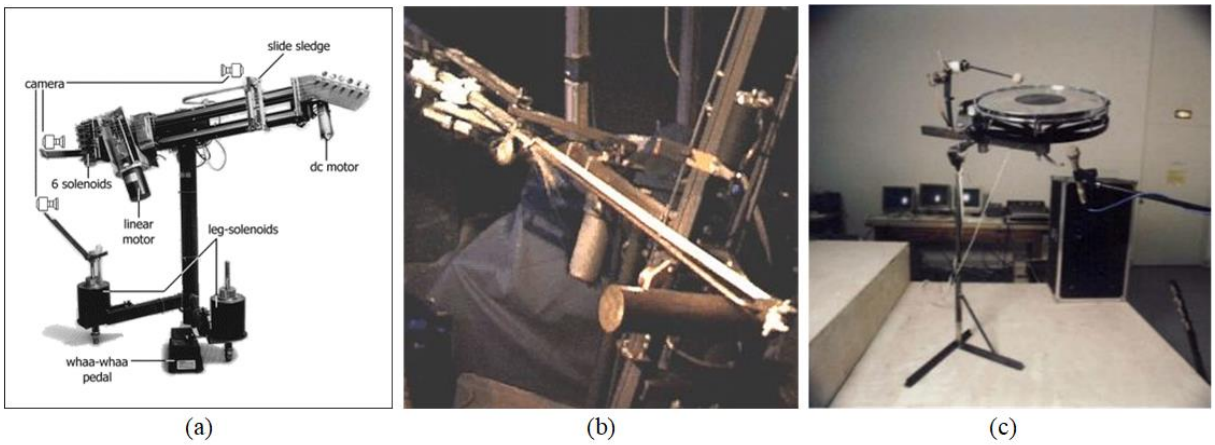

Figura 1 – Banda musical robótica The Three Sirens: (a) guitarra elétrica robótica com *slide* (Aglaopheme), (b) contra-baixo robótico (Peisinoe) e (c) bateria robótica (Thelxiepeia).

Fonte: (a) (BAGINSKY, 199-?a), (b) e (C) (BAGINSKY, 199-?b)

Jordà (2002) desenvolveu um guitarra elétrica robótica (Figura 2 (a)), com 6 cordas e utilizou 72 solenoides para cobrir as 12 casas, representando a mão esquerda de um guitarrista destro. Porém essa guitarra não possui dispositivos para reproduzir a função da mão direta, assim, o som da corda era gerado no momento em que o solenoide apertava a corda contra o braço da guitarra, utilizando a técnica de *hammer on*[2]. Além disso, todo o controle do robô foi feito utilizando o protocolo de comunicação musical MIDI (*Musical Instrument Digital Interface*)[3]. Outro robô guitarrista é o LEMUR GuitarBot (SINGER; LARKE; BIANCIARDI, 2003; SINGER; FEDDERSEN, 2004), uma espécie de *slide guitar* controlada por mensagens MIDI com 4 cordas. Como mostra a Figura 2 (b), cada corda possui dispositivos de controle independentes (Figura 2 (c)). Assim, para tocar uma corda foram fixadas 4 palhetas em um motor CC com um potenciômetro para controle de posição. Também um solenoide foi instalado para abafar a corda e proporcionar interromper o som completamente ou tocar uma nota abafada. O movimento da ponte deslisante sobre a corda foi controlado por um motor CC e um potenciômetro para controle do posicionamento, já na ponte deslizante foi acoplado um sistema para permitir o deslisamento encostando ou não na corda.

Na classe de instrumentos musicais robóticos percussivos pode-se destacar o Haile (Figura 3 (a)), um robô antropomórfico desenvolvido para tocar um instrumento percussivo multiusuário conhecido como *Native American Pow-wow*, esse instrumento foi escolhido para suportar a premissa de ser um projeto para colaboração entre usuários e o robô (Figura 3 (b)). Assim, o robô Haile "ouve" os sons dos outros participantes, faz uma

[2] Técnica onde o guitarrista não toca a corda com a mão direita, ele apenas pressiona a corda com a mão esquerda utilizando certa força e velocidade, como se estivesse "martelando" a corda com o dedo.
[3] MIDI, um protocolo de mensagens musical criado na década de 1980 (MOOG, 1986)

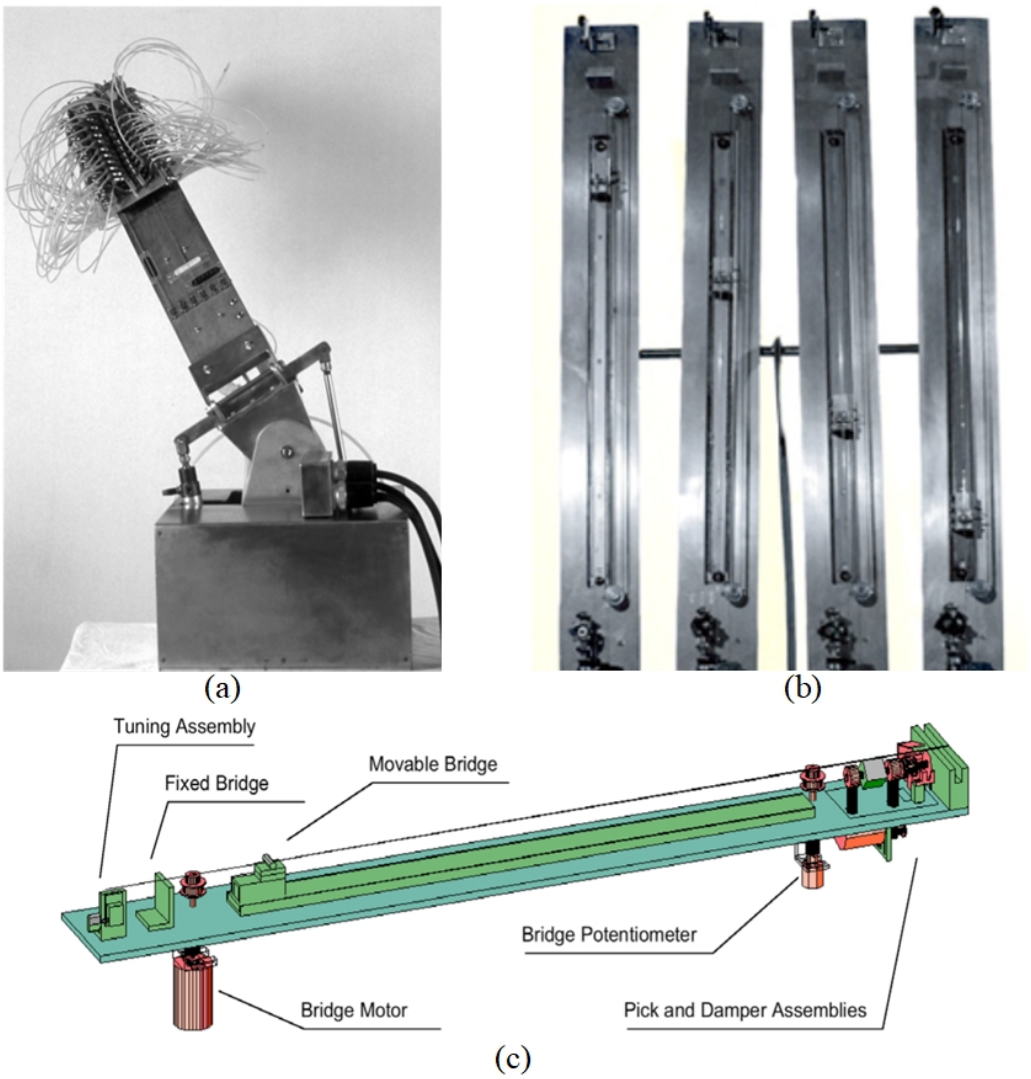

Figura 2 – Instrumentos musicais robóticos: (a) guitarra elétrica robótica, (b) guitarra elétrica com *slide* (LEMUR GuitarBot) e (c) dispositivo de slide para cada corda do GuitarBot.

Fonte: (a) (JORDà, 2002), (b) (SINGER; FEDDERSEN, 2004) e (c) (SINGER; LARKE; BIANCIARDI, 2003).

análise perceptual em tempo real e os acompanha musicalmente. O robô possui dois braços mecânicos, onde o direito (Figura 3 (c)) foi projetado para tocar notas rápidas e o esquerdo para produzir movimentos de interação mais amplos e visíveis. Para tocar o tambor foi implementado um solenoide a fim de controlar o movimento vertical da baqueta. Já para mudar o tom da batida, ou seja, mudar a posição onde a baqueta bate, foi utilizado um motor e um potenciômetro para controle, movimentando a baqueta para frente e para trás.

O Shimon (HOFFMAN; WEINBERG, 2010) é um robô antropomórfico capaz de tocar marimba de forma autônoma. Esse robô, demostrado na Figura 4, possui braços

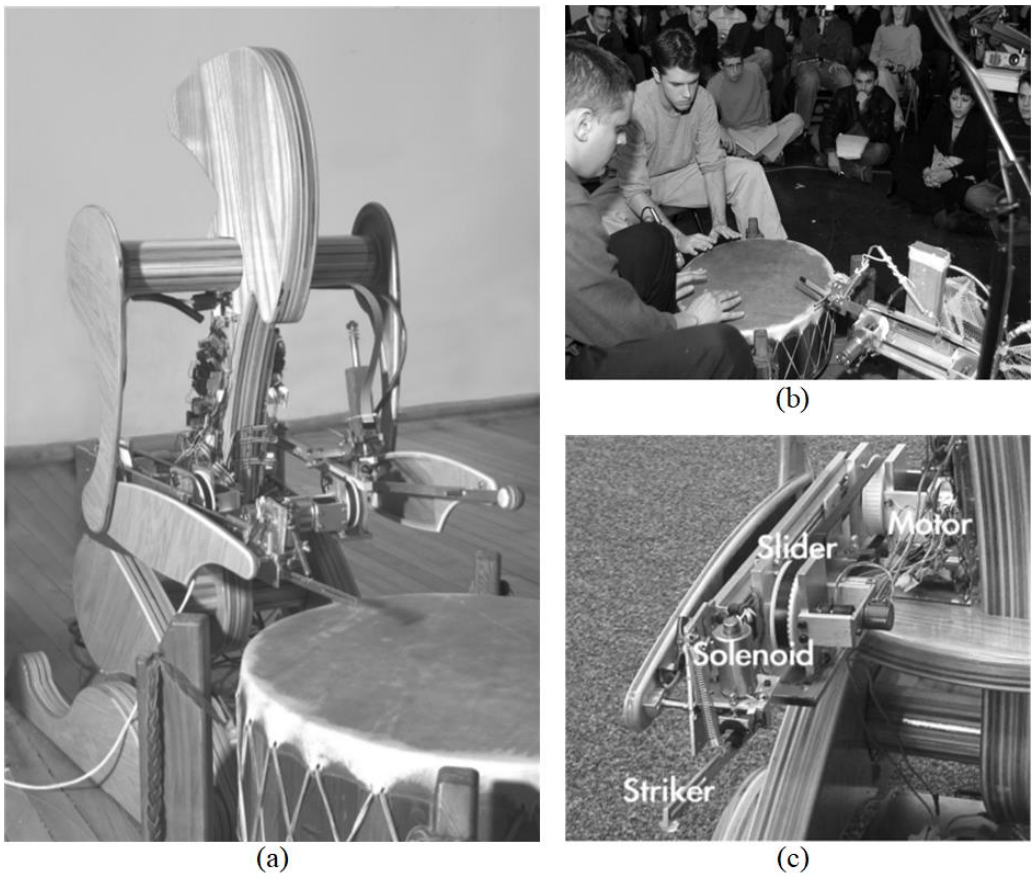

Figura 3 – Robô Haile: (a) estrutura do robô, (b) performance colaborativa com o robô e (c) braço direito do robô

Fonte: (WEINBERG; DRISCOLL, 2006)

mecânicos movimentados paralelamente ao marimba por um motor linear, cobrindo toda a extensão do instrumento. Cada braço do robô contém duas solenoides rotacionais para movimentar as baquetas e tocar o instrumento. Além disso, esse robô possui mais características antropomórficas, possuindo uma cabeça não humanoide para interação gestual com os participantes da performance e da audiência que segundo os autores proporciona maior engajamento, além de facilitar a sincronização dos músicos para com o robô, uma vez que em uma performance musical feita por humanos o som não é a única forma de interação, existe uma comunicação visual que auxilia na sincronização (GOEBL; PALMER, 2009).

2.2 Ubimus

Datado de meados de 1950, o computador, uma recente tecnologia na época, despertou interesse dos músicos para atividades de composição musical. Com o passar do tempo e com a popularização desse dispositivo, houve uma grande evolução no campo da

Figura 4 – Marimba robótico Shimon

Fonte: (HOFFMAN; WEINBERG, 2010)

Interação Homem-Computador (IHC) de modo que o computador passou a ser utilizado por pessoas não especializadas. No entanto, certamente o fazer musical não figurava entre as principais atividades suportadas pelo computador, tornando a evolução de instrumentos musicais digitais um processo mais lento se comparado a outros produtos da computação, como por exemplo, editores de texto. Atualmente, computadores (em todas as suas formas modernas) estão envolvidos em praticamente todas as atividades conhecidas pelos seres humanos, incluindo a música. Essa ampla presença dos dispositivos computacionais na vida cotidiana remete ao conceito de computação ubíqua (ou ubicomp – *ubiquitous computing*) descrito por Weiser (1991), que tem como objetivo tornar a interação humano-computador "invisível", ou menos perceptível e mais natural ao ser humano.

Bem como o computador, e a tecnologia em geral, despertou interesse em músicos amadores e profissionais para criação musical, a conjunção de sistemas tecnológicos com a música também despertou interesse dos pesquisadores, proporcionando o surgimento de uma área de pesquisa chamada ubimus (*ubiquitous music*) (KELLER; LAZZARINI; PIMENTA, 2014). Esse campo direciona suas pesquisas para o uso de sistemas onipresentes de agentes humanos e recursos materiais, os quais proporcionam atividades musicais através de ferramentas de apoio à criatividade. Na prática, ubimus pode ser considerada música, ou atividade musical, apoiada por conceitos de computação ubíqua e suas tecnologias. Existe ainda uma relação com conceitos como: portabilidade, mobilidade, conectividade e disponibilidade (incluindo para não músicos) (PIMENTA et al., 2015). Assim, ubimus adota, por exemplo, dispositivos de comunicação móvel presente no cotidiano das pessoas, como smartphone, tablets e etc, para apoiar as atividades musicais (KELLER et al., 2011).

O desenvolvimento de sistemas de ubimus trás consigo diversos fatores para alcançar os conceitos supracitados, segundo Pimenta et al. (2015), dentro de ubimus, a criação musical pode ser definida como um conjunto de atividades envolvendo interações agente-agente (interação social) e agente-objeto (interação natural). Os autores também definem

três metáforas para um *framework* conceitual, estas são:

- metáfora do corpo: aplicada em contextos de performances multimídia e instalações interativas, onde a interface de exploração interativa do participante é o corpo, que por meio de um sistema de sensores pode-se associar ações do público às luzes, *slides*, projeções de áudio e vídeo e etc;
- metáfora da marcação temporal: definida como um conjunto de elementos virtuais organizados sobre um cronograma marcado, onde as operações sônicas se acoplam com as conceituais;
- metáfora da marcação espacial: visa estabelecer um canal de interação natural entre os elementos virtuais e/ou físicos, estabelecendo forte relação entre os elementos do contexto (objetos físicos, lugares, situações e etc).

2.3 Internet of Musical Things

Uma "coisa" musical (*musical thing*), de uma perspectiva da ciência da computação, pode ser definida como um dispositivo computacional com propósitos musicais. Esses dispositivos devem ser capazes de detectar, adquirir, processar, atuar e trocar informações visando alcançar seus propósitos (TURCHET et al., 2018). Como complemento, IoMusT (*Internet of Musical Things*) também pode ser descrita como a conjunção de interfaces, protocolos e representações de informações musicais com objetivo de fornecer serviços e aplicações para atividades artísticas baseadas em interações entre humanos e coisas musicais ou entre coisas musicais, seja no mundo físico ou digital. Assim, IoMusT pode ser retratada como uma interseção de IoT e sistemas musicais.

Os ecossistemas digitais trazem consigo uma perspectiva de um ambiente colaborativo (BOLEY; CHANG, 2007). Segundo Turchet et al. (2018), um ecossistema de IoMusT pode ser constituído por: usuários envolvidos em atividades musicais, provedores de informações e serviços, e padrões. Os autores também definem três componentes para um ecossistema de IoMusT dentro de uma perspectiva tecnológica, estes são:

- coisas musicais: representa um novo conjunto de dispositivos musicais com conexão à Internet que promete uma grande transformação na maneira de interação do usuário com atividades musicais. Essas coisas musicais herdam diversas características de IoT como sensores, atuadores, conectividade, processamento de dados e etc;
- conectividade: nos ecossistemas de IoMusT considera-se comunicação sem fio multidirecional, podendo ser remota ou localizada. Toda essa interconectividade é alcançada por meio das tecnologias de hardware e software, e, também, por padrões e protocolos de gerenciamento voltados a IoMusT. Algumas performances artísticas possuem

como requisito a comunicação em tempo real, assim, a rede deve fornecer baixa latência, alta confiabilidade, alta qualidade e sincronização entre os dispositivos conectados;

- aplicações e serviços: os tipos de aplicações e serviços voltados a IoMusT são variados, onde o público-alvo pode ser músicos, engenheiros de áudio, membros da audiência, entre outros. Os *serviços* podem ser do tipo interativo ou não, como por exemplo, a conectividade do *smartphone* ao instrumento inteligente (explicado abaixo); a conexão do instrumento inteligente com redes sociais; os serviços de análises para conteúdo criativo; os serviços de mapeamento dos dados e o controle de coisas musicais. Já as *aplicações* apresentam a camada mais alta de interação com o usuário. Como exemplo pode-se citar o aumento da experiência dos membros da audiência em um concerto baseado na interação com coisas musicais. Para os músicos, um exemplo é possibilidade de ensaios remotos. Para engenheiros de áudio encontram-se os estúdios de produção inteligentes. Para estudantes e professores, cita-se a possibilidade de seções de ensino remotas propiciadas pelos instrumentos inteligentes.

Segundo Turchet et al. (2018), a área de IoMusT envolve diversas áreas como: performances musicais em rede; performances interativas; dispositivos vestíveis; realidade aumentada e etc. Como exemplos de IoMusT, pode-se citar:

- instalações multimídia interativas: visam proporcionar a interação entre usuários e dispositivos multimídia. Um exemplo de IoMusT é a instalação multimídia "Árvore das Memórias" (NETTO et al., 2015). Inicialmente, a instalação buscava facilitar a aproximação dos estudantes do Campus de São Mateus da Universidade Federal do Espírito Santo através do uso de um ponto focal de interação. Aproveitando o comportamento comum de comunicação por meio de aplicativos de mensagens instantâneas, usou-se uma árvore estrategicamente posiciona defronte à secretaria geral dos estudantes como um amigo virtual do corpo discente. A comunicação com a árvore, travestida em traje natalino, se dava através de mensagens de áudio do WhatsApp. Assim, a árvore transmitia as mensagens por meio de alto falantes escondidos em enfeites natalinos para todos que estivessem sob sua copa. A instalação ficou em exibição por duas semanas. Após análise, notou-se que muitas mensagens enviadas eram trechos de músicas natalinas. Ainda, várias mensagens eram oriundas de outros países, indo além da concepção inicial em um movimento de apropriação da mesma pelo público. Desta forma, em uma segunda versão da instalação, buscou-se facilitar o seu uso com instrumento musical sugerindo a audiência o envio de mensagens ligadas a um determinado tema. Isso foi feito através de uma página no Facebook (@MemoriesTree) da própria "árvore" que interagia com a audiência remotamente. Ainda, a árvore foi pensada para trocar informações (e até a identidade

visual) com instalações irmãs espalhadas em outras localidades, como em uma floresta[4];

- instrumentos inteligentes: também conhecido como instrumentos aumentados, são criados a partir da modificação de instrumentos convencionais utilizando sensores, atuadores e sistemas embarcados para proporcionar ao usuário novas formas de controle e expressividade musical. Além disso, alguns ainda possuem inteligência computacional, mecanismos de síntese e processamento de som, conectividade sem fio e um sistema de interação com o usuário (CHRIST et al., 2007). Como exemplos, pode-se citar o *Sensus Smart Guitar* (TURCHET; MCPHERSON; FISCHIONE, 2016), o *Smart Cajón* (TURCHET; MCPHERSON; BARTHET, 2018) e o *Smart Mandolin* (TURCHET, 2018);

- dispositivos vestíveis: são dispositivos alocados ao corpo do usuário (ex. *smartwatch*) com objetivo de captar informações fisiológicas como temperatura corporal, frequência cardíaca e captação de movimentos, onde essas informações podem ser usadas na performance. Além disso, podem fornecer *feedback* háptico ao usuário de acordo com um contexto da atividade musical;

- robôs musicais: os instrumentos musicais robóticos podem ser robôs virtuais ou físicos. Os físicos fazem uso sensores, atuadores, componentes eletrônicos, ou seja, coisas musicais. Na classe de robôs controlados por seres humanos, em geral, demanda-se de conexão, representando o componente conectividade e interfaces de controle (aplicações e serviços). Assim, as performances musicais robóticas podem ser classificadas como atividades de IoMusT. Exemplos de robôs musicais são demonstrado na Seção 2.1.

2.4 Processamento Digital de Sinais de Áudio

2.4.1 Detecção de Picos em Sinais

A detecção de picos em sinais visa encontrar os pontos de máximo de um sinal. Na música esses pontos podem ser bons indicadores do momento em que um som foi iniciado (uma batida de bateria, tocar uma corda e etc), sendo usando como detector de batidas (BÖCK; KREBS; SCHEDL, 2012). Assim, um algoritmo para detecção de picos é descrito e implementado pela biblioteca Librosa[5] com inspiração no algoritmo descrito por Böck, Krebs e Schedl (2012). Seja um sinal digital $x[n]$ e n a n-ésima amostra selecionada, esse algoritmo utiliza três condições para determinar se a amostra n é um pico, estas são:

[4] Vídeos da primeira versão da instalação podem ser acessados em <https://youtu.be/B_uE0HLupy4> e <https://youtu.be/CoslpB7PKl4>

[5] <https://librosa.github.io/librosa/generated/librosa.util.peak_pick.html>

1. $x[n] = max(x[n - pre_max : n + post_max])$;
2. $x[n] \geq mean(x[n - pre_avg : n + post_avg]) + \Delta_m$;
3. $n - previous_n > wait$.

A primeira condição determina que o valor da amostra n ($x[n]$) deve ser o maior valor (denotado pela função $max()$) em um intervalo de amostras definido pelas variáveis pre_max e $post_max$, onde $(n - pre_max)$ e $(n + post_max)$ definem o limite inferior e superior, respectivamente. A segunda condição determina que o $x[n]$ deve ser maior ou igual a média (descrito pela função $mean()$) de um intervalo em torno da amostra n somado de um deslocamento (Δ_m), esse intervalo de amostras é definido por pre_avg e $post_avg$, onde o limite inferior é definido por $(n - pre_avg)$ e o superior por $(n + post_avg)$. Por fim, a terceira condição determina uma distância mínima entre picos, assim, a distância entre o pico anterior e a amostra em avaliação n deve ser maior que a variável $wait$.

2.4.2 Transformada de Fourier de Curto Termo

A transformada de Fourier de curto termo (*Short-Term Fourier Transform* - STFT) (KEHTARNAVAZ, 2008), também conhecida como transformada de curto tempo (*Short-Time Fourier Transform*), é proveniente da transformada de Fourier (*Fourier Transform* - FT). Na STFT a FT é aplica em intervalos curtos de tempo, ou janelas, com objetivo de extrair as informações de frequência desse pequeno intervalo e não de todo o sinal, assim, é possível obter a variação de frequência ao longo do tempo de um sinal, vale ressaltar que pode haver sobreposição entre janelas. A fórmula da STFT é dada por:

$$X_{STFT}[d, k] = \sum_{n=0}^{L-1} x[n]g[n - d]e^{-j2\pi kn/L}(k = 0, ...N - 1), \tag{2.1}$$

onde $x[n]$ representa o sinal de áudio, $g[n]$ representa a função da janela com L pontos, d é o deslocamento da janela que também determina o *frame* e k são os N pontos da transformada. O STFT também pode ser interpretada como a transformada de Fourier do produto $x[n]g[n - d]$. A Figura 5 ilustra o processo descrito acima, mostrando o deslocamento da janela (no modelo de janela Hamming (Rabiner; Schafer, 2007)) ao longo do tempo.

2.4.3 Espectrograma Mel

A real percepção humana é mais sensitiva para frequências graves, assim o ser humano tende a ser capaz de discernir, mais facilmente, pequenas mudanças em frequências mais graves do que em mais agudas. Portanto, a escala Mel faz uma representação aproximada da real percepção humana (STEVENS; VOLKMANN, 1940), tornando o

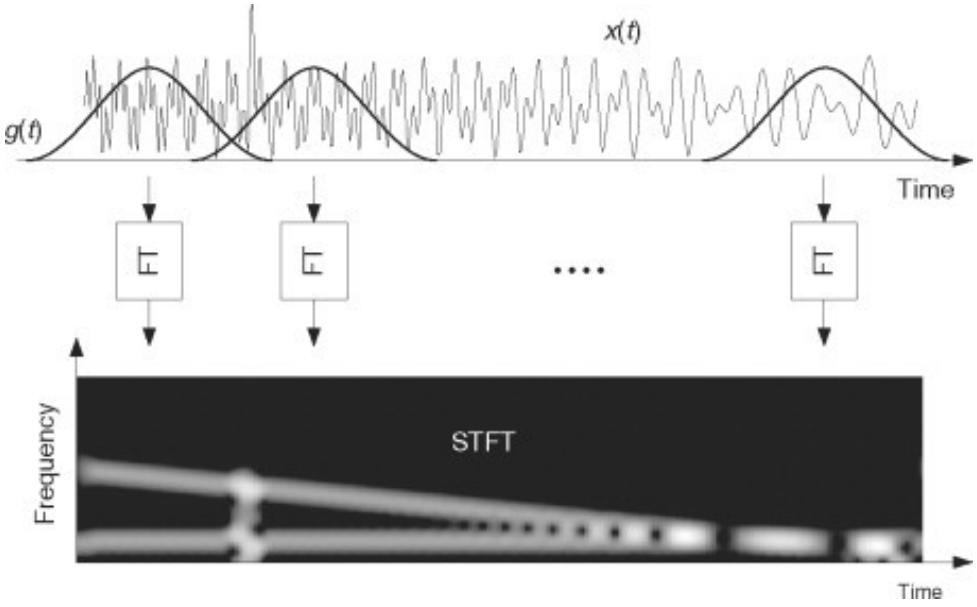

Figura 5 – Exemplo de aplicação da STFT em um sinal.

Fonte: (KEHTARNAVAZ, 2008)

processamento de sinal mais próximo da interpretação humana. A fórmula da escala Mel pode ser definida como:

$$B(f) = 1125 \ln(1 + f/700), \tag{2.2}$$

onde f representa a frequência em Hertz e a função inversa é dada por: $B^{-1} = 700(exp(\frac{b}{1125}) - 1)$, onde b é a frequência em Mel.

Para gerar um espectrograma Mel (HUANG et al., 2001), primeiro é preciso aplicar um banco de filtros Mel. Esse banco é composto por M filtros triangulares, também chamado de bandas Mel, espaçados igualmente entre a maior e a menor frequência na escala Mel, onde cada filtro inicia na metade do anterior (vide Figura 6)[6]. Os filtros podem ser calculados utilizando as seguintes fórmulas:

$$H_m[f] = \begin{cases} 0 & f < F[m-1] \\ \frac{(f - F[m-1])}{(F[m] - F[m-1])} & F[m-1] \leq f \leq F[m] \\ \frac{(F[m+1] - f)}{(F[m+1] - F[m])} & F[m] \leq f \leq F[m+1] \\ 0 & f > F[m+1] \end{cases} \tag{2.3}$$

onde f é a frequência em Hz, m é o número da banda Mel pertencendo ao intervalo de valores de $(0, 1, ..., M)$ e a função F[m] é dado por:

$$F[m] = \left(\frac{N}{F_s}\right) B^{-1} \left(B(f_l) + m \frac{B(f_h) - B(f_l)}{M+1}\right), \tag{2.4}$$

tendo N como o tamanho (número de pontos) da transformada descrita na Subseção anterior, F_s como a taxa de amostragem do sinal em que o banco de filtro será aplicado,

[6] Note que na escala em Hertz os filtros aumentam de tamanho.

f_l como a menor frequência em Hz do filtro e f_h como a maior frequência em Hz. Essa fórmula visa determinar os pontos do triangulo de cada banda, o ponto inicial (F[m-1]), o ponto médio (F[m]) e o ponto final (F[m+1]), como pode ser observado na Figura 6.

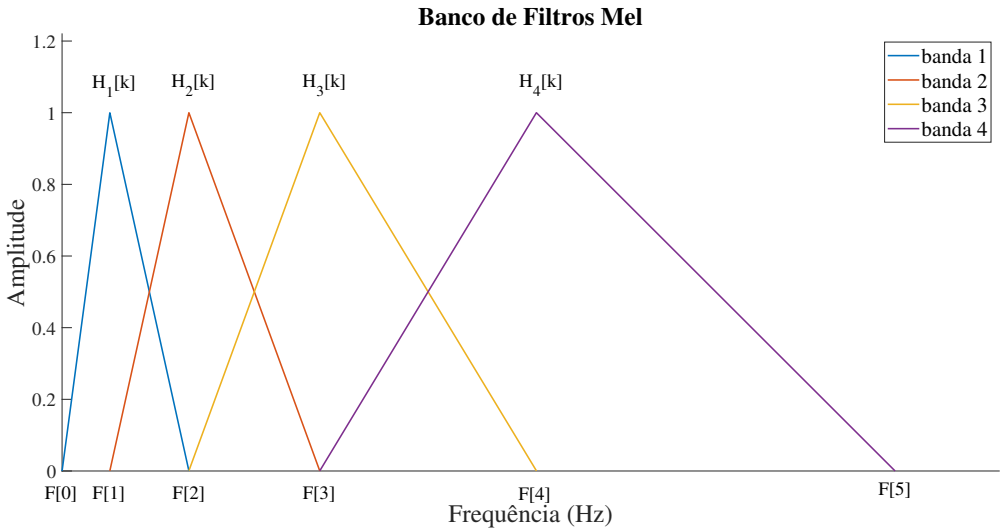

Figura 6 – Filtros triangulares espaçados na frequência

Fonte: Autor.

Cada banda Mel é formada pela soma dos valores após a aplicação do seu respectivo filtro. Assim, o espectrograma Mel é obtido através da aplicação do banco de filtros para cada *frame* gerado pela STFT. A Figura 7 ilustra o processo para cada *frame*. A fórmula é descrita a seguir:

$$MS[m,d] = \sum_{k=0}^{N-1} X_{STFT}[d,k] H_m[vf(k)], \tag{2.5}$$

onde $m = (1, ..., M)$, M é número total de bandas e vf é a função que converte o ponto k da STFT em frequência em Hz.

2.4.4 Sistema de Rastreamento de Batidas

O rastreador de batidas (*beat tracker*) desenvolvido por Ellis (2007) tem por objetivo encontrar as batidas, ou pulsos musicais[7], de uma música e também sua velocidade, que geralmente é expressada em BPM (Batidas Por Minuto). A informação de tempo musical é de suma importância para o entendimento e construção do ritmo[8], sendo utilizado para sincronização entre instrumentistas. O sistema de rastreamento desenvolvido tem como entrada um BPM global estimado e *Onset Strength Envelope* (OSE) que serão descritos nas próximas Subseções.

[7] Geralmente, quando um ouvinte bate palma, ou o pé, junto com a música representa o pulso da música (BENWARD; SAKER, 2008).

[8] O ritmo pode ser brevemente descrito como um conjunto sons e pausas organizado em padrões (BENWARD; SAKER, 2008).

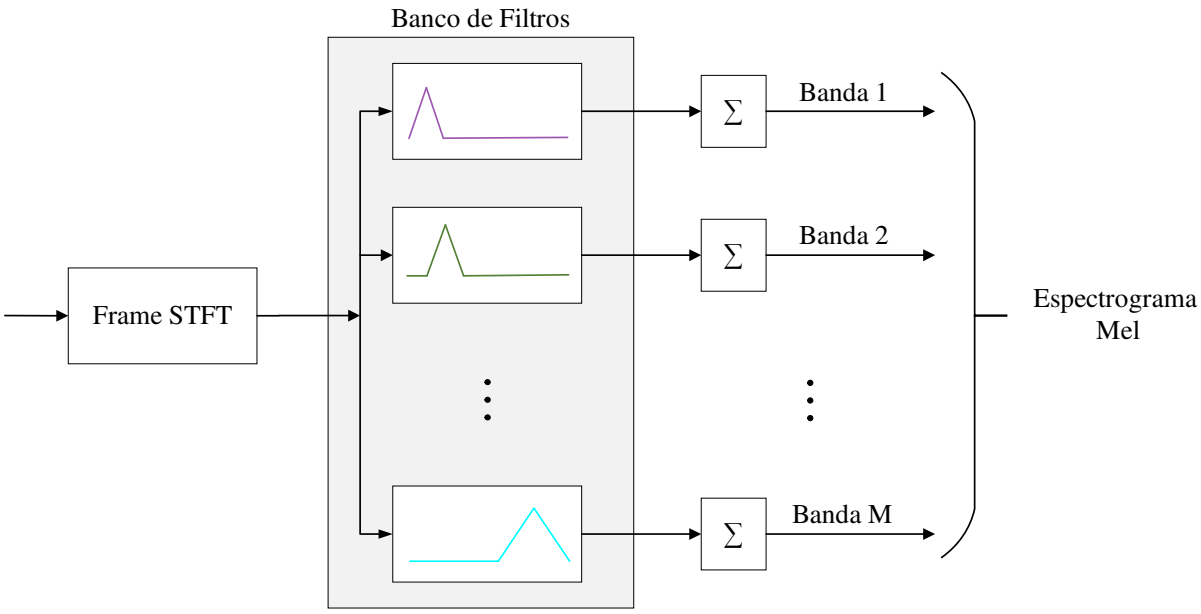

Figura 7 – Esquemático para geração do espectrograma Mel

Fonte: Autor.

2.4.4.1 Onset Strength Envelope

O *onset strength envelope* representa o fluxo espectral de um sinal, demonstrando as mudanças da potência do sinal ao longo do tempo. No sistema desenvolvido por Ellis (2007), primeiro converte-se o sinal de áudio para a taxa de amostragem de 8 kHz e aplica-se a STFT utilizando uma janela de 32 ms com 4 ms de deslocamento. Após esse processo o espectrograma é convertido em um espectrograma Mel com 40 bandas. Contudo, o cálculo do OSE é determinado pela soma das diferenças entre uma banda do espectrograma e a banda antecessora, onde os valores negativos são convertidos em zero (retificador de meia onda). A seguinte fórmula define o fluxo espectral (BöCK; WIDMER, 2013):

$$O[t] = SF[d] = \sum_{m=2}^{M} H[|MS[m,d]| - |MS[m-1,d]|], \tag{2.6}$$

onde $H(x) = \frac{x+|x|}{2}$ representa o retificador de meia onda, MS representa o espectrograma Mel, m o número da banda.

2.4.4.2 BPM Global

O BPM global de uma música pode ser obtido através do uso da autocorrelação do sinal de áudio (ELLIS, 2007). A autocorrelação consiste no produto entre um sinal e uma versão deslocada no tempo do mesmo. Assim, a autocorrelação aplicada ao OSE $O(t)$[9] pode revelar regularidades e estruturas periódicas, possibilitando estimar o BPM de uma

[9] Este trabalho usa sinais discretos, porém, o autor Ellis (2007) descreve as fórmulas utilizando a notação de sinais contínuos (parêntesis). Assim, segue-se a notação do autor.

música. Contudo, sinais periódicos apresentam valores de alta correlação para múltiplos inteiros de seu período fundamental, dificultando a escolha dos picos de autocorrelação. Entretanto, a percepção humana é tendenciosa a uma velocidade de 120 BPM. Assim, o autor aplicou uma janela perceptual no sinal de autocorrelação atenuando as amplitudes dos picos periódicos longes de 120 BPM. A fórmula de autocorrelação com a aplicação da janela é definida como:

$$TPS(\tau) = W(\tau) \sum_t O(t)O(t-\tau) \tag{2.7}$$

onde $W(\tau)$ é uma janela Gaussian definida por:

$$W(\tau) = \exp\left\{-\frac{1}{2}\left(\frac{\log_2 \tau/\tau_0}{\sigma_\tau}\right)^2\right\}. \tag{2.8}$$

onde τ_0 é o tempo central da janela e σ_τ controla da curva de ponderação. O Autor demonstrou bons resultado para $\tau_0 = 0,5s$ (120 BPM) e $\sigma_\tau = 1,4$. O período t é então estimado pelo maior valor para $TPS(t)$, assim, o período é convertido em BPM pela fórmula $\frac{60}{t}$. Na Figura 8 demonstra-se o sinal OSE, a autocorrelação do OSE e a aplicação da janela ao sinal de autocorrelação.

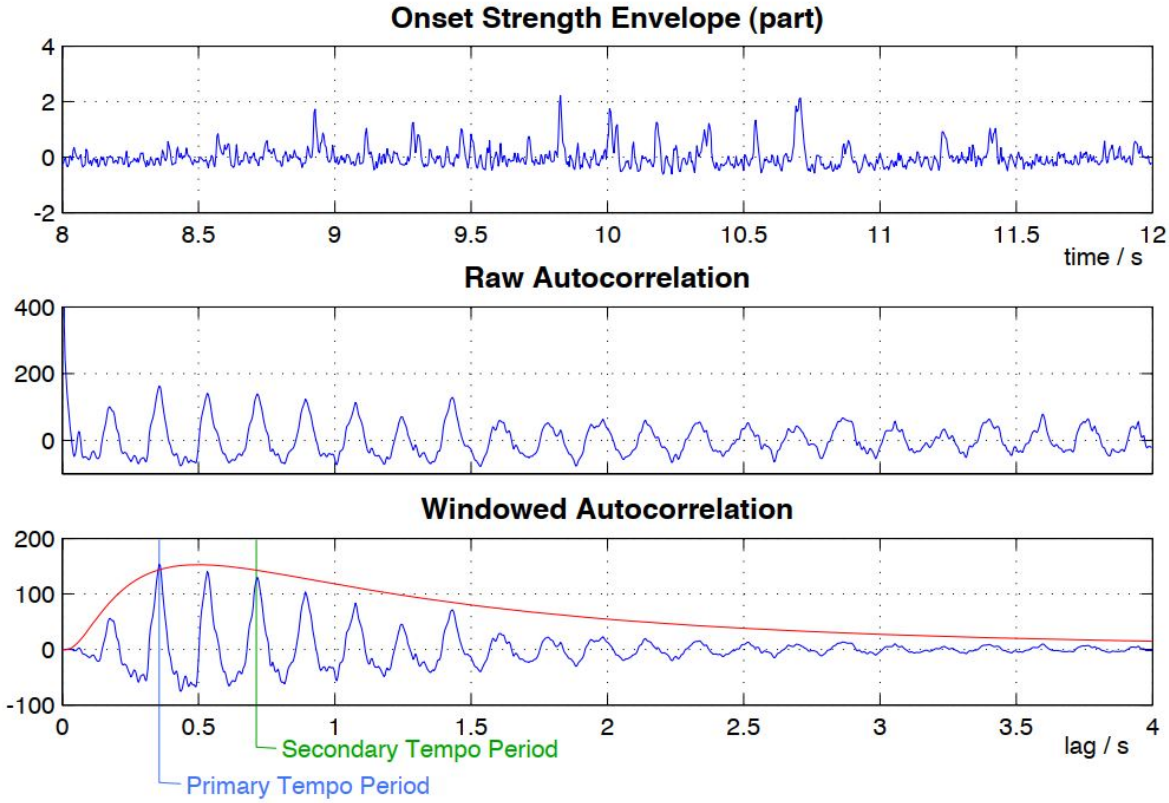

Figura 8 – Gráficos do OSE, da autocorrelação do OSE e aplicação da janela no gráfico da autocorrelação.

Fonte: (ELLIS, 2007)

2.4.4.3 Formulação do Rastreador de Batidas

O algoritmo de rastreamento de batidas (ELLIS, 2007) assume como base uma velocidade previamente conhecida e constante da música, assim o BPM é fornecido pelo algoritmo supracitado. Contudo, o algoritmo de rastreamento tem o objetivo de gerar uma sequência que corresponde aos pontos onde ocorrem os pulsos musicais. Assim, para alcançar esses objetivos o autor define uma função objetivo dada por:

$$C\left(\{t_i\}\right) = \sum_{i=1}^{N_b} O\left(t_i\right) + \alpha \sum_{i=2}^{N_b} F\left(t_i - t_{i-1}, \tau_p\right), \tag{2.9}$$

onde $\{t_i\}$ representa uma sequência de N_b pulsos, $O(t)$ o OSE proveniente do sinal de áudio e α o fator de balanço entre os termos. Já $F(\Delta t, \tau_p)$ é a função que mede a consistência entre o intervalo Δt entre dois pulsos e o intervalo ideal τ_p (retirado do BPM global). $F(\Delta t, \tau_p)$ é definido pela fórmula:

$$F(\Delta t, \tau) = -\left(\log \frac{\Delta t}{\tau}\right)^2. \tag{2.10}$$

Observe que o máximo valor da função é zero quando $\Delta t = \tau$.

A função objetivo pode ser redefinida de forma recursiva para calcular a melhor sequência até um tempo final t, assim, não há necessidade de usar a sequência de pulsos t_i como entrada (Equação 2.9). O Autor define a função recursiva como:

$$C^*(t) = O(t) + \max_{\tau=0\ldots t} \left\{\alpha F\left(t - \tau, \tau_p\right) + C^*(\tau)\right\}. \tag{2.11}$$

Enquanto o cálculo da equação acima é desenvolvido, o pulso predecessor com maior valor do OSE é armazenada, esse processo é dado pela fórmula:

$$P^*(t) = \arg\max_{\tau=0\ldots t} \left\{\alpha F\left(t - \tau, \tau_p\right) + C^*(\tau)\right\}. \tag{2.12}$$

Contudo, na pratica é preciso determinar o escopo de busca, para isso limita-se a variável τ, o Autor determinou $\tau = t - 2\tau_p \ldots t - \frac{\tau_p}{2}$. Por fim, após todo processo executado, seleciona-se o maior valor de C^* que provavelmente representa o último pulso no instante t_n, onde n é o número total de pulsos. Os pulsos anteriores são encontrados através da função P^* onde $t_{N-1} = P^*(t_N)$, e assim o processo é feito de forma regressiva até o início do sinal, e, por fim, determina-se a sequência $\{t_i\}^*$ de batidas.

2.4.5 Extração da Assinatura de Tempo

A assinatura de tempo, também chamada de fórmula do compasso[10], é de suma importância para o entendimento da estruturação temporal de uma música, pois com ela

[10] O compasso define a divisão da música em sequências regulares de tempos. Já a fórmula do compasso (ou assinatura de tempo) define o tamanho de um compasso. Geralmente, ela é representada por

é possível entender as divisões de tempo da música. Coyle e Gainza (2007) desenvolveram um método para detecção da fórmula de compasso, onde utiliza-se como peça chave as estruturas de repetição de compasso contidas na maioria das músicas. Assim, esse método depende apenas das repetições e não da presença de instrumentos percussivos ou com batidas extremamente fortes.

O método descrito por Coyle e Gainza (2007) necessita da informação de tempo da música (BPM) como entrada (que pode ser fornecido pelo algoritmo supracitado), seguido do espectrograma do sinal que é gerado através da STFT. O espectrograma é calculado considerando uma janela de Hamming de tamanho L definido como $\frac{1}{32}$ do BPM e o deslocamento da janela, definido como metade do *frame* STFT, ou seja, $\frac{1}{64}$ do tempo da batida. Após esse processo é desenvolvido o cálculo da matriz de similaridade (FOOTE, 1999) a partir do espectrograma, que visa medir o nível de similaridade entre todos os *frames*, para assim, encontrar estruturas musicais repetidas e estimar a assinatura de tempo. A matriz de similaridade é construída a partir do cálculo da distância Euclidiana entre todas as combinações de *frames* do espectrograma, assim, quanto menor a distância, maior é a similaridade. O cálculo utiliza a seguinte fórmula:

$$ASM(a,b) = \sum_{k=0}^{N-1} [X(a,k) - X(b,k)]^2. \tag{2.13}$$

Para exemplificar, a Figura 9 (a) demonstra o espectrograma de um sinal e a Figura 9 (b) ilustra a matriz de similaridade para esse espectrograma. Nesse exemplo, cada compasso (ou *bar*, termo utilizado na língua inglesa) é composto por 6 batidas, na Figura 9 (b) o grupo 1-2 representa a relação de similaridade entre o compasso 1 e o 2. Ainda é possível notar que no grupo 2-3 tem-se alta similaridade, pois são compassos iguais (quanto mais escuro a célula da matriz, maior é a similaridade).

O método é executado a partir de candidatos de tamanho de compasso que variam entre 2 e 12 batidas. Após selecionar um candidato, seleciona-se todas as diagonais inteiras múltiplas do seu tamanho, porém escolhe-se apenas um lado da matriz, pois a mesma é simétrica, como é mostrado na Figura 9 (b). Depois, cada diagonal é particionada em seguimentos do mesmo tamanho do compasso candidato, exceto os compassos incompletos. Assim, G define os compassos completos e P os incompletos que são mostrados no final do eixo horizontal da Figura 9 (b). Para medir a similaridade do tamanho de compasso candidato é definido SCS e SIS, estes promovem o nível de similaridade entre dois compassos, por exemplo, a aplicação para o grupo 1-2 (Figura 9 (b)) mede a similaridade entre o compasso 1 e 2. As equações SCS e SIS são definidas como:

números em forma de fração, onde o numerador indica quantas figuras de tempo cabem dentro de um compasso e o denominador determina a espécie da figura de tempo. Essa figura de tempo tem relação com BPM, assim, relacionando o BPM, a figura de tempo e o compasso é possível interpretar a sua duração (MED, 1996). Por exemplo, uma assinatura de tempo 2/4 significa que cada compasso contém 2 duas figuras de tempo do tipo 4 (semínima), e a duração dessa figura de tempo é determinada pelo BPM, onde o tempo de figura é igual ao tempo de uma batida.

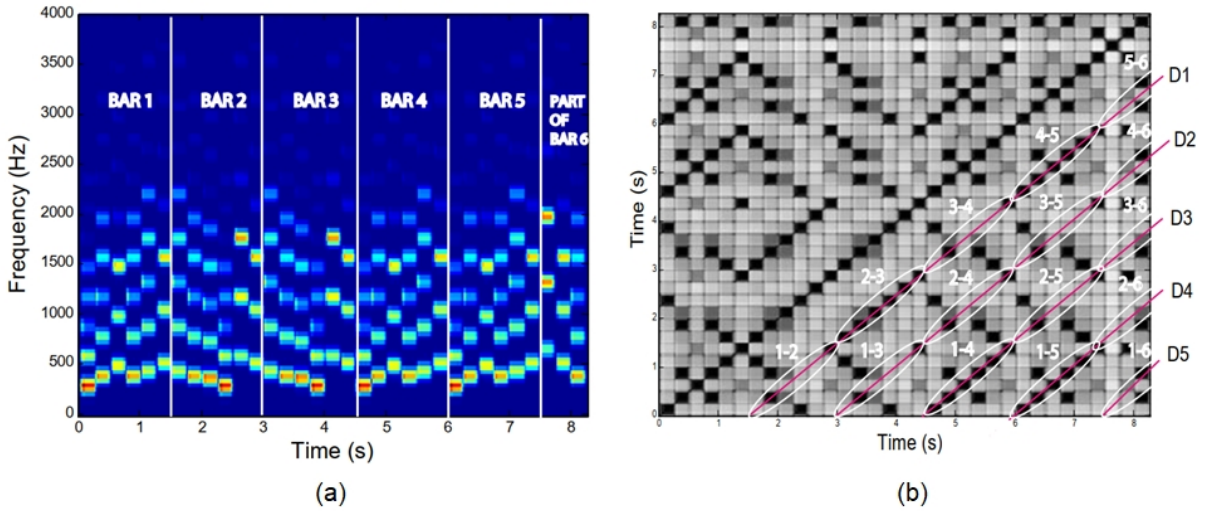

Figura 9 – (a) Gráfico do espectrograma de um sinal, representando as intensidades de frequência para cada *frame* por cores e (b) ilustração da matriz de similaridade, que representa níveis de similaridade entre *frames*.

Fonte: (COYLE; GAINZA, 2007)

$$SCS = \sqrt{\frac{\sum_{i=1}^{Bar} G_i^2}{Bar}} \tag{2.14}$$

e

$$SIS = \sqrt{\frac{\sum_{i=1}^{r} P_i^2}{r}}, \tag{2.15}$$

onde G_i é $i-ésimo$ componente do compasso completo, Bar é o tamanho do compasso candidato, P_i é o $i-ésimo$ elemento do compasso incompleto e r é o tamanho o compasso incompleto. Assim, é possível montar uma nova matriz de similaridade a partir dos elementos SCS e SIS como mostra a Figura 10. Como exemplo, $SCS_{(1,2)}$ corresponde a medida de similaridade formada pelos compassos 1 e 2. Por fim, como é gerado uma nova matriz de similaridade para cada tamanho de compasso candidato, é preciso medir o nível de similaridade dessa nova matriz como um todo, para isso utiliza-se a seguinte fórmula:

$$SM = \frac{Bar \times \sum_{i=1}^{S_c} SCS_i + r \times \sum_{i=1}^{S_i} SIS_i}{Bar \times s_c + r \times s_i}, \tag{2.16}$$

onde S_c e S_i correspondem ao número de seguimentos SCS e SIS, respectivamente.

Contudo, o método é executado partindo-se do princípio de que a primeira batida do áudio em analise representa também a primeira batida do compasso, porém isso pode não ser uma verdade absoluta. Assim, para contornar a situação é preciso tratar todos os casos, para isso desloca-se a batida inicial candidata para as batidas subsequentes, e para cada deslocamento refaz-se todo o processo gerando-se uma nova matriz de similaridade de compassos e um novo SM. O deslocamento corresponde aos valores que vão de 0 até valor do tamanho candidato menos 1, ou seja, se o tamanho candidato for 4, irá gerar uma

					$SIS_{(5,6)}$
				$SCS_{(4,5)}$	$SIS_{(4,5)}$
			$SCS_{(3,4)}$	$SCS_{(3,5)}$	$SIS_{(3,6)}$
		$SCS_{(2,3)}$	$SCS_{(2,4)}$	$SCS_{(2,5)}$	$SIS_{(1,6)}$
	$SCS_{(1,2)}$	$SCS_{(1,3)}$	$SCS_{(1,4)}$	$SCS_{(1,5)}$	$SIS_{1,6)}$

Figura 10 – Nova matriz de similaridade gerada a partir da Figura 9 (b).

Fonte: (COYLE; GAINZA, 2007)

nova SM para deslocamento 0, 1, 2 e 3, cobrindo assim, todas as possibilidades do áudio começar na batida 1, 2 ou 3. Por fim, o tamanho candidato que apresentar o menor valor SM será considerado como numerador da fórmula do compasso.

2.5 Open Sound Control

O Open Sound Control (OSC) (WRIGHT; FREED, 1997), desenvolvido em 1997, é um protocolo de código aberto, eficiente e independente da camada de transporte para comunicação entre computadores, sintetizadores de som e outros dispositivos multimídia que implementam tecnologias de rede.

O OSC segue a premissa de funcionamento sobre várias tecnologias de rede como USB, *Ethernet*, *fast Ethernet* e outras. Assim, por ser independente do transporte, ou seja, não possuir camada de transporte particular, seus dados podem ser formatados para as novas tecnologias de rede emergentes. Os Autores, Wright e Freed (1997), descrevem possibilidade de transmissões com velocidade maior do que 10 Megabit/segundo (dependendo da tecnologia de rede). Além disso, as novas tecnologias de rede permitem inúmeras conexões de dispositivos.

Todos os dados OSC são múltiplos de 4 bytes. Os dados numéricos são codificados pela representação nativa da máquina de 32 bits ou 64 bits *big-endian*, complemento de dois para inteiros e o padrão IEEE 754 (IEEE. . . , 2008) para pontos flutuantes. *Strings* são representadas por sequências de caracteres ASCII não nulos, contudo são completados com caracteres nulos extras para obter um tamanho múltiplo de 4 bytes.

A unidade de dados básica do OSC é a *message*, que consiste em um endereço simbólico e uma série de dados binários que representam os argumentos da mensagem (*strings*, inteiros, *floats* e etc). Um OSC *packet* pode conter uma *message* ou um *bundle*. O *Bundle* consiste em uma *string* de reconhecimento ("bundle"), 64 bits representando o

time tag (tempo absoluto em que a mensagem deve ser executadas) e qualquer quantidade de mensagens ou *bundles*, precedidos por um contador (de mensagens ou *bundle*) de 4 bytes. Note que os *bundles* são estruturas recursivas, ou seja, dentro de um *bundle* pode conter vários outros.

Uma mensagem enviada em uma rede gasta um tempo (dezenas de milissegundos) para chegar ao receptor devido aos componentes da rede, além disso pode haver uma variação no tempo gasto pela mensagem, chamada de *jitter*. Assim, o OSC implementa o *time tag* que permite incluir o tempo absoluto com precisão de picossegundos em que a mensagem deve ser executada, dessa maneira, a mensagem pode ser enviada com antecedência para o dispositivo receptor, contornando o atraso e *jitter* da rede, e o dispositivo pode executar a mensagem de acordo com seu relógio interno. Entretanto, quando se trata do controle de múltiplos dispositivos não há garantia de sincronia entre os relógios internos desses dispositivos, o protocolo OSC não fornece mecanismos para sincronização de relógios, esse tema é discutido na Seção 2.6.

O modelo de endereçamento do protocolo OSC segue uma perspectiva hierárquica similar ao utilizado em notações URL, por exemplo, /voices/drone-b/resonators/3/set-Q. O OSC não descreve como deve ser organizado o endereçamento, assim, cada sistema definirá a sua própria organização.

MIDI (*Musical Instrument Digital Interface*) é uma protocolo para comunicação entre instrumentos musicais digitais criado da década de 80 (MOOG, 1986). Esse protocolo foi originalmente desenvolvido para transmissão via cabo, com velocidade de transmissão limitada e baixo número de canais para controle dos instrumentos, trabalhando apenas com o tipo de dados inteiro. Neste trabalho o OSC foi escolhido como protocolo de comunicação, a escolha se deu pelo fato do OSC ser originalmente definido independente da camada de transporte, podendo ser implementado em tecnologias de rede sem fio atuais para proporcionar alta velocidade de transmissão. Além disso, o OSC permite a transmissão de número real, *string* e inteiro com tipo de dados.

2.6 Simple Network Time Protocol

A sincronização é um fator importante para aplicações onde há colaboração entre robôs, computadores, ambientes de automação e outros, pois podem existir ações escalonadas no tempo, onde os robôs demandam de ações conjuntas com certo intervalo de tempo para execução de determinada tarefa. Assim, um dos requisitos básicos para sincronização entre robôs, que demandam execução de ações por tempo absoluto, é manter a sincronia entre seus relógios. Isto posto, uma maneira de manter os relógios dos dispositivos sincronizados é por intermédio do SNTP (*Simple Network Time Protocol*) (MILLS, 2006). O SNTP é uma versão simplificada do NTP (*Network Time Protocol*) (MARTIN et al., 2010).

O NTP utiliza vários algoritmos complexos e várias referências de tempo para fornecer o tempo correto para o cliente. Contudo, o SNTP utiliza um sub-conjunto desses algoritmos e apenas um servidor de referência. Em contra partida, a simplicidade do SNTP remete a um menor esforço computacional, tornando-o atraente para computadores menos potentes, micro-controladores e sistemas embarcados.

O protocolo SNTP utiliza uma estrutura de cliente-servidor, assim, o cliente inicia o processo fazendo uma requisição SNTP para o servidor, para então calcular a diferença entre o seu relógio e o do servidor, como é mostrado na Figura 11. Assim, o cliente envia uma requisição no tempo t_1 (e armazena essa informação), após chegar no servidor, no tempo t_2, o mesmo responde a requisição com uma mensagem contendo o tempo t_2 e t_3 (momento que o servidor envia a resposta), quando a resposta chega no cliente o mesmo grava o tempo de chegada t_4. Por fim, com todas essas informações de tempo disponíveis é possível calcular a diferença entre os relógios utilizando a fórmula dada por:

$$\Delta t_{Relógio} = \frac{(t_2 - t_1) + (t_3 - t_4)}{2}, \tag{2.17}$$

e a fórmula para cálculo do tempo de ida e volta da informação (RTT - *Round Trip Time*) é dada por:

$$t_{RTT} = (t_4 - t_1) - (t_3 - t_2)\,. \tag{2.18}$$

O protocolo SNTP assume que o tempo de atraso da transmissão é simétrico em ambas direções, o que certamente afeta a acurácia da sincronização.

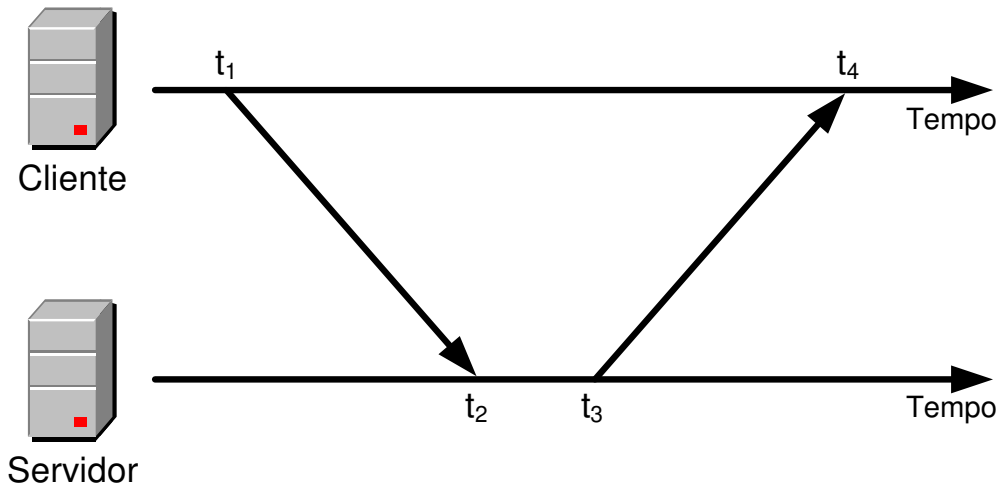

Figura 11 – Troca de mensagens entre um cliente e um servidor SNTP

Fonte: Autor.

2.7 Redes Neurais Artificiais

De acordo com Haykin (2001), uma rede neural artificial pode ser definida como um processador paralelamente distribuído formado por unidades de processamento mais simples, onde essas unidades possuem características de aprendizado experimental.

2.7.1 Modelo de um Neurônio Artificial

O neurônio é uma unidade fundamental para a operação de redes neurais, onde na Figura 12 é demonstrado o diagrama de um modelo. Haykin (2001) identifica três elementos básicos para um modelo neural, estes são:

- conjunto de sinapses: pesos atrelados aos sinais de entrada;
- somador: unidade responsável por somar os valores de entrada ponderados pelos pesos sinápticos;
- função de ativação: função que restringe a amplitude de saída de um neurônio.

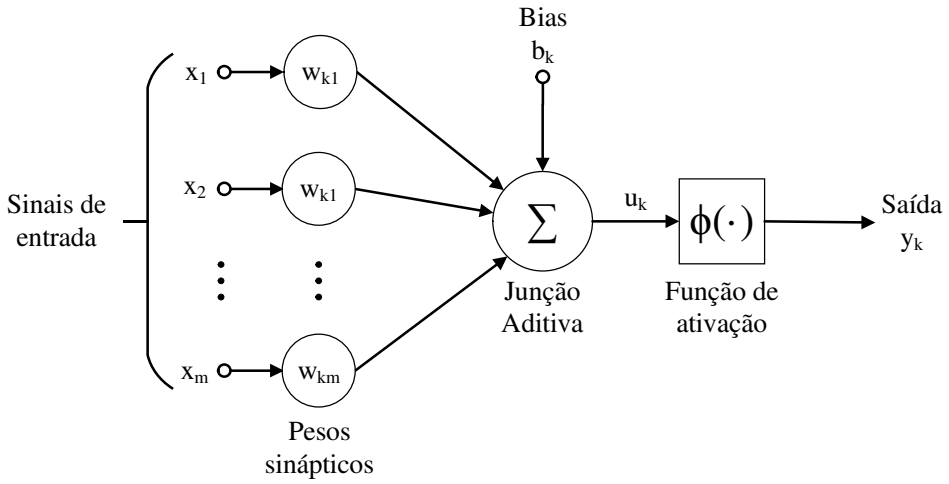

Figura 12 – Modelo de um neurônio artificial

Fonte: Adaptado de (HAYKIN, 2001)

O modelo da Figura 12 inclui também a adição de um *bias* que é aplicado externamente, representado por b_k, onde o objetivo é modificar a entrada da função de ativação. Em termos matemáticos, o neurônio pode ser representado por:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.19}$$

e

$$y_k = \varphi\left(u_k + b_k\right) \tag{2.20}$$

onde x_j representa os sinais de entrada, w_{kj} denota os pesos sinápticos, u_k representa o resultado do somatório do sinal de entrada ponderado pelos pesos, b_k é o *bias*, φ é a função de ativação e y_k é o sinal de saída do neurônio.

2.7.2 Funções de Ativação

A saída da rede neural é definida pela função de ativação (HAYKIN, 2001). As funções de ativação são diversas, este trabalho utiliza duas funções, a função sigmóide definida por:

$$\varphi\left(v_k\right) = \frac{1}{1 + e^{-\alpha v_k}}, \tag{2.21}$$

onde $v_k = u_k + b_k$ e α é o fator de curvatura da função; e a função ReLU, definida por:

$$\varphi\left(v_k\right) = max(0, v_k). \tag{2.22}$$

A Figura 13 mostra a representação gráfica da Sigmóide com $\alpha = 1$ e a ilustração da ReLU.

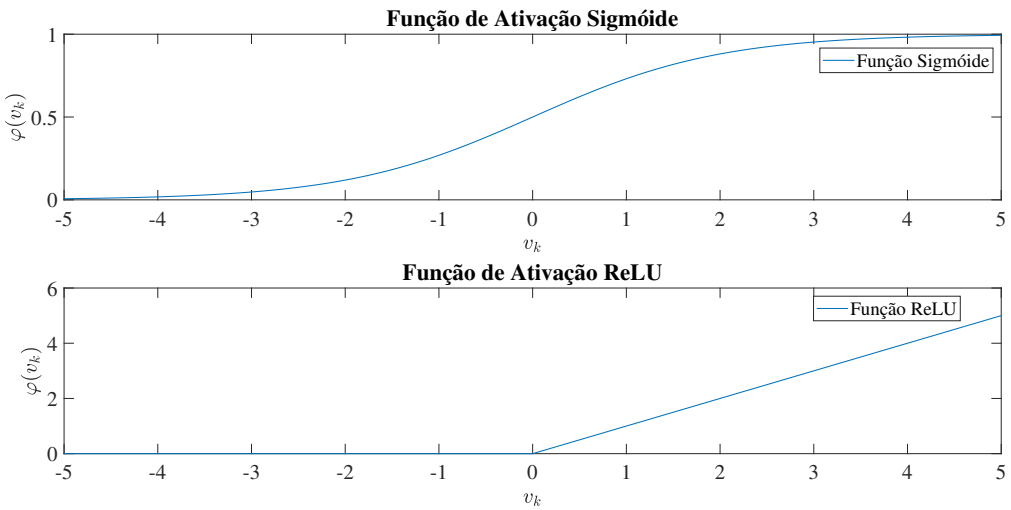

Figura 13 – Função de ativação Sigmóide com $\alpha = 1$ e função de ativação ReLu

Fonte: Autor.

2.7.3 Estrutura de Redes de Múltiplas Camadas Alimentadas Diretamente

De acordo com Haykin (2001), a arquitetura de uma rede neural pode ser de uma camada, contendo apenas os nós de entrada (que não processam o sinal) e a camada de neurônios de saída, ou redes de múltiplas camadas, onde os sinais de saída da segunda camada são utilizados como entrada para a terceira, e assim por diante até a camada de saída. As camadas entre os nós de entrada e a camada de saída são chamadas de camadas ocultas, essas trazem mais robustez a rede, propiciando extrair estatísticas de

ordem elevada. A Figura 14 demonstra uma estrutura 10-4-2, ou seja, 10 nós de entrada, 4 neurônio ocultos e 2 neurônios de saída. Cada neurônio segue a formulação matemática expressa no modelo da Subseção 2.7.1. Vale destacar que uma das primeiras publicações sobre redes neurais seguindo essa estrutura é descrito por Rosenblatt (1958), chamada de Perceptron.

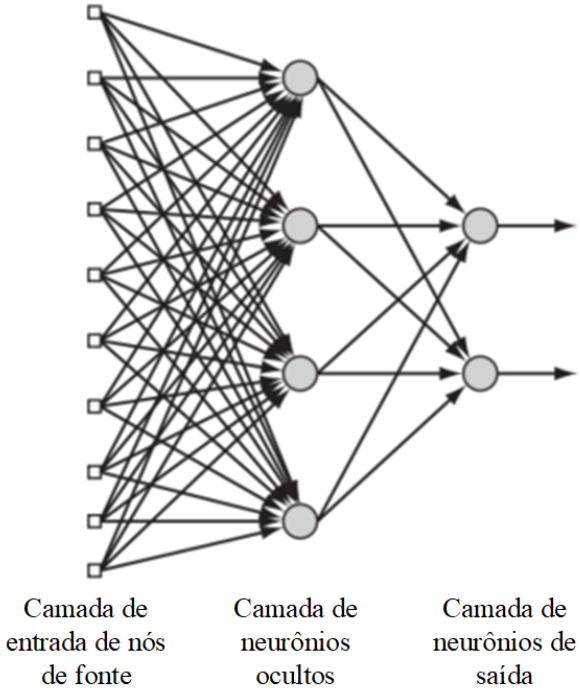

Figura 14 – Estrutura de rede neural de múltiplas camadas

Fonte: (HAYKIN, 2001).

Nota-se que no exemplo descrito na Figura 14 todos os neurônios de uma camada estão conectados com todos os neurônios da camada seguinte, sendo uma rede totalmente conectada, também chamada de densa. No entanto, a rede pode ser parcialmente conectada, onde uma porcentagem das conexões é excluída visando obter menor erro e evitar a especialização nos dados de treinamento. Esse processo também é conhecido como *dropout*.

A propriedade mais fascinante em redes neurais é a habilidade absorver conhecimento por meio de experiências. Um dos métodos utilizados para isso é o aprendizado supervisionado, onde cada exemplo contém tanto os sinais (parâmetros) de entrada da rede quanto o sinal de saída (resposta correta). Assim, um algoritmo muito utilizado para aprendizagem é o de retropropagação (*backpropagation*) que tem como base a correção de erro. A rede compara a resposta gerada com a resposta correta e propaga a diferença entre as repostas (o erro) da saída até a entrada, onde os pesos dos neurônios são alterados de acordo com alguma estratégia para tentar minimizar o erro (HAYKIN, 2001).

2.8 Resumo do Capítulo

Esse Capítulo descreveu o referencial teórico necessário para entendimento deste trabalho. Iniciando com um levantamento bibliográfico sobre instrumentos musicais robóticos, seguido pela apresentação dos conceitos de ubimus (*ubiquitous music*) que visa uso de sistemas onipresentes de agentes humanos e recursos materiais tecnológicos para atividades musicais, e IoMusT (*Internet of Musical Things*) que faz uso de infraestrutura de IoT (*Internet of Things*) para propiciar aplicações de ubimus.

Ainda, apresentou-se sobre processamento de sinais digitais, descrevendo algoritmos para detecção de picos, rastreamento de batidas (pulsos musicais) e assinatura de tempo. Em sequência, retratou-se sobre o OSC (*Open Sound Control*), um protocolo para comunicação entre computadores, sintetizadores de som e outros dispositivos multimídia. Como o OSC não promove algoritmos para sincronização dos relógios dos dispositivos em rede, o SNTP (*Simple Network Time Protocol*) foi apresentado com essa finalidade. E, por fim, foi descrito sobre redes neurais artificiais de múltiplas camadas alimentadas diretamente.

3 RoboMus

3.1 A plataforma RoboMus

A plataforma RoboMus (CAMPOREZ et al., 2018b) tem por objetivo a performance musical utilizando instrumentos musicais robóticos controlados por humanos através de múltiplas interfaces utilizando uma estrutura cliente-servidor, podendo conter módulos autônomos nos robôs. Para isso, a plataforma define formatos escalonáveis de mensagens para controle dos robôs utilizando o protocolo OSC. Além disso, possui arquitetura aberta de *hardware* e *software* visando facilitar a réplica da plataforma por outros usuários. Esse projeto tem base na configuração de múltiplos robôs em performances musicais, as quais demandam de tempo real. Assim, estratégias de sincronização são propostas através de um servidor de macrossincronização de mensagens multimídia e algoritmos de microssincronização embarcados nos próprios robôs.

O projeto da plataforma se divide em três frentes de trabalho: a) projeto e desenvolvimento dos robôs musicais com estratégia de microssincronização; b) interfaces de controle dos robôs, que abrangem linguagem de programação musical, API, interfaces colaborativas, pervasivas/ubíquas, tangíveis, gestuais, análise de biossinais, entre outras e c) definição de um padrão de comunicação através de formato de mensagens, onde um servidor centralizador é utilizado no escalonamento de mensagens para empregar técnicas de compensação de atrasos mecânicos provenientes dos robôs (macrossincronização). A arquitetura da plataforma RoboMus é descrita na Figura 15, onde na parte esquerda mostram-se as interfaces de controle (Seção 3.3), no centro, o Servidor de Sincronização de Mensagens Musicais (SSMM) (Seção 3.2) os instrumentos musicais robóticos ou robôs musicais (Seção 3.7 e Apêndice A).

3.2 Servidor de Sincronização de Mensagens Musicais

O servidor de sincronização de mensagens musicais funciona como um centralizador de mensagens (uma espécie de *buffer*) entre os diversos clientes (interfaces de controle) e os robôs musicais. As interfaces de controle e os robôs devem se apresentar a esse servidor para iniciar seu funcionamento (processo explicado na Subseção 3.4.2). A principal função do SSMM é na aplicação de estratégias para macrossincronização entre as interfaces de controle e os robôs. Por exemplo, fornecer mecanismo para sincronização de relógios entre componentes da plataforma, aprender os perfis de atrasos mecânicos dos robôs, aplicar estratégias para compensar os atrasos e converter as mensagens de tempo relativo (tempo musical), utilizadas nas interfaces de controle, em mensagens de tempo absoluto (*time tag*

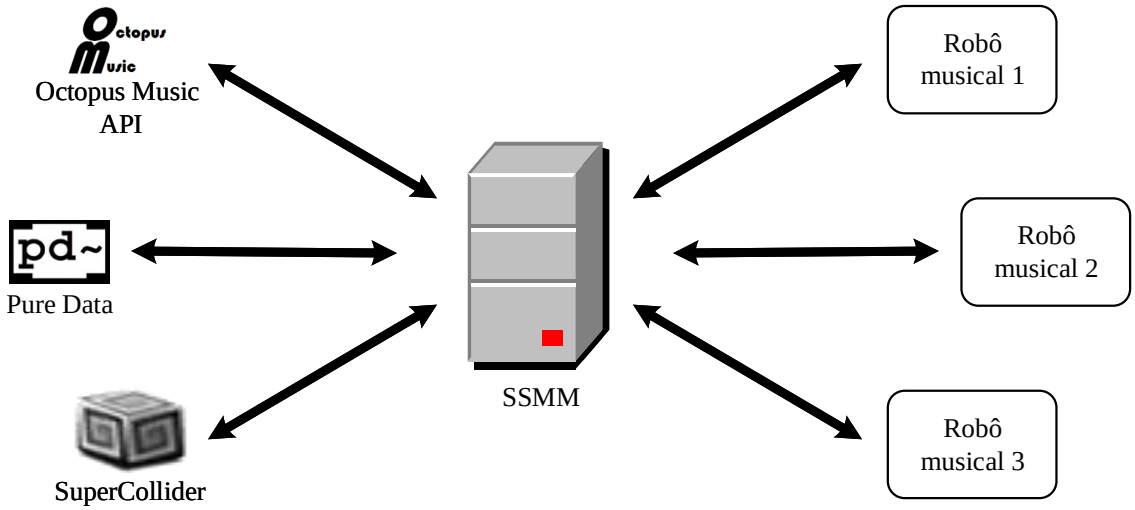

Figura 15 – Arquitetura da plataforma RoboMus

Fonte: Autor.

do OSC) que são aplicadas aos dispositivos de controle dos robôs.

A percepção sonora humana é capaz de distinguir diferenças temporais entre dois eventos acústicos sucessivos entre 500 e 1.500 Hz em até 20ms (PISONI, 1977). Já a capacidade de detecção de falhas em um sinal entre 400 e 2000 Hz é de cerca de de 6 a 8 ms (MOORE; PETERS; GLASBERG, 1993). Assim, trabalhar com janelas de tempo tão pequenas se torna um desafio computacional para sincronização de áudio, principalmente devido aos componentes eletromecânicos dos robôs. Os músicos possuem a impressionante habilidade de replicar o perfil de expressão de uma peça musical com variabilidade nas propriedades de tempo de 1% ou menos (CLARKE, 1993). No entanto, os instrumentistas não podem controlar todas as variações no desempenho, mesmo quando eles tentam tocar sem qualquer elemento de expressividade (PALMER, 1997). Do ponto de vista do controle temporal, humanos são capazes de se manter estáveis em níveis hierárquicos mais altos e complexos, por exemplo, ao executar peças musicais inteiras. O desvio padrão total na duração de 35-40 minutos é de cerca de 1%.

3.3 Interação Humano-Robô

As interfaces de interação para controle dos instrumentos musicais robóticos podem ser de inúmeros modelos, uma linguagem de programação, uma API, um dispositivo de controle – por exemplo: *WiiMote*[1], *smartphone* e instrumentos aumentados. As interfaces devem conhecer o protocolo OSC e os formatos de mensagens RoboMus definidos na Seção 3.4. As interações com os robôs alinham o trabalho juntamente com os sistemas de música

[1] controle de videogame

ubíqua (ubimus), pois este suporta múltiplas formas de interação e múltiplas fontes sonoras (geradas pelos robôs) de forma integrada (KELLER; LAZZARINI; PIMENTA, 2014).

Para controle dos robôs idealiza-se uma linguagem de programação interpretada (tempo real), concorrente, de fácil aprendizado e boa legibilidade, que faça uso de conceitos facilmente reconhecidos por músicos. Assim sendo, essa linguagem de programação pode servir como base para desenvolvimento de interfaces de controle, como por exemplo, interface direcionadas a pessoas portadoras de deficiência motora fazendo o uso de biossinais ou também um ambiente de programação gráfica voltada ao publico leigo, tanto em música quanto em programação. A base dessa linguagem está na biblioteca de modelagem de performances musicais denominada Octopus Music API (COSTALONGA; MILETO; MIRANDA, 2007). Os motivos dessa escolha estão ligados a facilidade em se modelar dados musicais, principalmente os dados da performance. Entretanto, o desenvolvimento dessa linguagem de programação perpassa o escopo deste trabalho.

O Octopus Music API tem uma característica marcante que é a modelagem de parâmetros de expressividade musical visando a simulação da humanização da performance. O termo expressividade, refere-se ao desvio sistemático da regularidade mecânica em relação aos valores nominais descritos na partitura ou em outra forma de codificação. Os parâmetros como variações no tempo, intensidade, timbre e articulação, bem como as variações de altura (conhecido como vibrato), constituem importantes características das performances musicais expressivas (DOGANTAN-DACK, 2006). Assim, as performances musicais robóticas não são nativamente expressivas, porém essas características podem ser simuladas.

Dentre o desenvolvimento de interfaces assistivas encontra-se o trabalho, ainda em andamento, Olhar Musical[2] (CAMPOREZ et al., 2018a) que visa a inclusão de pessoas portadoras de deficiência motora no mundo musical. Essa interface faz uso de um dispositivo para rastreamento de olho (*eye tracker*), em um primeiro momento, para controle de amostras musicais. Entretanto, os formatos de mensagens da plataforma RoboMus podem ser implementados na interface para permitir o controle de robôs em vez de amostras musicais. A Figura 16 demonstra seis células centrais que controlam uma amostra de áudio cada, onde o usuário pode selecionar uma célula central e escolher o botão *play*, para inciar, ou *pause*, para encerrar a execução. Essa interface faz comunicação via protocolo MIDI com o software de produção musical Ableton Live[3], ou seja, as amostras de áudio são executadas pelo live, porém são controladas pela interface. Vale ressaltar que os botões *play* e *pause* do lado esquerdo da Figura 16 possibilitam o usuário conhecer previamente as amostras através de um canal de áudio separado da performance. Já os botões do lado direito executam a amostra diretamente na performance.

[2] Vídeos demonstrativos podem ser encontrados em: <https://youtu.be/YDj8pr3Cnx8> e <https://youtu.be/PzSIt603LWo>

[3] <https://www.ableton.com/en/live/>

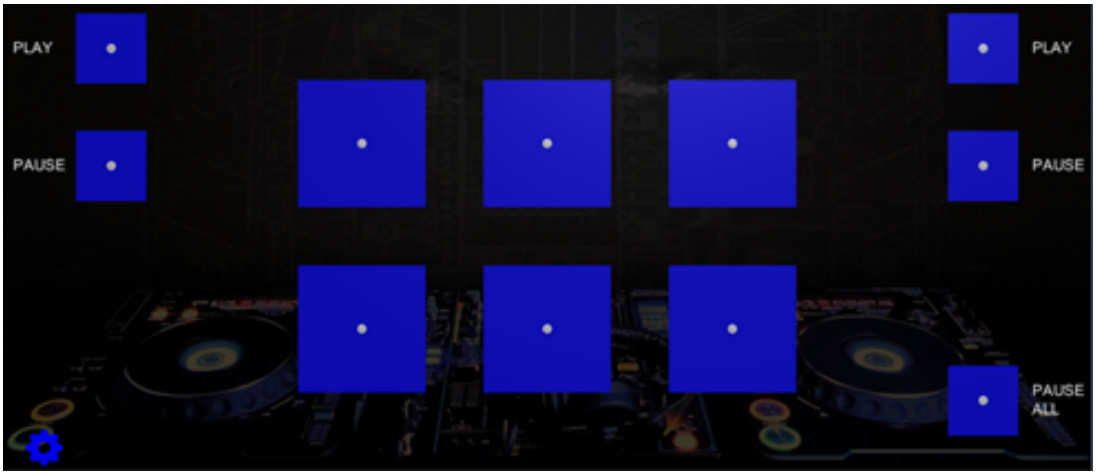

Figura 16 – Interface assistiva para controle musical chamada Olhar Musical

Fonte: (CAMPOREZ et al., 2018a).

3.4 Comunicação RoboMus

3.4.1 Atributos dos Robôs

O Octopus Music API (COSTALONGA; MILETO; MIRANDA, 2007) define um padrão de atributos para instrumentos em um modelo hierárquico de classes. Inspirado no Octopus, os robôs RoboMus seguem os atributos estabelecidos pelos autores e a divisão de classes de instrumentos. Assim, todo robô deve conter os seguintes atributos:

- identidade: nome do instrumento robótico;
- polifonia: quantidade máxima de notas que o robô consegue tocar ao mesmo tempo;
- ações específicas: os movimentos que o robô consegue fazer, ou seja, as ações que podem se executadas e os parâmetros necessários para tais;
- endereço IP: endereço na rede do robô;
- porta: a porta para comunicação;
- tipo/família: tipo do instrumento, que segue uma classificação ergonômica; ex: teclas, trasteado de cordas, percussivo, etc;
- endereço OSC: endereço de acordo com o formato do OSC.

O padrão definido para o atributo *ações específicas* define que cada ação deve ser representada entre "<>", onde, dentro de cada ação, as informações são separadas por ";". Entretanto, o primeiro componente da ação representa o endereço OSC e os demais são os parâmetros seguindo o formato de nome e tipo separados por "_". Os tipos são definidos no

Quadro 1. Cada parâmetro da ação possui um conjunto de valores válidos, esses conjuntos são adicionados ao final das ações, onde os conjuntos para uma determinada ação são representados entre parenteses e separados por ";", exceto o primeiro termo que é o nome da ação. Abaixo segue o modelo descrito:

$$< /\text{ação}_1; par\hat{a}metro_{11}_tipo; par\hat{a}metro_{12}_tipo; ...; par\hat{a}metro_{1N}_tipo > ... <$$
$$/\text{ação}_N; par\hat{a}metro_{N1}_tipo; ...; par\hat{a}metro_{NN}_tipo >$$
$$(/\text{ação}_1; conjunto_{11}; conjunto_{12}; ...; conjunto_N)...$$
$$(/\text{ação}_N; conjunto_{1N}; conjunto_{2N}; ...; conjunto_{NN}).$$

Os conjuntos supracitados podem ser representados de duas formas: 1) determinando o intervalo com limite superior e inferior, como por exemplo, $]a, b]$ onde a é o limite inferior, mas não pertence ao conjunto, representado por $]a$, e b é o limite superior que pertence ao intervalo, denotado por $b]$, seguindo o padrão matemático de intervalos; 2) passagem literal dos valores, nesse caso os valores são contidos entre chaves ("{}") e separados por vírgula, por exemplo, $\{p1, p2, p3, ..., pn\}$. Para melhor exemplificar, segue um exemplo:

$$< /tocar; corda_i; traste_i; intensidade_f >< /tocarCorda; corda_i >$$
$$(/tocar; \{1, 2, 3, 4, 5, 6\}; [1, 12]; [0, 1])(/tocarCorda; [1, 6]).$$

No exemplo anterior o robô possui apenas duas ações, a de *tocar* e a de *tocarCorda*. Note que a ação *tocar* possui três parâmetros *corda_i*, *traste_i* e *intensidade_f*, seus conjuntos são $\{1, 2, 3, 4, 5, 6\}$, $[1, 12]$ e $[0, 1]$, respectivamente. Já a ação *tocarCorda* possui apenas o parâmetro *corda_i* com intervalo definido por $[1, 6]$. Vale ressaltar que quando um parâmetro não possui intervalo, o campo do mesmo deve ficar vazio, como mostra o exemplo:

$$< /tocarCorda; corda_i; dura\tilde{c}ao_i; intensidade_f > (/tocarCorda; \{1, 2, 3, 4\}; ; [0, 1]).$$

A plataforma RoboMus dispõe de 4 tipos de dados: inteiro, ponto flutuante, *string* e nota musical (Quadro 1). Os três primeiros tipos seguem o formato definido pelo OSC. Já o tipo nota musical é enviado como tipo *string*, porém com o formato dado por *símbolo da nota musical* + *acidente*[4] + *oitava*[5], onde o símbolo da nota segue o padrão universal para notas utilizando caracteres (C = Dó, D = Ré, E = Mi, F = Fá, G = Sol, A = Lá e B - Si), o acidente pode ser representado por ♯ para sustenido, ♭ para bemol ou vazio para nota natural e a oitava descrita por um número inteiro. Por exemplo:

[4] São símbolos utilizados para alterar a altura de uma nota, tornando ela meio tom mais grave, denotado pelo simbolo ♭, ou meio tom mais aguda, denotado pelo símbolo ♯. Na música ocidental existem 12 notas, Dó, Dó♯, Ré, Ré♯, Mi, Fá, Fá♯, Sol, Sol♯, Lá, Lá♯ e Si. Note que Mi e Si tem exceções, onde o Mi♯ é igual a Fá e Si♯ é igual a Dó. Cada nota é espaçada de meio tom, assim, ré♭ é igual a dó♯.

[5] Denota a região de frequência que a nota se encontra, quanto maior o valor, maior é a frequência.

$$< /tocarNota; nota_n; duration_i > (/tocarNota; \{C4, C\sharp 4, D4, E4\}; [0, 500]).$$

Quadro 1 – Tipos de dados da plataforma RoboMus

Tipo	Identificador
Inteiro	i
Ponto flutuante	f
String	s
Nota musical	n

3.4.2 Mensagem de Apresentação

Dado que uma das premissas da plataforma é não limitar as futuras gerações de robôs e que, possivelmente, cada robô terá suas próprias características e ações devido as suas estruturas físicas, determina-se um formato de apresentação em que o robô se apresenta para o SSMM e o SSMM repassa as informações para as interfaces. Nesta apresentação visa-se transmitir todos os atributos do robô, principalmente, suas ações específicas que demonstram o que o robô é capaz de fazer, permitindo que alguma interface o controle sem nenhum conhecimento prévio. Esse conceito é denominado *handshake*.

Os robôs RoboMus ao se conectarem a uma sala de concerto virtual devem enviar uma mensagem *broadcast* de *handshake* utilizando a porta 7000 (porta padrão do servido SSMM) para se conectar ao SSMM. Essa mensagem é enviada com o endereço OSC */handshake/instrument* e os atributos citados na Subseção 3.4.1. O Quadro 2 demonstra todos os componentes da mensagem.

Quadro 2 – Formato da mensagem de *handshake* de um robô

Componente	Tipo
/handshake/instrument	Endereço OSC
Nome	*String*
Endereço OSC do instrumento	*String*
Endereço IP	*String*
Porta	Inteiro
Polifonia	Inteiro
Tipo/Família	*String*
Protocolo específico	*String*

As interfaces de controle também devem enviar suas mensagens de *handshake* ao se conectarem a uma sala de concerto. Da mesma forma, a interface envia uma mensagem *broadcast* de *handshake* utilizando a porta 7000. Contudo, a mensagem é enviada para o endereço OSC */handshake/client* contendo seus atributos básicos definidos no Quadro 3.

Quadro 3 – Formato da mensagem de *handshake* da interface

Componente	**Tipo**
/handshake/client	Endereço OSC
Nome	*String*
Endereço OSC da interface	*String*
Endereço IP	*String*
Porta	Inteiro

Para que os robôs ou interfaces tenham conhecimento do SSMM, após receber e processar uma mensagem de *handshake*, o servidor envia uma mensagem de resposta para o remetente. A mensagem de retorno segue o formato descrito no Quadro 4.

Quadro 4 – Formato da mensagem de resposta ao *handshake*

Componente	**Tipo**
/Endereço OSC do remetente/handshake	Endereço OSC
Nome	*String*
Endereço OSC do SSMM	*String*
Endereço IP do SSMM	*String*
Porta do SSMM	Inteiro

3.4.3 Mensagem de Desconexão

A desconexão de um robô, ou de uma interface, de uma sala de concerto deve ser feita através do envio de uma mensagem informando a saída. A mensagem é enviada diretamente ao SSMM com o endereço OSC */Endereço OSC do servidor/disconnect* e o endereço OSC do instrumento ou interface a ser desconectado como parâmetro (Quadro 5).

Quando há uma desconexão de um robô, tem-se a necessidade de avisar as interfaces de controle que um robô se desconectou da performance. Assim, o SSMM envia uma mensagem para todas as interfaces, o formato da mensagem é mostrado no Quadro 6.

Quadro 5 – Formato da mensagem de desconexão de um robô ou interface

Componente	Tipo
/Endereço OSC do servidor/disconnect	Endereço OSC
Endereço OSC do robô ou interface a ser desconectado	*String*

Quadro 6 – Formato da mensagem para informar às interfaces sobre a desconexão de um robô

Componente	Tipo
/Endereço OSC da interface/disconnect	Endereço OSC
Endereço OSC do robô desconectado	*String*

3.4.4 Mensagem para Cancelamento de uma Ação

O usuário pode, por algum motivo, desejar cancelar uma mensagem que já foi enviada ao robô, caso haja tempo hábil para isso. Além do usuário, a necessidade de cancelamento pode ser proveniente de alguma estratégia de sincronização do SSMM. Assim, a mensagem de cancelamento contém o seguinte endereço OSC */Endereço OSC/deleteMessage*, onde o endereço pode ser o do servidor, caso a mensagem seja proveniente de alguma interface, ou pode ser o do robô caso a mensagem seja proveniente do SSMM. A mesma possui um único parâmetro, o identificador da mensagem a ser cancelada, dado que toda mensagem de ação enviada aos robôs contém um identificador. Vale ressaltar que quando a mensagem é proveniente de alguma interface, o SSMM faz a conversão da mensagem antes de enviar ao robô. O formato da mensagem é descrito no Quadro 7.

Quadro 7 – Formato da mensagem para cancelar uma mensagem RoboMus

Componente	Tipo
/Endereço OSC/deleteMessage	Endereço OSC
Identificador da mensagem	Inteiro

3.4.5 Mensagem para Repasse de Informações dos Robôs

Ao se conectar a um sala de concerto, o robô envia suas informações apenas para o SSMM, entretanto é necessário que as interfaces de controle tenham conhecimento das informações dos robôs. Para isso, as interfaces clientes podem requisitar ao SSMM as informações dos robôs disponíveis através da mensagem mostrada no Quadro 8.

O SSMM, por sua vez, ao receber essa requisição envia várias mensagens para o cliente referente aos robôs disponíveis na sala de concerto, as informações de cada robô

Quadro 8 – Formato da mensagem para requisitar informações dos robôs disponíveis

Componente	Tipo
/Endereço OSC do servidor/getInstruments	Endereço OSC
Endereço OSC do cliente requisitante	*String*

são enviadas separadamente. O Quadro 9 demonstra o formato da mensagem.

Quadro 9 – Formato da mensagem para repasse das informações de um robô

Componente	Tipo
endereço OSC do cliente/instrument	Endereço OSC
Nome	*String*
Endereço OSC do instrumento	*String*
Endereço IP	*String*
Porta	Inteiro
Polifonia	Inteiro
Tipo/Família	*String*
Protocolo específico	*String*

3.4.6 Mensagem de Atraso

Uma das estratégia de macrossincronização é o aprendizado do perfil de atraso mecânico dos robôs, que será explicada na Subseção 3.5.1. Para isso, define-se um formato de mensagem onde o robô pode retornar o tempo gasto para executar uma ação. O formato da mensagem é descrito no Quadro 10, onde o endereço OSC é formado pela junção entre o endereço do servidor, a palavra *delay* e o endereço do instrumento remetente. O primeiro parâmetro é o identificador da mensagem (o mesmo usado na mensagem enviada pelo servidor) e o segundo parâmetro é o atraso mecânico.

Quadro 10 – Formato da mensagem para retorno do atraso (tempo gasto para executar uma ação)

Componente	Tipo
/Endereço OSC do servidor/delay/Endereço OSC do instrumento	Endereço OSC
Identificador	Inteiro
Atraso mecânico	Inteiro

3.4.7 Mensagem para Execução de Nota

Acredita-se que todo instrumento musical seja capaz de reproduzir notas musicais, até mesmo os percussivos. Assim, define-se que todo robô musical RoboMus seja capaz de interpretar a ação de tocar uma nota. Essa ação contém os seguintes parâmetros:

- identificador: identificador da mensagem;
- duração: duração do som da nota em milissegundos;
- nota: nota musical de acordo com o tipo nota musical da plataforma;
- intensidade: intensidade de execução da nota (número real de 0 à 1).

Vale ressaltar que essa ação não faz parte do atributo de protocolo específico, pois é um formato que deve ser conhecido por todos os participantes da plataforma. Entretanto, pode ser possível que algum instrumento não consiga executar essa ação com todos os parâmetros especificados. Nesse caso, o robô deve contornar a situação para melhor atender os requisitos. Por exemplo, se um robô não dispõe de controle de intensidade, o mesmo pode tocar na sua intensidade padrão. Vale ressaltar que as ações são enviadas como OSC *bundle*, logo contêm um campo específico para determinar o momento da execução (*time tag*), por isso a informação de tempo não está na lista acima. O Quadro 11 descreve o formato da mensagem.

Quadro 11 – Formato da mensagem para a ação de tocar uma nota

Componente	**Tipo**
/Endereço OSC do robô/PlayNote	Endereço OSC
Identificador	Inteiro
Duração	Inteiro
Nota	Nota
Intensidade	Ponto flutuante

3.4.7.1 Formato de Mensagens para a Família de Instrumentos Trasteados de Cordas

Para os instrumentos do tipo trasteado de cordas (violão, guitarra e etc) são definidos alguns requisitos e algumas ações. Assim, todo instrumento dessa família deve adicionar dois parâmetros na mensagem de *handshake* (Quadro 2) e o SSMM deve adicionar esses parâmetros na mensagem de repasse, descrita no Quadro 9, estes parâmetros são:

- número de trastes: quantidade de casas (trastes) que podem ser tocadas pelo robô;

- afinação das cordas: afinações abertas[6] das cordas, que são enviadas seguindo o padrão $\{número,\ afinação\ aberta\}$, onde o *número* é um inteiro que representa a numeração da corda e a *afinação aberta* segue o formado nota musical da plataforma. Um exemplo desse formando é dado por: $\{1, A3\}\{2, G3\}\{3, E4\}\{1, B4\}$.

Além disso, define-se um formato de mensagem para tocar nota que deve ser conhecido e executado por todos os instrumentos da família trasteado de corda. Essa ação contém os seguintes parâmetros:

- identificador: identificador da mensagem;
- corda: corda a ser tocada;
- traste: traste em que a coda deve ser pressionada, onde 0 é corda solta;
- dedo para pressionamento: qual dedo, análogo a mão humana, deve pressionar a corda (valores inteiros de 1 à 5);
- direção: direção de excitação da corda, de cima para baixo (valor 0) ou de baixo para cima (valor 1);
- dedo para tocar: dedo, análogo a mão humana, para tocar a corda (valor inteiro de 1 à 5);
- tipo do toque: se o toque é com palheta (valor 0) ou o dedo (valor 1);
- proximidade: distância entre o toque e a região de captação do som do instrumento, determinado por um número real de 0 à 1.
- intensidade: intensidade do toque (número real de 0 à 1).

O formato da mensagem de tocar nota por instrumentos robóticos trasteados é demonstrado no Quadro 12, vale ressaltar que essa ação não precisa ser enviada junto do atributo de ações específicas, pois é um formato que todos robôs RoboMus, desse tipo, devem conhecer. Entretanto, caso um robô não consiga executar a ação com todos os parâmetro definidos, o mesmo deve então contornar a situação para melhor se adequar a ação. Por exemplo, se o robô não possuir mecanismos para alterar o tipo do toque, o toque pode ser realizado com o seu tipo padrão.

[6] Afinação quando está solta, ou seja, quando a corda é tocada sem estar apertada em nenhuma casa.

Quadro 12 – Formato da mensagem para a ação de tocar uma nota de um instrumento do tipo trasteado.

Componente	Tipo
/Endereço OSC do robô/PlayNoteFretted	Endereço OSC
Identificador	Inteiro
Corda	Inteiro
Casa	Inteiro
Dedo para pressionamento	Inteiro
Direção	Inteiro
Dedo para tocar	Inteiro
Tipo do toque	Inteiro
Proximidade	Ponto flutuante
Intensidade	Ponto flutuante

3.5 Sincronização entre Robôs

Esta Seção visa descrever sobre as estratégias de sincronização da plataforma RoboMus, sendo dividida em macrossincronização, executada pelo SSMM que visa fornecer uma sincronia básica, e microssincronização que é formada por algoritmos embarcados nos robôs para fornecer uma sincronia fina.

3.5.1 Macrossincronização

As mensagens RoboMus possuem a informação de tempo de execução no formato absoluto (manda-se exatamente a hora em que deve ser executada), logo é preciso que os relógios dos robôs estejam em sincronia. Para isso, o SSMM funciona também como um servidor SNTP para fornecer a hora certa aos robôs. Entretanto, a sincronia dos relógios deve ser feita de tempo em tempo para que a imprecisão dos relógios não influencie na performance, essa imprecisão causa o efeito *drift* nos relógios (MAROUANI; DAGENAIS, 2008), onde esse passa a contabilizar mais "tempo" do que deveria.

Devido aos movimentos dos membros, os seres humanos possuem atrasos mecânicos. No entanto, eles contornam esses atrasos iniciando o movimento antes da desejada resposta sonora. Por exemplo, ao bater palmas é preciso antecipar o início do movimento dos membros para acertar a próxima palma no tempo certo. Os robôs possuem atrasos similares causados pelos componentes eletromecânicos. Assim, uma das estratégias de macrossincronização é aprender o perfil de atraso dos robôs e antecipar o seu movimento para mantê-los no tempo certo.

Como uma das premissas da plataforma é não limitar as futuras gerações de robôs, acredita-se que cada robô possua um desenho estrutural próprio e, consequentemente, seu próprio perfil de atraso. Além disso, esse perfil pode mudar ao longo do tempo devido aos desgastes naturais dos componentes e folgas nas estruturas. Assim, o SSMM utiliza redes neurais para aprender os perfis de atraso dos robôs e promover a macrossincronização. Por exemplo, o SSMM envia centenas de mensagens para um robô que ao terminar a execução de cada mensagem envia uma mensagem ao servidor contendo o tempo gasto, seguindo o modelo de mensagem descrito na Subseção 3.4.6. Após a execução dessas mensagens, o SSMM pode treinar uma rede neural específica para esse robô a fim de aprender o seu perfil de atraso. Vale ressaltar que todo o processo de aquisição de dados e treinamento da rede neural demanda certo tempo, podendo levar minutos ou horas. No entanto, esse processo pode ser feito nos bastidores da plataforma.

O atraso mecânico, para alguns robôs, possivelmente depende da última mensagem executada e da próxima mensagem, pois a última mensagem determina a posição atual dos componentes mecânicos. Por exemplo, o tempo para transladar entre duas casas em um violão robótico depende da atual casa em que se encontra o conjunto mecânico (última mensagem executada), e da próxima casa desejada (próxima mensagem a ser executada). Assim, uma estrutura genérica com variáveis de entrada e saída para as redes neurais é definida, onde a estrutura interna é definida posteriormente para cada robô de modo que possa alcançar o melhor resultado. As variáveis de entrada da rede são definidas utilizando as informações da última mensagem e da próxima mensagem, ou seja, os identificadores das ações[7] e os parâmetros das mensagens como é mostrado na Figura 17. O número de parâmetro para cada mensagem pode variar de acordo com a ação, assim o número de variáveis de entrada das redes neurais é definido como $2 \times n + 2$, onde n é quantidade de parâmetros da mensagem que contém mais parâmetros. Vale ressaltar que ações em que a quantidade de parâmetros é menor do que n, o SSMM completa as variáveis de entrada da rede com zeros.

3.5.2 Microssincronização

As estratégias de microssincronização visam a execução de algoritmos embarcados nos robôs para sincronização. Uma dessas estratégias é o módulo perceptual autônomo para permitir que um robô possa se sincronizar e acompanhar musicalmente outros robôs ou seres humanos. Música, geralmente, possui uma sequência de pulsos regulares que representam o seu andamento. Esses pulsos podem ser identificados até mesmo por pessoas não treinadas, e muitas vezes correspondem a bater palmas, ou o pé, no ritmo da música (BENWARD; SAKER, 2008). A música, em geral, pode ser estruturada em grupos de batidas (pulsos), denominado compassos, sendo que o tamanho do grupo e a duração da

[7] Esses identificadores das ações são definidos pelo SSMM para que a rede possa identificar de qual mensagem se trata. Esses são diferentes dos identificadores de mensagens.

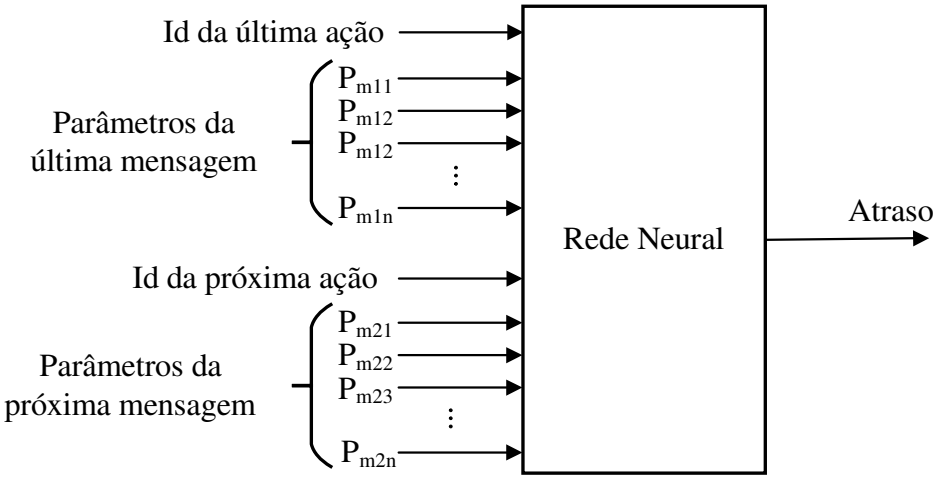

Figura 17 – Estrutura genérica para variáveis de entrada e saída para as redes neurais utilizadas no SSMM

Fonte: Autor.

batida são componentes que definem a assinatura de tempo. Essas informações permitem que os robôs criem padrões rítmicos para acompanhamento. Assim, o módulo perceptual é definido na Figura 18, onde os algoritmos para determinar o BPM global e a sequência de pulso são retratados nas Subseção 2.4.4.2 e 2.4.4.3, respectivamente, e a assinatura de tempo é obtida através do algoritmo definido na Subseção 2.4.5.

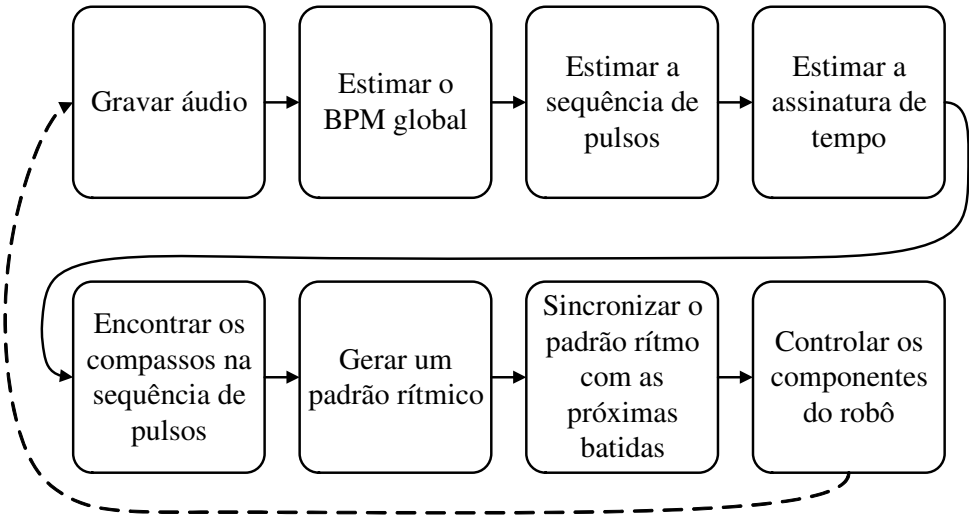

Figura 18 – Diagrama do processo de microssincronização baseado em pulsos musicais

Fonte: Autor.

3.6 Estrutura de Controle dos Robôs RoboMus

As salas de concerto da plataforma RoboMus disponibilizam rede sem fio para conexão. Assim, os robôs obrigatoriamente devem implementar os formatos de mensagens da RoboMus e possuir conectividade sem fio. Esses requisitos podem ser alcançados de inúmeras formas, no entanto, faz-se sugestões de estruturas de controle. Visando o baixo custo e reutilização de materiais, cita-se uma estrutura composta por um *smartphone* que em muitos casos costumam ficar esquecidos em gavetas, ou acabam virando lixo eletrônico, por estarem ultrapassados tecnologicamente, e um Arduino, como demonstrado na Figura 19 (a). Nesse caso, o *smartphone* faz todo processamento das mensagens RoboMus e se comunica via USB com o Arduino para controle dos componentes robóticos, onde o formato das mensagens enviadas via USB são definidos pelo projetista do robô. Outra possível estrutura é utilizando apenas um Raspberry, pois o mesmo possui interface de conexão sem fio e pinos de controle (Figura 19 (b)). Além dessas estruturas, um notebook pode substituir o celular demostrado na Figura 19 (a) para fazer todo processamento das mensagens como é demostrado na Figura 19 (c).

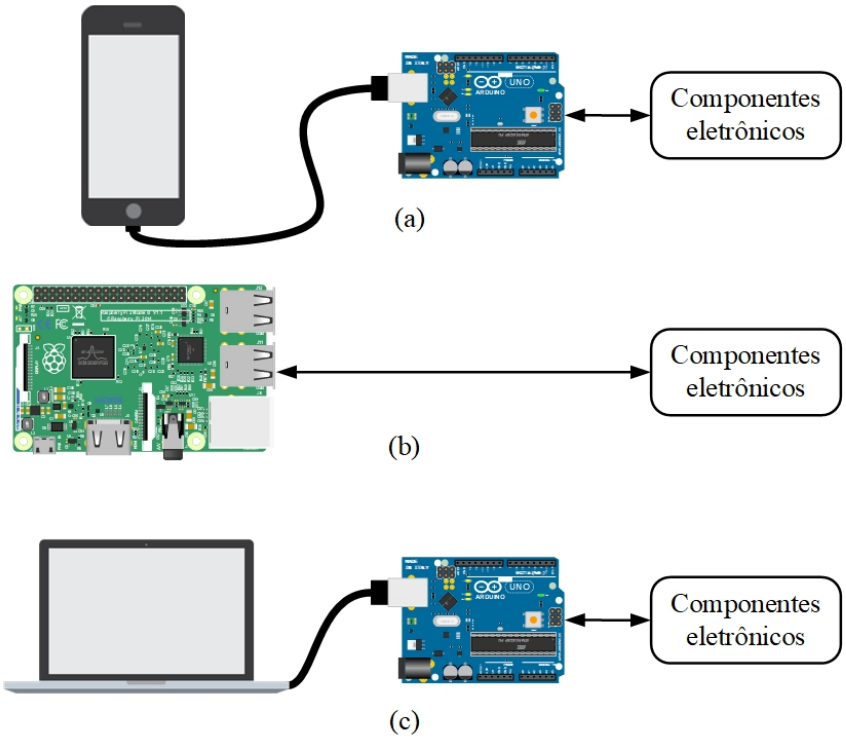

Figura 19 – Estruturas de controle para a plataforma RoboMus

Fonte: Autor.

3.7 Protótipo Bongô RoboMus

O BongoBot envolve a robotização de um bongô, um instrumento percussivo, seguindo a premissa de baixo custo no processo de desenvolvimento, já que essa é uma das premissas da RoboMus. O robô foi construído utilizando 4 solenoides (12 Volts, 300 mA e com força de 5 Newtons), 2 sensores piezoelétricos e um bongô. A Figura 20 representa o modelo 3D do robô, bem como o modelo real construído.

Figura 20 – A esquerda encontra-se o modelo 3D do BongoBot e, à direita, a representação real

Fonte: Autor

O dispositivo escolhido para controle do robô foi um Raspberry pi 3 model B[8], onde foram implementados todos os formatos de mensagens da RoboMus. No entanto, o Raspberry não é capaz de fornecer, através dos pinos de saída (GPIO), a tensão e corrente necessária para controlar os solenoides de 12 Volts e 300 mA. Assim, fez-se necessário a implementação de um circuito *driver* de corrente para obter essa tensão e corrente. O funcionamento desse circuito é basicamente como uma chave de interrupção, assim, o Raspberry aciona o circuito que consequentemente aciona o solenoide, a Figura 21 (a) demonstra o seu esquemático. Visando medir o tempo do atraso mecânico do robô, dois sensores piezoelétricos foram instalados embaixo das peles do Bongô, um em cada tambor. O Raspberry possui apenas portas de entradas digitais limitadas a 3,3 Volts, assim, um circuito amplificador não inversor com ganho ajustável foi implementado (Figura 21 (b)) para permitir que o Raspberry consiga ler o picos de tensão do sensor piezoelétrico para

[8] <raspberrypi.org/products/raspberry-pi-3-model-b/>

identificar as batidas. Entretanto, os ganhos foram ajustados de modo que a batida no tambor vizinho não interfira no sensor do outro tambor. Vale ressaltar que o circuito amplificador limita tensões acima de 3,3 (valor máximo da fonte de alimentação) e tensões abaixo de 0 Volts, fornecendo proteção à porta do Raspberry. A Figura 22 demonstra um diagrama com as conexões entre os componentes do robô.

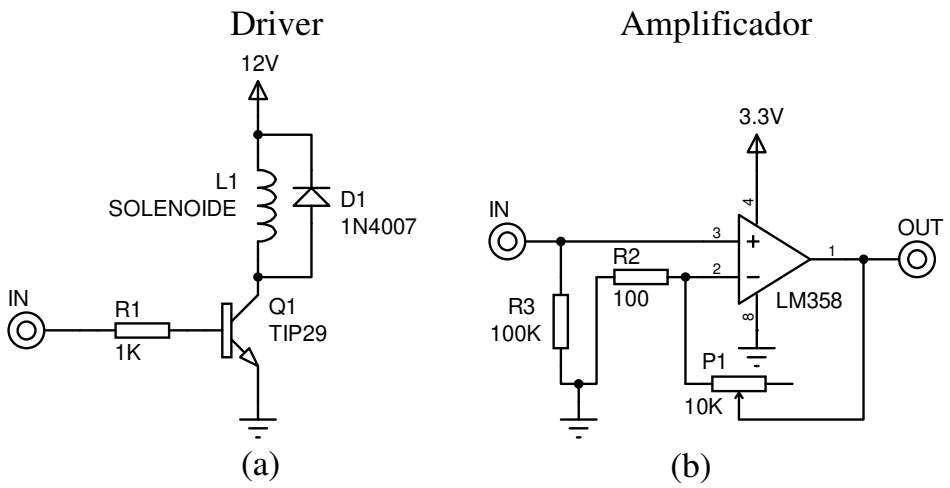

Figura 21 – (a) Circuito *driver* e (b) circuito amplificador

Fonte: Autor

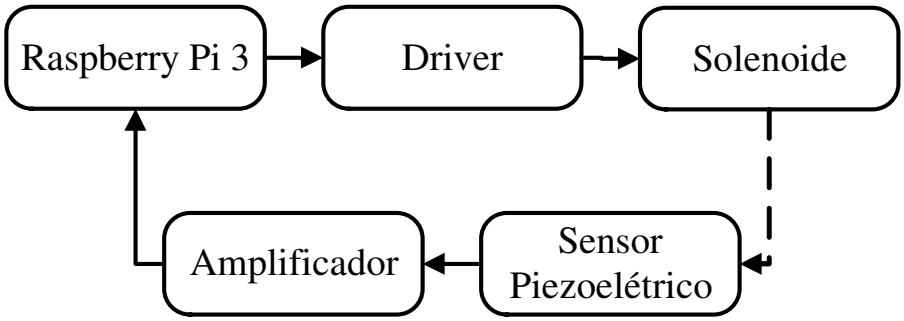

Figura 22 – Diagrama de blocos representando as estruturas do BongoBot

Fonte: Autor

3.7.1 Formato das Mensagens do BongoBot

O bongô robótico possui duas ações: tocar o tambor maior denominada *playHembra* (Quadro 13) e tocar o tambor menor *PlayMacho* (Quadro 14), onde ambas ações possuem os mesmos parâmetros e estes são:

- identificador: identificador da mensagem;

- solenoide: qual solenoide tocar, onde 0 representa o solenoide central e 1 o solenoide lateral;
- intensidade: a intensidade da batida representada por um valor real entre 0 e 1, onde 0 é a intensidade mais fraca e 1 a mais forte.

Quadro 13 – Formato da Mensagem tocar o tambor maior (*hembra*).

Componente	Tipo
/Endereço OSC do robô/PlayHembra	Endereço OSC
Identificador	Inteiro
Solenoide	Inteiro
Intensidade	Ponto flutuante

Quadro 14 – Formato da Mensagem tocar o tambor menor (*macho*).

Componente	Tipo
/Endereço OSC do robô/PlayMacho	Endereço OSC
Identificador	Inteiro
Solenoide	Inteiro
Intensidade	Ponto flutuante

3.8 Metodologia para Avaliação da Macrossincronização entre Relógios

Toda mensagem de ação proveniente do SSMM com destino a um robô contém um tempo absoluto para ser executada, assim, a sincronia entre os relógios dos robôs é de extrema importância para se obter sincronismo musical nas performances. Portanto, o SSMM funciona também como um relógio mestre, ou seja, um servidor SNTP para fornecer a hora para os robôs.

Para testar a sincronia entre dois relógios utilizou-se dois dispositivos Rasberry pi 3 model B, como mostra o *setup* de teste representado na Figura 23. O SSMM envia centenas de mensagens iguais para ambos dispositivos. Assim, Quando o relógio interno de algum Raspberry alcança a hora de execução da mensagem, o mesmo envia um pulso pela porta de saída (GPIO). Esses pulsos são lidos pelo Arduino que calcula a diferença de tempo entre dois pulsos (cada um proveniente de um Raspberry), e, assim, pode-se estimar a diferença entre os relógios internos dos dispositivos. Nesse *setup*, o Arduino

e o PC foram utilizados apenas para avaliar o atraso entre os dispositivos Raspberry. Dado que uma das premissas da plataforma RoboMus é o baixo custo, foi utilizado um roteador residencial de baixo custo Tp-Link TL-WR841ND[9] para conexão do SSMM com os dispositivos Raspberry.

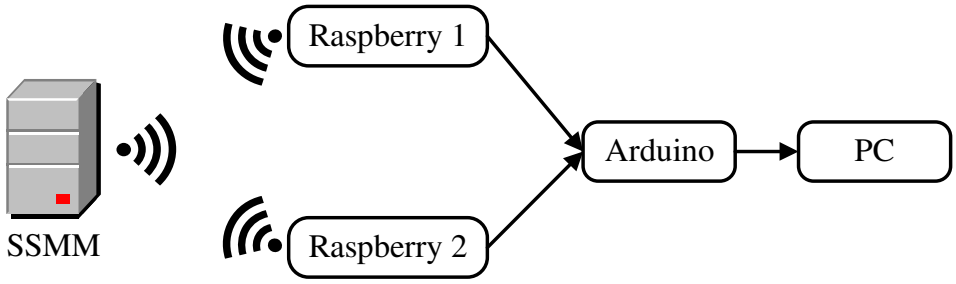

Figura 23 – Setup para estimação da diferença entre relógios

Fonte: Autor.

3.9 Metodologia para Avaliação Macrossincronização entre Robôs

Essa metodologia tem objetivo de avaliar a sincronia entre robôs da plataforma RoboMus. Entretanto, para efetuar tal avaliação são necessários, pelo menos, dois robôs. Assim, o BongoBot descrito na Seção 3.7 foi dividido em dois robôs seguindo a mesma proposta do BongoBot original, porém usando um Raspberry para controlar o tambor grande (robô 1 *hembra*) e outro Raspberry para controlar o tambor pequeno (robô 2 *macho*). Esses novos robôs seguem os formatos de mensagens definidos para o BongoBot.

O *setup* experimental para avaliação do atraso entre dois robôs está descrito na Figura 24, onde o Arduino e o PC fazem a função de avaliação do atraso, não sendo componentes dos robôs. Portanto, cada Raspberry recebe uma mensagem contendo uma determinada hora para ser executada (igual para os dois) e a executa quando seu relógio interno marcar a hora da mensagem. Desse modo, o robô irá enviar o comando para o solenoide bater no tambor e o Arduino Mega fará a leitura do sinal proveniente do sensor piezoelétrico de ambos os robôs visando estimar a diferença de tempo entre as batidas e, consequentemente, o atraso entre os robôs.

3.10 Metodologia para Avaliação da Microssincronização

A estratégia de microssincronização descrita na Subseção 3.5.2 tem o objetivo de fornecer um módulo perceptual autônomo ao robô, possibilitando o mesmo acompanhar musicalmente outro robô ou ser humano em tempo real. Assim, esta Seção visa

[9] <https://www.tp-link.com/br/support/download/tl-wr841nd/>

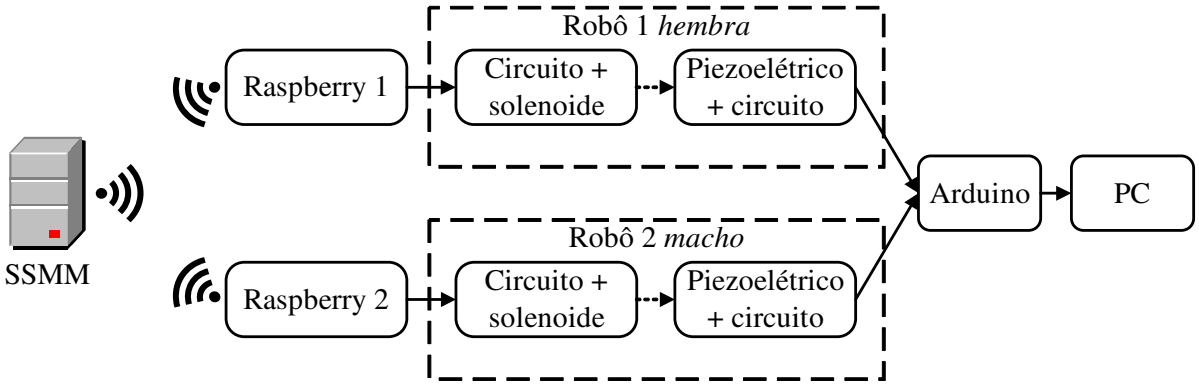

Figura 24 – Diagrama do *setup* experimental para estimar o atraso entre robôs bongô

Fonte: Autor.

descrever métodos para avaliação da microssincronização, fazendo observações numéricas e perceptuais.

3.10.1 Avaliação Numérica

A avaliação numérica da microssincronização utiliza o *setup* mostrado na Figura 25. Como fonte ativa de áudio (fonte para sincronização rítmica) foi utilizado um metrônomo digital executado por um *smartphone*, onde a saída de áudio do *smartphone* foi conectada via cabo[10] na interface de áudio USB (a)[11] do robô, que obteve o sinal de áudio do metrônomo e aplicou todo processo de microssincronização demostrado na Figura 18.

Nessa avaliação foi utilizado o BongoBot na sua forma completa como descrito na Seção 3.7. Então, para avaliar o atraso entre o som proveniente do robô e do metrônomo foi implementado a interface de áudio USB (b)[11] com dois canais, onde no canal 1 foram conectados os sensores piezoelétrico instalados no bongô e no canal 2 o som do metrônomo. Dado que ambos os sinais são amostrados pela interface ao mesmo tempo, para avaliar a diferença temporal entre a batida do metrônomo e a batida do bongô foi utilizado o algoritmo de detecção de picos, descrito na Subseção 2.4.1, onde para os parâmetros de entrada pre_max, $post_max$, pre_avg, $post_avg$ foram utilizados valores iguais a 1000 amostras (22,67ms a uma taxa de amostragem de 44100 amostra), $delta = 0{,}1$ e o $wait =$ 4410 amostras (100ms a uma taxa de amostragem de 44100 amostra). O parâmetro de entrada $wait$ evita a detecção de dois picos em uma mesma batida, pois a duração da batida do bongo é inferior a 100ms, além de funcionar teoricamente para até 600 BPM (1 batida a cada 100ms). Nos testes será usado no máximo 120 BPM, onde o intervalo entre

[10] Em primeiro momento visa-se estudar o comportamento do algoritmo para um sinal de áudio com baixa interferência, dado que o módulo perceptual do robô pode conter um microfone para "ouvir" toda a performance musical e se sincronizar, assim a conexão direta via cabo foi utilizada.

[11] Utilizou-se uma interface de baixo custo behringer UCA222 <https://www.behringer.com/Categories/Behringer/Computer-Audio/Interfaces/UCA222/p/P0A31>.

batidas é de 500ms. Assim, aplica-se o algoritmo para cada canal de forma independente considerando o momento de maior amplitude do sinal de áudio como o tempo da batida. Após o processo é possível calcular a diferença temporal entre batidas correspondente.

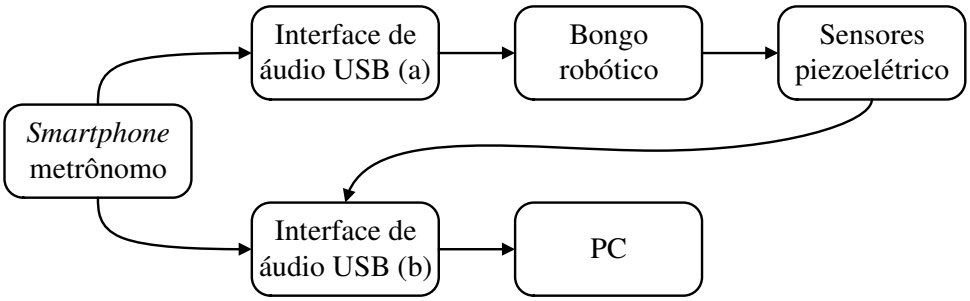

Figura 25 – Diagrama do *setup* experimental para avaliação do atraso dos robôs após a aplicação dos algoritmos de microssincronização

Fonte: Autor.

3.10.2 Avaliação Perceptual

A avaliação dos resultados da microssincronização também pode ser feita por meio da percepção sonora dos seres humanos. Assim, foi montado um questionário *online* anônimo para que as pessoas pudessem acessar, ouvir os áudios e votar sobre o sincronismo. Foi utilizado um banco de 20 amostras de áudio[12] de 10 segundos cada, gravadas de acordo com o *setup* experimental demonstrado na Subseção 3.10.1. Escolheu-se a duração de 10 segundos para evitar um grande acumulo do atraso que será descrito na Subseção 4.4.1, bem como evitar o desconforto dos participantes frente a áudios longos. O questionário foi dividido em duas partes, uma para identificação do perfil contendo 2 questões: *1) Qual a sua idade?* e *2) Qual a sua experiência com atividades relacionadas a música (audição, composição, performance, etc)?*, onde para a questão 2 as seguintes opções de resposta foram apresentadas: *1 (Nenhuma); 2 (Pouca); 3 (Média); 4 (Muita) e 5 (Bastante)*. Já para a avaliação de cada áudio foi feita a seguinte pergunta: *Você considera que os pulsos sonoros nesse áudio estão em sincronia?*, com as opções de resposta *sim* e *não*. Por tanto, o questionário completo foi composto por 10 perguntas, sendo 2 questões para avaliação do perfil e 8 para a avaliação das 8 amostras de áudio, sendo essas amostras selecionadas aleatoriamente.

3.11 Resumo do Capítulo

Este Capítulo apresentou a plataforma para performances musicais robóticas RoboMus, que visa a construção de robôs de baixo custo e o desenvolvimento de protocolos

[12] As amostras de áudio estão disponíveis em: <https://bit.ly/3cgz2hg>

para controle dos robôs por interfaces de interação. A RoboMus define formatos de mensagens musicais utilizando o protocolo OSC (*Open Sound Control*) para comunicação dos robôs com o Servidor de Sincronização de Mensagens Musicais (SSMM), bem como os formatos para comunicação entre o SSMM e as interfaces de controle.

Estratégias de sincronização para a RoboMus também foram apresentadas. A macrossincronização atende a demanda de sincronização entre os relógios dos componentes da plataforma, onde o SSMM implementa o SNTP (*Simple Network Time Protocol*) para fornecer essa sincronia. Ainda na macrossincronização foi idealizado o uso de redes neurais para aprendizado e compensação dos atrasos mecânicos presentes nos robôs. A microssincronização idealizou a aplicação de processamento de sinais para extração do BPM (Batidas Por Minuto), da sequência de pulsos musicais e da assinatura de tempo para que o robô possa se sincronizar ritmicamente, e de forma autônoma, com outros robôs ou humanos. Além disso, foi apresentado sobre a construção de um bongô robótico. Por fim, foram apresentadas metodologias para avaliação da macrossincronização e da microssincronização.

4 Resultados Experimentais

Este Capítulo visa avaliar as estratégias de micro e macrossincronização propostas no Capítulo anterior. Assim, inicia-se caracterizando o atraso mecânico presente no robô BongoBot, logo após é apresentado a sincronização entre os relógios de dois dispositivos Raspberry pi. Também é demonstrado a aplicação de redes neurais para aprendizado e compensação do atraso mecânico incluindo testes para dois robôs e, por fim, a sincronização rítmica em tempo real entre um robô e um metrônomo.

4.1 Caracterização do Atraso Mecânico do BongoBot

O atraso mecânico do robô BongoBot (Seção 3.7) está diretamente ligado à distância entre a ponta do solenoide e a pele do bongô e, principalmente, a velocidade da haste do solenoide. A velocidade dessa haste é dependente da tensão aplicada no solenoide, assim, foi usado o PWM (*Pulse Width Modulation*) para controle da tensão média e, consequentemente, controle da velocidade da haste e da força da batida.

O atraso mecânico do robô é contabilizado do momento em que o Raspberry envia seus sinais elétricos até a efetiva resposta sonora, que é capturada através do sensor piezoelétrico pelo mesmo Raspberry (Figura 22). Assim, a Figura 26 (a) mostra o perfil de atraso do solenoide central e a Figura 26 (b) demonstra o perfil de atraso do solenoide lateral (mais próximo a extremidade) do tambor maior (chamando de *hembra*), variando valores do PWM de 0 à 1023, ou seja, de 0 à 12 Volts médios. Já a Figura 27 mostra o perfil de atraso para o tambor menor (chamado de *macho*), onde a Figura 27 (a) mostra os atrasos do solenoide central e a Figura 27 (b) demonstra os atrasos do solenoide lateral.

Nas Figuras 26 e 27 é possível notar que o atraso máximo ficou entorno de 200ms, esse valor representa o tempo máximo de espera da resposta sonora pelo Raspberry, logo não houve uma batida, ou a mesma foi extremamente fraca, mostrando uma região de inatividade. Em geral, todos os solenoides iniciam sua região de atividade em torno do 400, entretanto essa desigualdade da região inativa acontece por causa da diferença na distância entre o solenoide e a pele do bongô. Porém, isso não impacta na performance, pois o atraso pode ser corrigido de acordo com cada solenoide.

4.2 Macrossincronização entre Relógios

Com objetivo de avaliar a sincronização entre os relógios de dois dispositivos Raspberry Pi, o *setup* descrito na Seção 3.8 foi utilizado. O primeiro teste foi efetuado com

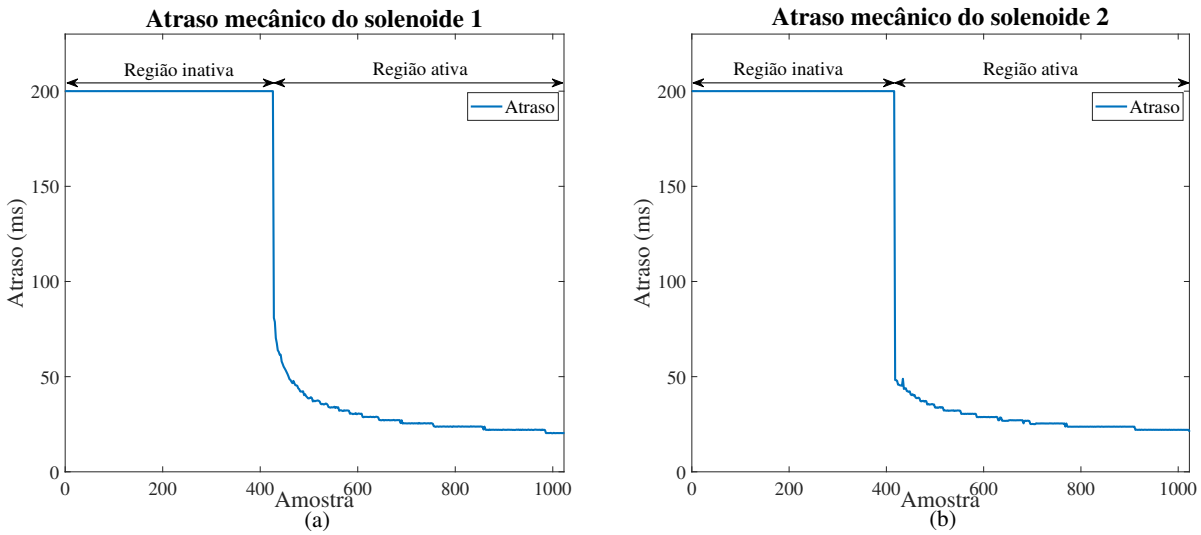

Figura 26 – Atraso mecânico do robô BongoBot em relação ao tambor maior (*hembra*): (a) o perfil de atraso do solenoide central e (b) o perfil do solenoide lateral

Fonte: Autor.

a finalidade de se observar um efeito conhecido como escorregamento de relógio (*clock drift*) que faz o mesmo não contabilizar a hora exata ao longo do tempo devido a imprecisão dos seus componentes. Assim, efetuou-se um teste de cerca de 8 minutos com 1000 mensagens utilizando dois dispositivos Raspberry. No entanto, a sincronização dos relógios foi efetuada apenas no início do teste, logo, pode-se observar nos resultados mostrado na Figura 28 (a) que houve um aumento constante do atraso entre os dispositivos, o que é causado pelo

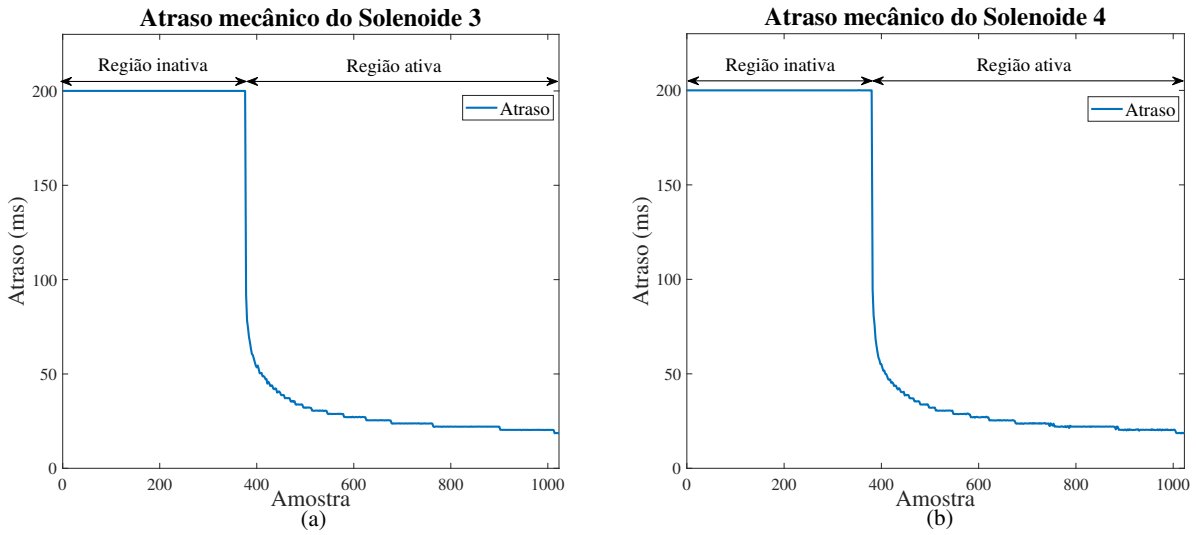

Figura 27 – Atraso mecânico do robô BongoBot relacionado ao tambor menor (*macho*): (a) o perfil de atraso do solenoide central e (b) o perfil de atraso do solenoide lateral

Fonte: Autor.

escorregamento de relógio.

O problema de escorregamento de relógio pode ser resolvido fazendo uma nova sincronização de relógio de tempos em tempos. Com objetivo de estudo do comportamento foi efetuado um teste sincronizando os relógios a cada 30 segundos (valor definido empiricamente). Assim, seguindo o *setup* e sincronização de relógio supracitados, a Figura 28 (b) mostra o resultado para 1000 mensagens, onde as transições abruptas representam o momento de sincronização do relógio com o servidor SNTP, onde pode-se notar oscilações proveniente da imprecisão do algoritmo. A média do atraso foi igual a $7,82ms$ e o desvio padrão igual a $5,55ms$, onde o atraso máximo foi de $18,81ms$. Todos os atrasos tiveram valores abaixo da percepção sonora humana ($20ms$), mostrando bons resultados preliminares para a macrossincronização. Além disso, um ajuste fino no atraso pode ser empregado por técnicas de microssincronização[1], por exemplo, módulos perceptuais para que o robô possa captar sons de outros robôs e identificar a falta de sincronismo, para, assim, tomar decisões de ressincronizar o seu relógio.

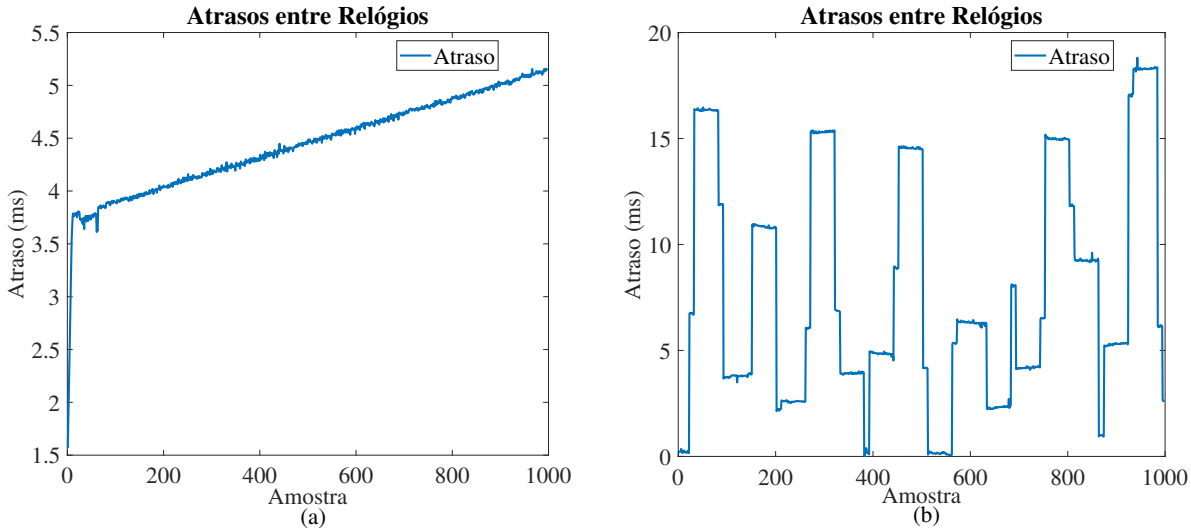

Figura 28 – Atrasos entre os relógios de dois dispositivos Raspberry: (a) mostra o resultado de um teste promovendo a sincronização de relógio apenas no início e (b) demostra o resultado do atraso fazendo uma sincronização a cada 30 segundos

Fonte: Autor.

4.3 Macrossincronização entre Robôs Musicais

Esta Seção visa avaliar o desempenho de sincronização entre robôs aplicando o SNTP em conjunto com redes neurais para aprendizado do atraso mecânico dos robôs. Por tanto, essa avaliação faz uso do *setup* experimental descrito na Seção 3.9.

[1] a implementação de estratégias para ajuste fino foge do escopo desse trabalho

A caracterização do atraso dos solenoides descritos na Seção 4.1 mostra uma similaridade nos atrasos dos solenoides. Isto posto, para o estudo da macrossincronização foi determinado que o robô 1 *hembra* tocasse com intensidade alta de 0,8 à 1 (alto PWM), provocando um atraso menor, e o robô 2 *macho* tocasse com intensidade de 0 à 0,2 (baixo PWM), provocando um atraso mecânico maior, ou seja, forçando uma possível dessincronia entre os robôs. Assim, seguindo essa proposta, e o *setup* da Figura 24, foram ensaiados dois testes com 1000 mensagens geradas aleatoriamente para cada robô, onde cada mensagem continha a mesma hora de execução e intensidades diferentes. Ambos os teste foram executados com as mesmas mensagens. A Figura 29 (a) mostra os resultados do primeiro teste com um atraso médio de $10,88ms$ entre os robôs e desvio padrão de $5,75ms$, onde $8,50\%$ das batidas tiveram atrasos acima de $20ms$ (percepção sonora humana). Já os resultados do segundo teste são demonstrados na Figura 29 (b), onde apresentam um atraso médio de $9,15ms$ com desvio padrão de $5,17ms$ e $2,2\%$ das batidas acima dos $20ms$.

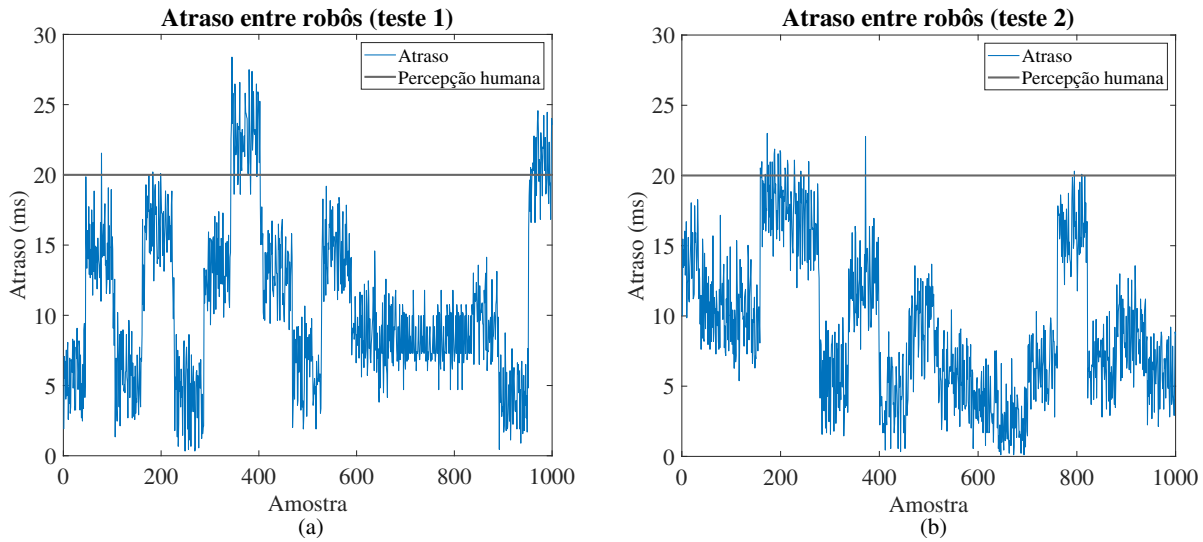

Figura 29 – Atrasos entre os dois robôs bongô: (a) apresenta os resultados para o teste 1 e (b) os resultado para o teste 2

Fonte: Autor.

A Subseção 3.5.1 apresenta uma estratégia para aprendizado e compensação do atraso mecânico utilizando redes neurais artificiais. Assim, foi usado a rede neural *perceptron multilayer* (HAYKIN, 2001) para aprendizado do perfil de atraso dos dois robôs. Devido a similaridade entre os robôs, a mesma estrutura da rede neural foi utilizada para cada um, porém cada robô possui sua própria rede configurada e treinada de forma específica, o modelo da rede neural utilizado é mostrado na Figura 30.

A rede neural para o robô 1 *hembra* apresentou um RMSE (*Root Mean Square Error*) de 0,82%, onde os dados de treinamento são mostrados na Figura 31 (a). Já a rede

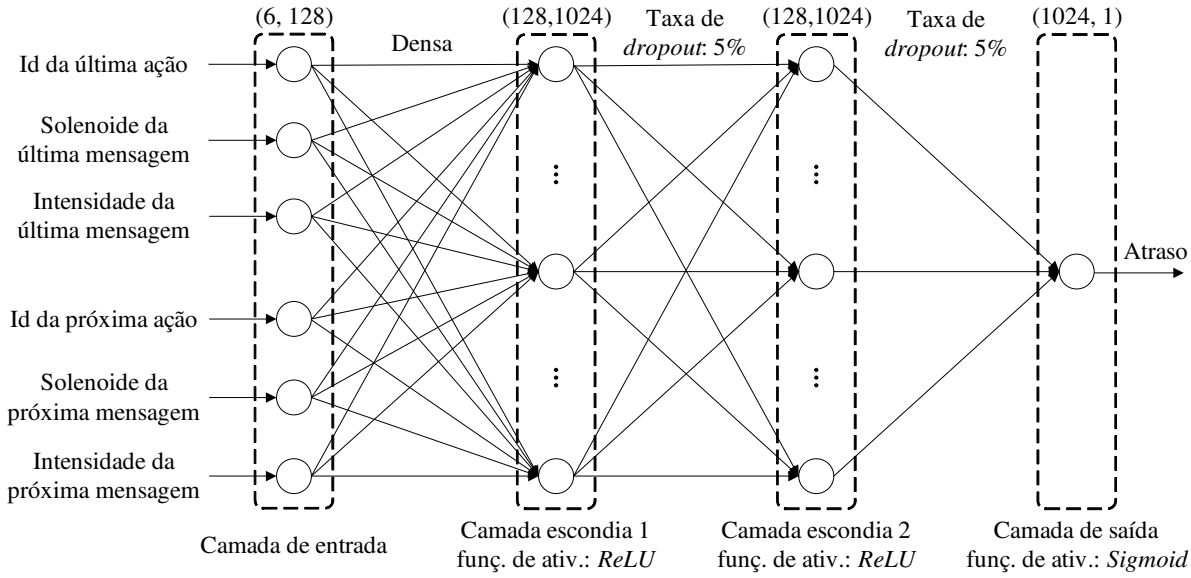

Figura 30 – Estrutura da rede neural usada especificamente para os dois bongôs robóticos

Fonte: Autor.

do robô 2 *macho* alcançou um RMSE de 1.2% e os dados de treinamento são mostrados na Figura 31 (b). Como pode-se observar na Figura 31, os erros dos dados de treinamento e de teste não se distanciaram muito ao longo do aprendizado, logo não houve *overfitting* no treinamento das redes, ou seja, as redes não se tornaram especialistas apenas no conjunto de treinamento. Após o treinamento das redes, alguns testes para previsão do atraso mecânico foram feitos, a Figura 32 (a) ilustra alguns valores reais e previstos, pela rede, do atraso para o robô 1 *hembra* e a Figura 32 (b) mostra alguns valores para o robô 2 *macho*.

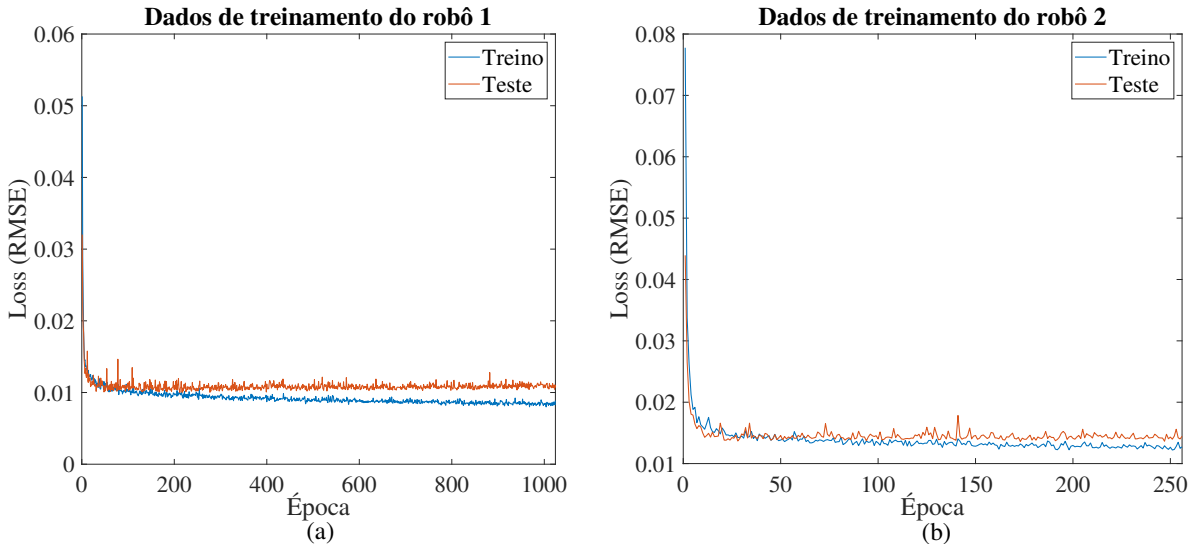

Figura 31 – Dados de treinamento das redes neurais: (a) mostra os dados de *loss* para o robô 1 *hembra* e (b) ilustra os dados de *loss* para o robô 2 *macho*

Fonte: Autor.

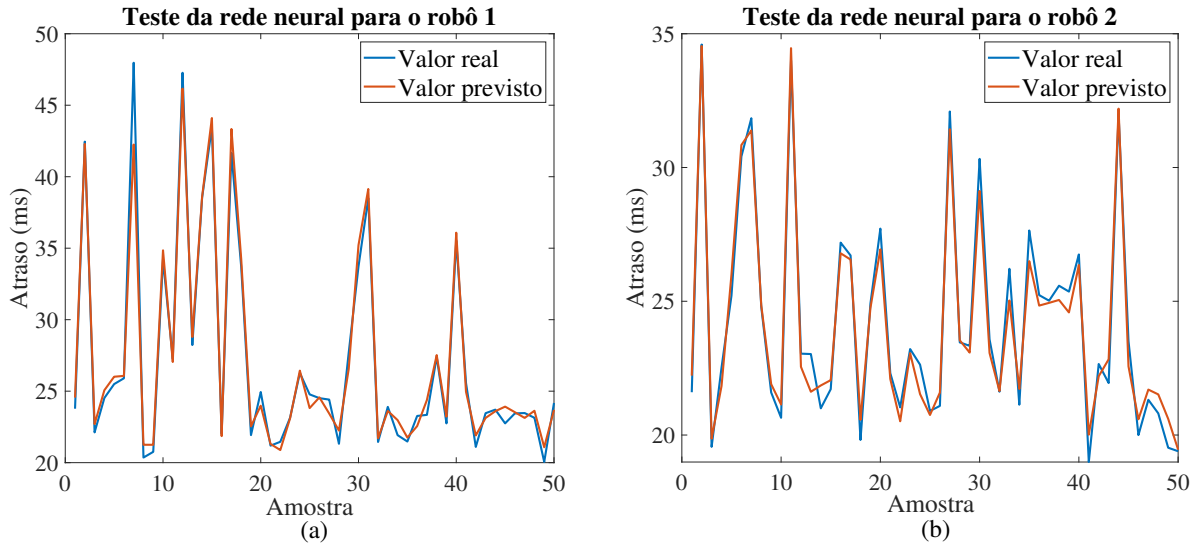

Figura 32 – Dados de teste das redes neurais: (a) mostra os dados de reais e previstos do atraso mecânico do robô 1 *hembra* e (b) ilustra os dados de reais e previstos do atraso mecânico do robô 2 *macho*

Fonte: Autor.

Com os robôs treinados, dois testes foram feitos aplicando a compensação prevista pela rede, ou seja, uma mensagem escalonada para ser executada às 10:00:00.000 (hora da resposta sonora) será processada pela rede neural instalada no SSMM e o resultado será subtraído da hora de execução, por exemplo, caso o atraso previsto seja de $25ms$, o novo horário de execução será 9:59:59.975. Assim, dois testes foram efetuados com as mesmas mensagens utilizadas nos testes sem compensação mostrado na Figura 29, porém compensando o atraso mecânico. A Figura 33 (a) mostra os resultados do teste 3 para cerca de 1000 mensagens, onde foi alcançado um atraso médio entre os robôs de $5,58ms$ e desvio padrão de $3,75ms$, além de 0% dos valores acima da percepção humana ($20ms$). Já a Figura 33 (b) mostra os resultados do teste 4, onde todos os valores ficaram abaixo da percepção humana e o atraso médio foi de $4,86ms$ com desvio padrão de $3,88ms$.

A Tabela 1 mostra os resultados para os testes supracitados. Para os testes sem compensação pode-se notar valores diferentes apesar de ambos usarem as mesmas mensagens, isso pode ter sido causado pela variação do algoritmo SNTP, idem para a variação dos testes com compensação. Como era esperado, houve uma redução no atraso médio, desvio padrão e, principalmente, na quantidade de batidas acima do perceptual sonoro humano. Comparando o teste 1 como o teste 4 houve uma redução de $55,33\%$ ($6,02ms$) do atraso médio e uma redução de $32,52\%$ ($1,87ms$) do desvio padrão, além de zerar as batidas acima de $20ms$. As comparações entre todos os testes encontram-se na Tabela 2, onde mostra-se a redução em valor percentual e absoluto do atraso médio e desvio padrão dos testes 1 e 2 (sem compensação) para com os testes 3 e 4 (com compensação).

A aplicação das técnicas de macrossincronização usando o algoritmo SNTP para

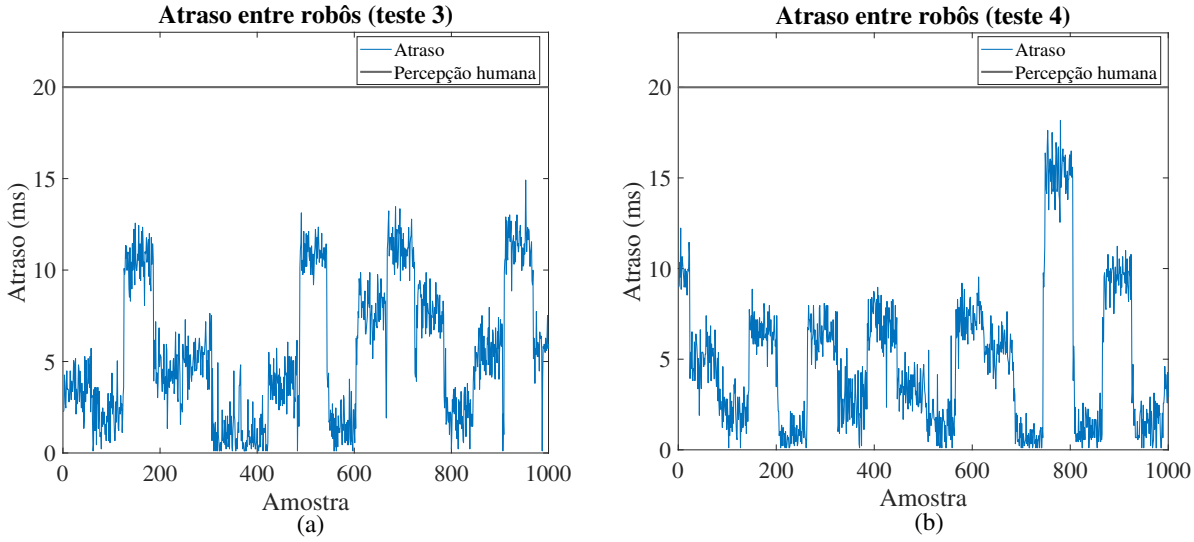

Figura 33 – Atrasos entre os dois robôs bongô: (a) apresenta os resultados para o teste 3 e (b) os resultado para o teste 4

Fonte: Autor.

Tabela 1 – Resultados dos testes para avaliar o atraso mecânico entre os robôs

	Sem compensação		**Com compensação**	
	Teste 1	**Teste 2**	**Teste 3**	**Teste 4**
Atraso médio entre os robôs (ms)	10,88	9,15	5,58	4,86
Desvio padrão do atraso (ms)	5,75	5,17	3,75	3,88
Valores acima de 20ms (%)	8,50	2,2	0	0

Tabela 2 – Comparação dos resultados após aplicação da compensação dos atrasos mecânicos por redes neurais

	Teste 3		**Teste 4**	
	Redução do atraso médio	**Redução do desvio padrão**	**Redução do atraso médio**	**Redução do desvio padrão**
Teste 1	48,71% (5,30ms)	34,78% (2,00ms)	55,33% (6,02ms)	32,52% (1,87ms)
Teste 2	39,02% (3,57ms)	27,47% (1,42ms)	46,89% (4,29ms)	24,95% (1,29ms)

sincronia de relógio e redes neurais para compensação do atraso mecânico presente nos robôs mostraram resultados satisfatórios, uma vez que as batidas acima do limiar da percepção sonora humana foram zeradas nos testes com compensação apresentados, além das reduções nos atrasos médios e desvios padrão. Entretanto, em geral, o desvio padrão apresentou um alto valor se comparado à média, o que mostra alta variação percentual em relação a média. Porém, após a compensação do atraso esse desvio padrão foi reduzido e não apresentou problema para a sincronia dos robôs, pois mesmo com a variação os

valores ainda ficaram abaixo da percepção sonora humana. A alta variação provavelmente é proveniente da variação do SNTP demonstrado na Figura 28 (b), pois a rede neural compensou apenas o atraso proveniente dos componentes mecânicos. Além disso, para se obter melhores resultados, estratégias de microssincronização podem fornecer um ajuste fino de sincronização.

4.4 Microssincronização por Pulsos Musicais

De acordo com a estratégia de microssincronização por pulsos musicais explicada na Subseção 3.5.2, esta Seção visa avaliar a aplicação dessa estratégia ao BongoBot seguindo a metodologia proposta na Seção 3.10.

4.4.1 Avaliação Numérica

Para essa avaliação, seguindo a metodologia descrita na Subseção 3.10.1, foi fixado apenas um nível de força do solenoide para evitar variação no atraso mecânico, pois foi efetuado uma calibragem manual do atraso do sistema com um todo, onde através da interface de áudio (b) foram gravados tanto os áudios do bongô quanto o do metrônomo em canais diferentes e sem compensar nenhum atraso. Após essa etapa, o atraso foi estimado manualmente e compensado via software no bongô. Tal atraso perpassa o atraso mecânico do solenoide supracitado, onde inclui-se também atraso da placa de áudio, *buffers*, da comunicação USB, dos *drivers* do sistema operacional e etc.

A microssincronização foi feita apenas no início do teste, onde o robô armazenou ("ouviu") 8 segundos de dados sonoros e aplicou os algoritmos. Visando melhorar a precisão dos algoritmos foi utilizado uma janela da STFT de 64 amostras a uma taxa de amostragem de 44.100 amostras/segundo, ou seja, uma janela de $1,45ms$. Em contrapartida, a redução do tamanho da janela aumenta o número de janelas no sinal e, por conseguinte, o tempo de processamento. Para avaliação dos dados de microssincronização foram utilizados um notebook com *core i7* e um Arduino para controle do robô, utilizando a arquitetura de controle descrita na Figura 19 (c). Com essa arquitetura o tempo de processamento ficou em torno de 7 segundos. Sendo assim, após armazenar os 8 segundos de áudio, o robô ainda gastou em torno de 7 segundos para começar seus movimentos mecânicos.

Para a avaliação numérica foram executados 10 testes com um minuto de áudio cada, seguindo o *setup* demonstrado na Figura 25. Na Tabela 3 mostra-se o BPM do metrônomo e o BPM estimado pelo algoritmo, bem como o módulo da diferença entre eles. Além disso, foram calculados a média e desvio padrão das diferenças temporais entre batidas correspondentes do metrônomo e do bongô. Na Figura 34 são mostrados os 10 primeiros segundos do sinal de áudio de ambos os dispositivos com as respectivas batidas.

Tabela 3 – Resultados experimentais dos testes de microssincronização por pulsos musicais.

Teste	BPM do metrônomo	BPM estimado	Módulo do erro do BPM	Atraso médio	Desvio padrão
teste1	110	109,96	0,04	5,68 ms	4,12 ms
teste2	70	69,96	0,04	11,58 ms	6,94 ms
teste3	80	79,97	0,03	9,36 ms	5,23 ms
teste4	60	60,01	0,01	14,25 ms	3,70 ms
teste5	120	119,84	0,16	35,39 ms	16,29 ms
teste6	90	90,07	0,07	37,88 ms	9,55 ms
teste7	60	60,01	0,01	13,55 ms	4,41 ms
teste8	75	75,03	0,03	21,91 ms	6,79 ms
teste9	100	100,11	0,11	47,51 ms	12,52 ms
teste10	80	79,97	0,03	7,37 ms	4,72ms

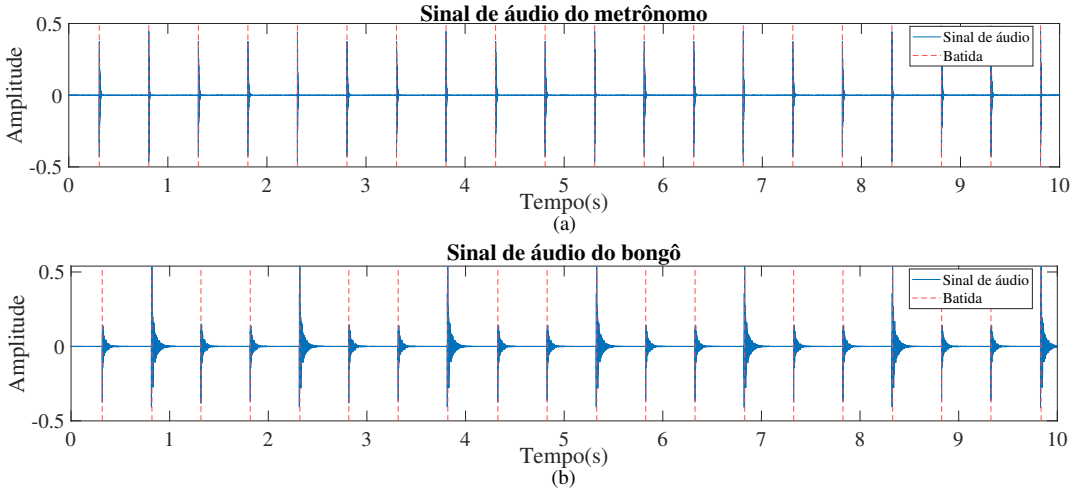

Figura 34 – Gráficos com os 10 primeiros segundos dos sinais de áudio e batidas detectadas: (a) representa os sinais proveniente do metrônomo e (b) os sinais do bongô robótico

Fonte: Autor.

A imprecisão na estimativa do BPM pode causar sérios problemas na performance, o RMSE entre o BPM do metrônomo e o BPM estimado foi de 6,98%. Tomando como exemplo o teste 5, um dos piores atrasos médios, a Figura 35 mostra os 10 últimos segundos do teste, onde pode-se notar que a batida do bongô passou a ocorrer após a batida do metrônomo. Isso ocorreu devido a diferença entre o BPM real e o BPM estimado. Para uma velocidade de 120 batidas por minuto uma batida ocorre a cada 500ms, já para 119,84 batidas por minuto uma batida ocorre a cada 500,668ms. Assim, a cada batida o robô acumula 0,668ms de atraso, resultando na dessincronização ao longo do tempo, onde na Figura 36 pode-se observar o crescimento do atraso no decorrer do tempo.

Outro fator impactante no atraso foi o tempo de processamento, pois o robô só

entra em ação, em média, 7 segundos após gravar o áudio (tempo de processamento). Então, o robô entra em cena de acordo com a última informação que ele possui, uma informação que aconteceu a 7 segundos atrás. Nesses tempo acumulam-se atrasos devido a imprecisão citada acima. Para o exemplo anterior, teste 5, com metrônomo a 120 BPM, pode-se estimar que ele bateu 14 vezes nos 7 segundos. Como em média é acumulado $0,668ms$ por batida, tem-se então uma acumulo estimado de $14 \times 0,668 = 9,35ms$, ou seja, ao esperar o processamento e estimar o BPM com erro, o robô entrou na performance com atraso de no mínimo $9,35ms$. Uma ressincronização de tempos em tempos pode resolver o problema do acumulo de atraso, no entanto, ainda acumula-se o atraso devido ao tempo de processamento da ressincronização.

Então, faz-se uma análise para os mesmos testes, porém considerando apenas os 10 primeiros segundos, visando amenizar o atraso acumulativo, a Tabela 4 demonstra os resultados. Pode-se notar que o teste 5 não apresentou o pior atraso médio para essa avaliação, já outros testes apresentaram atraso acima do limiar de 20 ms. Esse atraso é proveniente da imprecisão do estimador de BPM, acumulando atraso no tempo de processamento, nos 10 segundos de áudio e, possivelmente, na variação dos atrasos provenientes dos componentes de software e hardware para aquisição de dados. Assim, dos 10 testes, 2 (20%) apresentaram atraso médio acima da percepção sonora humana.

Tabela 4 – Resultados experimentais dos testes de microssincronização por pulsos musicais considerando apenas os 10 primeiros segundos.

Teste	**Atraso médio**	**Desvio padrão**
teste1	3,58	2,00
teste2	4,65	3,20
teste3	5,04	1,80
teste4	10,71	4,71
teste5	16,58	5,60
teste6	24,82	3,98
teste7	10,44	3,37
teste8	12,91	1,52
teste9	31,71	5,09
teste10	4,82	1,58

Essa estratégia de microssincronização apresentou inúmeras dificuldades ao projeto, elenca-se principalmente a imprecisão na estimativa do BPM e o elevado tempo de processamento. Certamente, ajustes nos algoritmos para embutir processamento paralelo e o uso de computadores mais robustos podem ajudar na solução do problema, ou até mesmo, buscar outros algoritmos menos complexos e mais precisos.

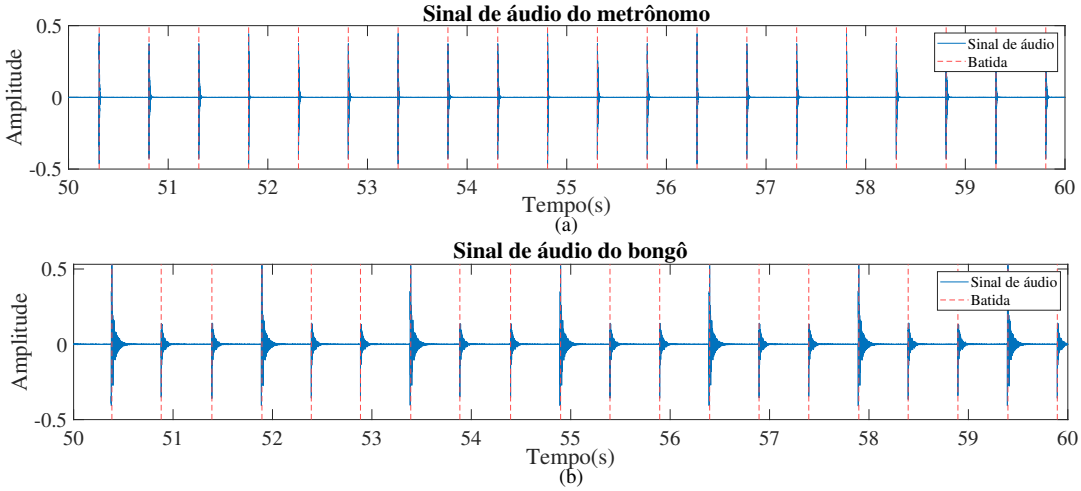

Figura 35 – Gráficos com os 10 últimos segundos dos sinais de áudio e batidas detectadas: (a) representa os sinais do metrônomo e (b) os sinais do bongô robótico

Fonte: Autor.

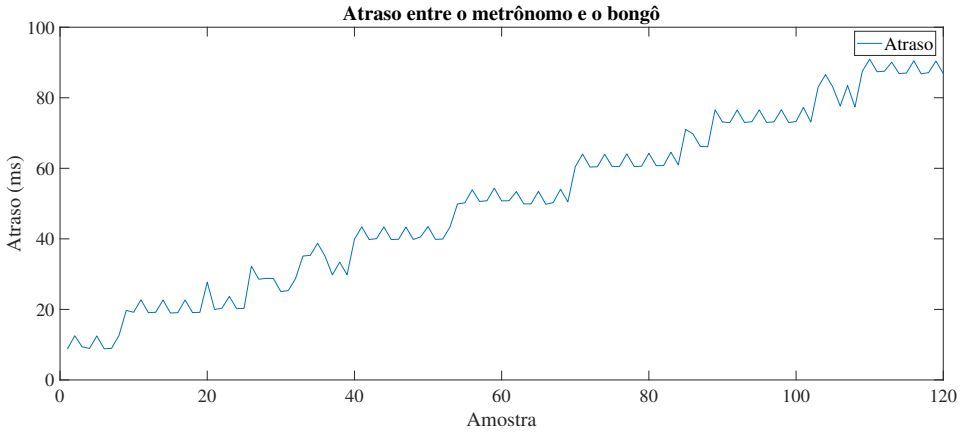

Figura 36 – Atraso entre o metrônomo e o bongô robótico

Fonte: Autor.

4.4.2 Avaliação Perceptual

A avaliação perceptual seguiu o modelo apresentado na Subseção 3.10.2. No total 121 pessoas responderam o questionário, onde a pessoa mais nova apresentou idade igual a 18 anos e a mais velha idade igual a 60. Entretanto, houve uma concentração maior do público mais jovem na pesquisa. Assim, não será feito análise considerando a idade devido a esse fator. A Tabela 5 e a Figura 37 mostram os resultados gerais dos votos para cada amostra de áudio, além disso foram calculados o atraso médio e o desvio padrão seguindo o procedimento descrito na Subseção 4.4.1.

A Figura 38 demonstra um gráfico do atraso médio calculado pelo percentual de votos sim dos testes. Além disso, foi incluído uma reta determinada pelo método de regressão linear simples, onde é possível observar uma inclinação decrescente. Assim,

Tabela 5 – Resultados perceptuais e calculados para os testes de microssincronização por pulsos musicais.

Áudio	**Total de votos**	**Votos sim (%)**	**Votos não (%)**	**Atraso médio calculado (ms)**
teste1	43	72,09	27,91	7,8 ± 4,12
teste2	47	74,47	25,53	10,61 ± 3,35
teste3	62	83,87	16,13	13,91 ± 3,06
teste4	43	76,74	23,26	4,53 ± 3,76
teste5	33	78,79	21,21	14,31 ± 3,72
teste6	53	47,17	52,83	20,3 ± 3,97
teste7	44	72,73	27,27	11,41 ± 6,2
teste8	43	81,40	18,60	4,28 ± 2,76
teste9	43	60,47	39,53	11,54 ± 3,04
teste10	51	68,63	31,37	2,61 ± 1,8
teste11	51	80,39	19,61	4,67 ± 2,47
teste12	41	65,85	34,15	14,09 ± 3,74
teste13	48	85,42	14,58	3,28 ± 2,73
teste14	61	60,66	39,34	4,53 ± 2,83
teste15	43	58,14	41,86	18,51 ± 4,25
teste16	64	60,94	39,06	13,13 ± 4,2
teste17	59	61,02	38,98	12,47 ± 4,04
teste18	43	55,81	44,19	23,5 ± 5,03
teste19	57	80,70	19,30	2,31 ± 1,74
teste20	31	77,42	22,58	4,66 ± 3,25

pode-se concluir que quanto maior o atraso médio entre o robô e o metrônomo, menor é a aceitação do público. A fórmula da reta é dada por:

$$y(x) = -1,13x + 81,52, \tag{4.1}$$

onde $y(x)$ representa a aceitação percentual e x o atraso médio.

A Tabela 6 descreve o percentual para determinado nível de aceitação, onde pode-se observar que 95% dos testes tiveram aceitação pela maioria, ou seja, aceitação acima de 50%. No entanto, as respostas tinham apenas duas opções e 50% de aceitação fica próximo de uma avaliação aleatória. Isto posto, neste trabalho considera-se aceitações acima 70%, onde 11 testes (55%) alcançaram acima desse nível de aceitação, mostrando um resultado razoável para as primeiras implementações dessa estratégia de microssincronização.

4.4.2.1 Avaliação Perceptual para o Grupo com Experiência 1 ou 2

O questionário identificou o grau de experiência musical dos participantes, visando realizar avaliações com diferentes níveis de confiabilidade. Assim, essa análise foi dividia em

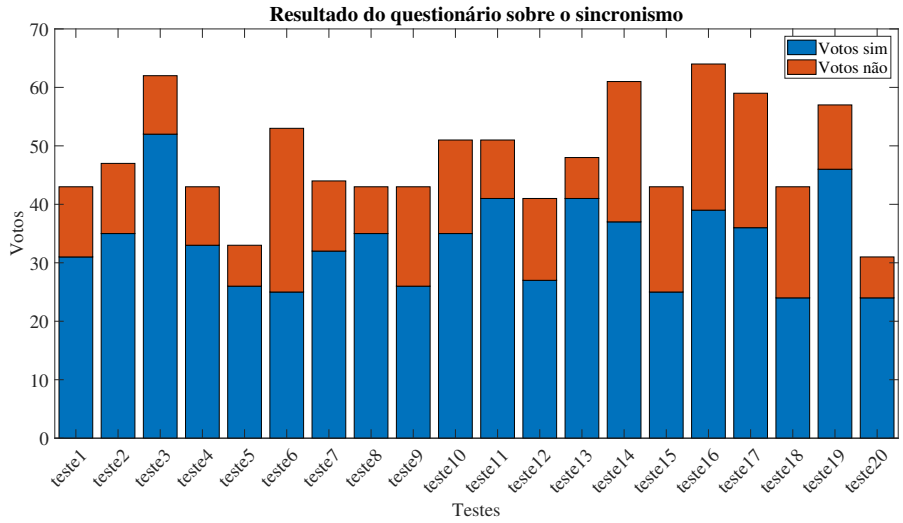

Figura 37 – Gráfico mostrando a quantidade absoluta de votos sim e não para cada teste

Fonte: Autor.

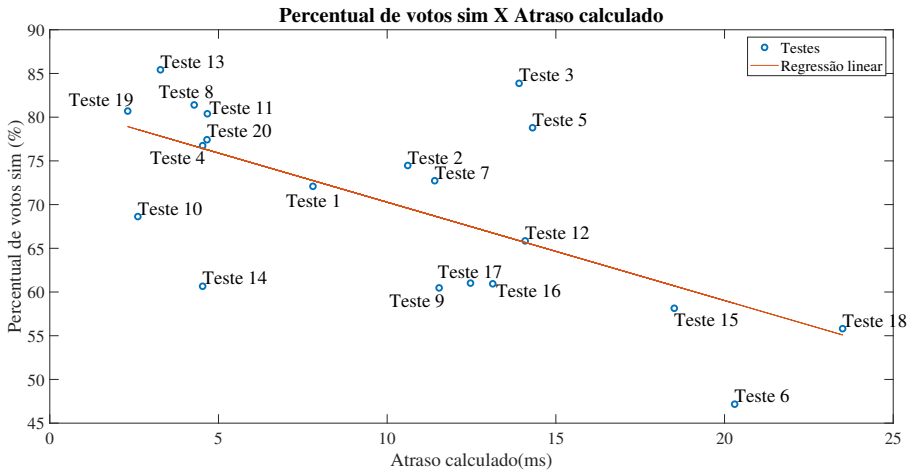

Figura 38 – Gráfico mostrando o percentual de votos sim versus o atraso médio calculado

Fonte: Autor.

dois grupos, um grupo considerando a experiência 1 ou 2 e outro grupo pra a experiência 4 ou 5, tomando a experiência 3 como neutra. Portanto, a Tabela 7 e a Figura 39 mostram os resultados para o grupo com experiência 1 (nenhuma) ou 2 (pouco).

O gráfico do percentual de votos sim versus o atraso médio calculado é demonstrado na Figura 40 para cada teste, bem como, uma reta calculada pelo método de regressão linear simples, seguindo uma inclinação negativa e demonstrando que a aceitação do público diminui quando o atraso médio aumenta. A formulação da reta é dado por:

$$y(x) = -1,16x + 82,93, \tag{4.2}$$

onde $y(x)$ representa a aceitação percentual e x o atraso médio.

Tabela 6 – Tabela demostrando a quantidade absoluta e percentual dos teste de acordo com o percentual de aceitação pelo público.

Testes	Quantidade de testes	Percentual de testes
Aceitação maior ou igual a 80%	5	25 %
Aceitação maior ou igual a 70%	11	55 %
Aceitação maior ou igual a 60%	17	85 %
Aceitação maior ou igual a 50%	19	95 %
Aceitação abaixo de 50%	1	5 %

Tabela 7 – Resultados perceptuais para os testes de microssincronização por pulsos musicais levando em consideração o grupo com experiência 1 ou 2.

Áudio	Total de votos	Votos sim (%)	Votos não (%)
teste1	20	70	30
teste2	23	65,22	34,78
teste3	42	78,57	21,43
teste4	17	88,24	11,76
teste5	15	86,67	13,33
teste6	31	45,16	54,84
teste7	25	72,00	28,00
teste8	27	81,48	18,52
teste9	26	65,38	34,62
teste10	29	62,07	37,93
teste11	29	89,66	10,35
teste12	25	72,00	28,00
teste13	22	86,36	13,64
teste14	34	61,76	38,24
teste15	22	63,64	36,36
teste16	41	60,98	39,02
teste17	42	64,29	35,71
teste18	26	53,85	46,15
teste19	37	83,78	16,22
teste20	11	72,73	27,27

O percentual de testes dado certo nível de aceitação é demonstrado na Tabela 8, onde pode-se observar que 95% dos testes foram aceitos pela maioria. No entanto, considerando uma aceitação de 70%, tem-se 55% dos testes acima desse nível.

4.4.2.2 Avaliação Perceptual para o Grupo com Experiência 4 ou 5

Visando uma análise com um grau de confiabilidade maior, considera-se o grupo em que as pessoas se declararam com experiência musical 4 ou 5. Então, os resultados dos votos são demonstrados na Tabela 9 e na Figura 41.

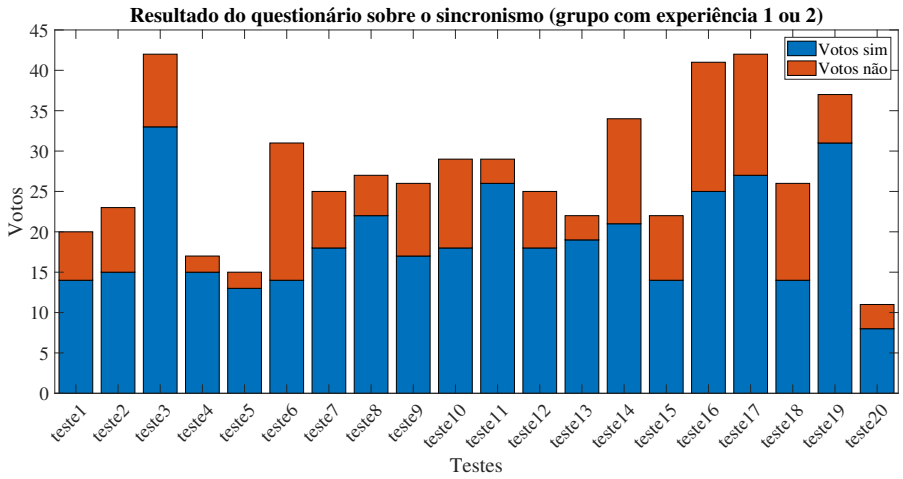

Figura 39 – Gráfico mostrando a quantidade absoluta de votos sim e não para cada teste considerando o grupo com experiência 1 ou 2

Fonte: Autor.

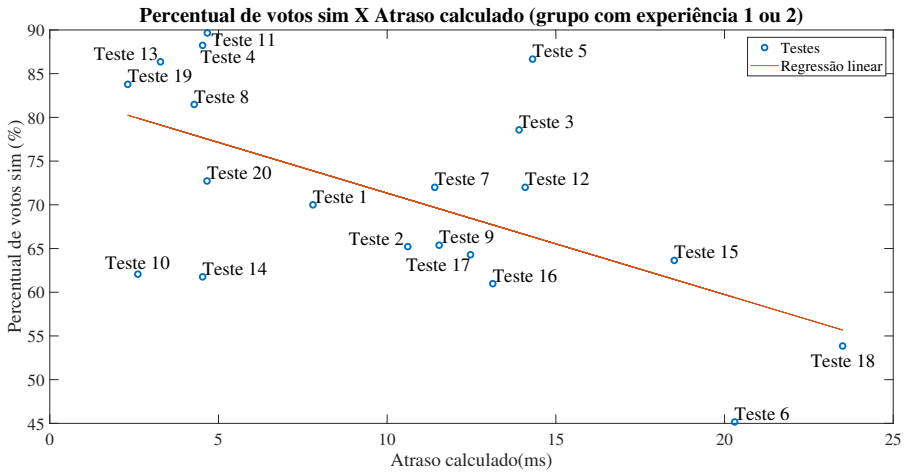

Figura 40 – Gráfico mostrando o percentual de votos sim, para o grupo com experiência 1 ou 2, pelo atraso médio calculado

Fonte: Autor.

Para melhor entender o comportamento dos votos sim frete ao atraso médio entre o robô e o metrônomo, a Figura 42 mostra um gráfico que relaciona essas duas características. Também foi traçada uma reta determinada por regressão linear simples, sua equação é dada por:

$$y(x) = -1,69x + 91,85, \tag{4.3}$$

onde $y(x)$ representa a aceitação percentual e x o atraso médio.

A Tabela 10 demonstra o percentual de testes de acordo com o nível de aceitação, tendo 90% dos teste com aceitação acima de 50%. Para o nível de 70% de aceitação considerado neste trabalho, tem-se 60% dos testes.

Tabela 8 – Tabela demostrando a quantidade absoluta e percentual dos teste de acordo com o percentual de aceitação pelo público do grupo com experiência 1 ou 2.

Testes	**Quantidade de testes**	**Percentual de testes**
Aceitação maior ou igual a 80%	6	30 %
Aceitação maior ou igual a 70%	11	55 %
Aceitação maior ou igual a 60%	18	90 %
Aceitação maior ou igual a 50%	19	95 %
Aceitação abaixo de 50%	1	5 %

Tabela 9 – Resultados perceptuais dos testes de microssincronização executados pelo grupo com experiência 4 ou 5.

Áudio	**Total de votos**	**Votos sim (%)**	**Votos não (%)**
teste1	14	85,71	14,29
teste2	14	85,71	14,29
teste3	15	93,33	6,67
teste4	10	90,00	10,00
teste5	6	50,00	50,00
teste6	16	43,75	56,25
teste7	8	87,50	12,50
teste8	6	83,33	16,67
teste9	5	60,00	40,00
teste10	12	83,33	16,67
teste11	9	88,89	11,11
teste12	8	50,00	50,00
teste13	9	88,89	11,11
teste14	11	63,64	36,36
teste15	9	66,67	33,33
teste16	12	41,67	58,33
teste17	9	66,67	33,33
teste18	4	75,00	25,00
teste19	3	100,00	0,00
teste20	12	91,67	8,33

A Tabela 11 apresenta os resultados comparativos entre o grupo geral, o grupo com experiência 1 ou 2 e com experiência 3 ou 4. Para o limiar de 70% de aceitação, pode-se notar que o grupo com maior experiência apresentou o maior percentual de testes, com o aumento de 5% ante ao grupo geral e ao grupo com experiência 1 ou 2. Além disso, considerando o limiar de 80% de aceitação, o aumento no percentual dos teste foi ainda maior, com 30% de aumento em relação ao grupo geral e 25% ante ao grupo com experiência 1 ou 2. Assim, mostrou-se maior aceitação pelas pessoas com mais conhecimento na área.

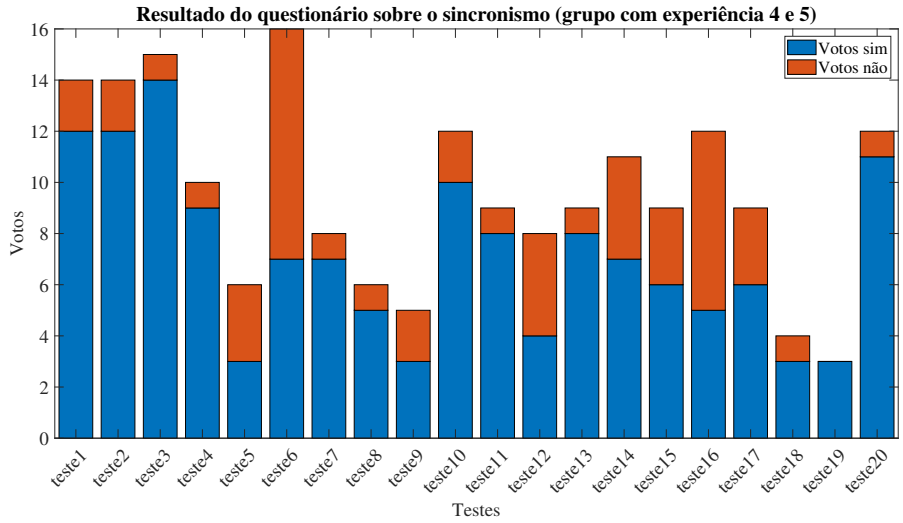

Figura 41 – Gráfico mostrando a quantidade absoluta de votos sim e não para cada teste considerando o grupo com experiência 4 ou 5

Fonte: Autor.

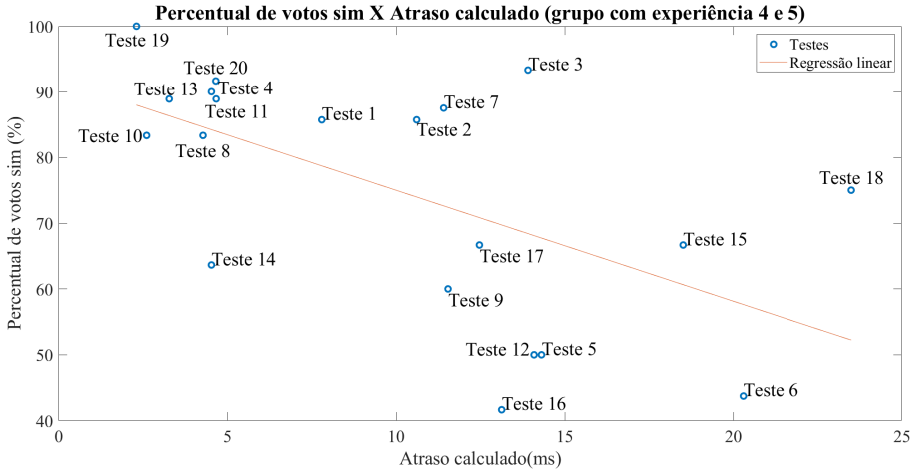

Figura 42 – Gráfico mostrando o percentual de votos sim, para o grupo com experiência 4 ou 5, pelo atraso médio calculado

Fonte: Autor.

4.5 Resumo do Capítulo

Este Capítulo avaliou os algoritmos de macrossincronização utilizando redes neurais para aprendizado e compensação do atraso mecânico aplicados a dois robôs. Ainda foi avaliado a microssincronização utilizando processamento de sinais para sincronização rítmica em tempo real, demonstrando resultados da sincronização do robô BongoBot com um metrônomo digital.

Nos testes de macrossincronização foram utilizados dois robôs, onde a rede neural

Tabela 10 – Tabela demostrando a quantidade absoluta e percentual dos teste de acordo com o percentual de aceitação pelo público do grupo com experiência 4 ou 5.

Testes	**Quantidade de testes**	**Percentual de testes**
Aceitação maior ou igual a 80%	11	55 %
Aceitação maior ou igual a 70%	12	60 %
Aceitação maior ou igual a 60%	16	80 %
Aceitação maior ou igual a 50%	18	90 %
Aceitação abaixo de 50%	2	10 %

Tabela 11 – Comparação entre os resultados do grupo geral, do grupo com experiência 1 ou 2 e com experiência 4 ou 5.

	Percentual de testes		
	Geral	Experiência 1 ou 2	Experiência 4 ou 5
Aceitação maior ou igual a 80%	25%	30%	55%
Aceitação maior ou igual a 70%	55%	55%	60%
Aceitação maior ou igual a 60%	85%	90%	80%
Aceitação maior ou igual a 50%	95%	95%	90%
Aceitação abaixo de 50%	5%	5%	10%

do robô 1 apresentou uma RMSE (*Root Mean Square Error*) de 0,82% e a do robô 2 um valor de 1,2%. Após a aplicação da compensação do atraso mecânico houve uma redução de 55,33% no atraso médio no melhor caso e para o pior caso uma redução de 39,02%.

Para a microssincronização, o robô "ouviu" o metrônomo por 8 segundos e aplicou os algoritmos de sincronização, onde a detecção de BPM apresentou um RSME de 6,98%. O tempo de processamento do robô ficou em torno de 7 segundos, assim, nesse tempo o robô acumulou atraso devido a imprecisão do BPM, pois o robô só atua no teste de acordo com informações de 7 segundos atrás. No total de 10 testes de 1 minuto cada, 4 apresentaram atrasos médio entre o robô e o metrônomo acima de 20ms (valor estimado da percepção sonora humana). Fazendo a mesma análise considerando apenas os 10 primeiros segundos dos testes, 2 ficaram com atraso acima de 20ms.

Na avaliação perceptual da microssincronização, pessoas responderam sobre a sincronização dos testes gravados em áudios de 10 segundos. Para a avaliação geral foi perguntado se o áudio estava sincronizado ou não, onde 55% dos testes apresentaram um percentual de votos positivos acima de 70%. Outro fato observado foi que as retas, determinadas pela regressão linear, mostraram que quanto maior o atraso entre o robô e o metrônomo, menor é a aceitação do público, como era esperado.

5 Conclusões

Este trabalho apresentou a RoboMus, uma plataforma para performances musicais robóticas em rede e controladas por seres humanos por meio de interfaces de interação. Também apresentou o desenvolvimento de instrumentos musicais robóticos, concentrando esforços especialmente no desenvolvimento de estratégias para sincronização entre os robôs (ou seres humanos e robôs).

O protocolo para comunicação das mensagens musicais escolhido foi o *Open Sound Control* (OSC), com esse protocolo foram definidos formatos de mensagem para comunicação da plataforma que seguiu um modelo cliente-servidor, utilizando um Servidor de Sincronização de Mensagens Musicais (SSMM). As mensagens foram pensadas de modo a não limitar futuras implementações dos robôs, assim sendo, cada robô pode enviar suas ações específicas com os seus parâmetros de controle. As estratégias de sincronização dos robôs foram divididas em duas partes: macrossincronização, que gerenciou o sincronismo dos relógios dos robôs pelo algoritmo SNTP (*Simple Network Time Protocol*) e a compensação dos atrasos mecânicos dos robôs utilizando redes neurais artificiais; e microssincronização que implementou um módulo perceptual autônomo para permitir que o robô "ouça" a performance e se sincronize ritmicamente, em tempo real, por meio de pulsos musicais.

Um instrumento musical robótico, chamado de BongoBot, utilizando um bongô foi construído para avaliação da plataforma RoboMus, onde o dispositivo de controle usado foi o Raspberry pi. No teste de macrossincronia para sincronização dos relógios utilizou-se o SNTP e dois Raspberry pi, o teste apresentou resultado médio de atraso igual a $7,82 \pm 5,55\ ms$, mostrando-se aceitável ante a percepção sonora humana (20ms). Entretanto, vale ressaltar que a rede era composta apenas pelo SSMM e os dois dispositivos Raspberry, assim, estudos sobre o efeito do aumento do número de robôs são necessários, pois o número de conexões na rede pode impactar diretamente na precisão do SNTP.

O bongô é um instrumento contendo dois tambores e, para os testes de macrossincronização visando a compensação do atraso mecânico, o robô foi divido em dois, onde cada tambor foi controlado por um Raspberry diferente, formando então dois robôs distintos. Esses robôs possuem atrasos provenientes dos atuadores mecânicos, assim, o SSMM aprendeu de forma automática, e por meio de redes neurais, o atraso dos dois robôs. Para o robô 1 (*hembra*) a rede apresentou RMSE (*Root Mean Square Error*) de 0,82% e o robô 2 alcançou um RMSE de 1,2%. Sem a compensação do atraso mecânico alguns testes apresentaram atrasos acima de 20 ms, após a aplicação da compensação do atraso nenhum dos testes apresentaram atrasos acima da percepção humana. Além

disso, a compensação reduziu o atraso médio na ordem de 50%, tendo uma redução de 55,33% do atraso médio para o melhor caso e de 39,02% para o pior caso. Esses resultados foram satisfatórios, uma vez que não apresentaram atrasos maiores do que a percepção sonora humana. Entretanto, cabe um estudo aprofundado com robôs mais complexos mecanicamente, elevando a necessidade de abstração da rede.

As estratégias de microssincronização fazem uso de algoritmos para detectar o BPM (Batidas Por Minuto), a sequência de pulsos e a assinatura de tempo aplicados ao áudio captado pelo módulo perceptual do robô. Nos testes foram usados um metrônomo digital e o BongoBot em sua forma completa, onde o robô capturou o som do metrônomo e aplicou os algoritmos para sincronização rítmica. O algoritmo de detecção do BPM apresentou uma imprecisão com RMSE (*Root Mean Square Error*) de 6,98%. O robô captou ("ouviu") 8 segundos da performance para então executar os algoritmos, o tempo de processamento desses algoritmos ficou em torno de 7 segundos, assim, nesse tempo atrasos foram acumulados devido a imprecisão do BPM, pois o robô entra em cena com base em informações de 7 segundos atrás. No total foram 10 testes com 1 minuto de duração cada, onde o áudio do robô e o do metrônomo foram gravados em canais de áudio separados.

As análises desses áudios de microssincronização mostraram que dos 10 testes, 4 apresentaram atraso médio acima de 20ms. Já fazendo a mesma análise para apenas os 10 primeiros segundos dos áudios, visando evitar o atraso acumulativo devido a imprecisão do BPM, apenas 2 testes apresentaram atraso médio acima de 20ms. Esses algoritmos apresentaram imprecisão e alto tempo de processamento, nos testes de microssincronização foi utilizado um notebook com *core i7* 8ª geração no lugar do Raspberry para controle do robô. Assim, os resultados foram razoáveis em relação a precisão do algoritmo, uma vez que pode-se contornar o atraso acumulativo ressincronizando de tempos em tempos, porém essa ressincronização sobrecarrega o sistema. Já em relação ao tempo de processamento, essa estratégia não apresentou resultados satisfatórios, uma vez que uma das premissa da plataforma é baixo custo e, geralmente, os dispositivos de baixo custo têm baixo poder computacional. No entanto, com os avanços tecnológicos a barreira de processamento desses dispositivos pode ser rompida.

Para verificação do módulo perceptual autônomo do BongoBot também foi executada uma avaliação perceptual feita por humanos por meio de um questionário *online*, visando obter respostas sobre o sincronismo entre o robô e o metrônomo. Para cada participante foram apresentados 8 áudios (contendo o som do metrônomo e do bongô) selecionados aleatoriamente de um banco de 20 áudios de 10 segundos cada. A avaliação geral mostrou que 55% dos testes tiveram aceitação acima de 70%, ou seja, votos sim para a pergunta: *você considera que os pulsos sonoros nesse áudio estão em sincronia?*. No questionário foi perguntado sobre a experiência musical, sendo de 1 (pouca) a 5 (bastante).

Para o grupo com experiência 1 ou 2, 55% dos testes apresentaram aceitação acima de 70% e para o grupo com experiência 4 ou 5, 60% dos testes apresentaram aceitação acima de 70%. Na avaliação geral e por grupos, notou-se também que a aceitação diminui de acordo com o aumento dos atrasos, como era esperado. De acordo com o teste perceptual, a microssincronização mostrou um resultado satisfatório para os primeiros robôs da plataforma RoboMus.

Como trabalhos futuros para macrossincronização, sugere-se o estudo do algoritmo de sincronização de relógios com mais robôs conectados na plataforma, bem como, a aplicação de algoritmos mais precisos na sincronização dos relógios. Além disso, estudos para determinar o melhor momento de ressincronização do relógio de um robô. Também inclui-se a construção de robôs mecanicamente mais complexos para testes mais robustos do aprendizado do atraso mecânico. Para microssincronização sugere-se a aplicação de processamento paralelo para redução do tempo de processamento e estudos sobre algoritmos mais precisos e menos complexos computacionalmente. Sugere-se ainda, o desenvolvimento de algoritmos para percepção de gênero musical, possibilitando a escolha de ritmos condizentes com o gênero.

Referências

ATZORI, L.; IERA, A.; MORABITO, G. The Internet of Things: A survey. *Computer Networks*, v. 54, n. 15, p. 2787–2805, 2010. ISSN 1389-1286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128610001568>. Citado na página 31.

BAGINSKY, N. A. *Aglaopheme*. 199–? Disponível em: <http://baginsky.de/agl/agl{_}index.h>. Citado na página 38.

BAGINSKY, N. A. *The Three Sirens*. 199–? Disponível em: <http://www.the-three-sirens.info/binfo.html>. Citado 2 vezes nas páginas 37 e 38.

BENWARD, B.; SAKER, M. *Music in Theory and Practice Volume 1*. Eighth edition. New York, NY: McGraw-Hill Education, 2008. Citado 2 vezes nas páginas 47 e 73.

BÖCK, S.; KREBS, F.; SCHEDL, M. Evaluating the Online Capabilities of Onset Detection Methods. In: *13th International Society for Musica Information Retrieval Conference (ISMIR 2012)*. Porto, Portugal: [s.n.], 2012. p. 49–54. Disponível em: <http://ismir2012.ismir.net/event/papers/049{_}ISMIR{_}20>. Citado na página 44.

BOLEY, H.; CHANG, E. Digital ecosystems: Principles and semantics. In: CITESEER. *Digital EcoSystems and Technologies Conference*. [S.l.], 2007. p. 398–403. Citado na página 42.

BRETAN, M.; WEINBERG, G. Chronicles of a Robotic Musical Companion. In: *Proceedings of the International Conference on New Interfaces for Musical Expression*. London, United Kingdom: Goldsmiths, University of London, 2014. p. 315–318. Disponível em: <http://www.nime.org/proceedings/2014/nime2014{_}303.> Citado na página 32.

BRETAN, M.; WEINBERG, G. A survey of robotic musicianship. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 59, n. 5, p. 100–109, abr. 2016. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/2818994>. Citado 2 vezes nas páginas 31 e 37.

BöCK, S.; WIDMER, G. Maximum filter vibrato suppression for onset detection. In: *16th International Conference on Digital Audio Effects*. Maynooth, Ireland: [s.n.], 2013. Citado na página 48.

CAMPOREZ, H. A. F. et al. Interface Computacional para Controle Musical Utilizando os Movimentos dos Olhos. *Revista Vórtex*, Curitiba, v. 6, n. 2, p. 1–17, 2018. Disponível em: <http://vortex.unespar.edu.br/camporez_et_al_v6_n2.pdf>. Citado 2 vezes nas páginas 63 e 64.

CAMPOREZ, H. A. F. et al. RoboMus: Uma Plataforma para Performances Musicais Robóticas. In: KELLER, D.; LIMA, M. H. D. (Ed.). *Aplicações em Música Ubíqua*. 1. ed. São Paulo, Brasil: ANPPOM, 2018. p. 58–93. Citado na página 61.

CAMPOREZ, H. A. F. et al. Protótipo robótico de baixo custo para performance musical em violão acústico. In: SCHIAVONI, F. L. et al. (Ed.). *Proceedings of the 15th Brazilian Symposium on Computer Music*. Campinas - SP - Brazil: Sociedade Brasileira de Computação, 2015. p. 137–147. Citado na página 33.

CASSAPIAN, M. R.; PELLENZ, C. C. D. O. Doenças ocupacionais e sua prevenção em estudantes de música – realidade de uma instituição de ensino superior de curitiba. *Revista Música Hodie*, v. 10, n. 2, p. 91–107, 2010. ISSN 16763939. Disponível em: <https://www.revistas.ufg.br/musica/article/view/19457>. Citado na página 32.

CHRIST, O. et al. Towards the Internet of Things. n. July, 2007. Citado na página 44.

CLARKE, E. F. Generativity, mimesis and the human body in music performance. *Contemporary Music Review*, v. 9, n. 1-2, p. 207–219, 1993. ISSN 14772256. Citado na página 62.

COSTALONGA, L.; MILETO, E. M.; MIRANDA, E. Octopus Music API: Modelling Musical Performance. In: *Proceedings of the XI Brazilian Symposium On Computer Music (SBCM 2007)*. São Paulo, Brasil: USP, 2007. p. 219–222. Citado 2 vezes nas páginas 63 e 64.

COSTALONGA, L.; MILETTO, E. M.; PIMENTA, M. Musicality Centred Interaction Design to Ubimus: a First Discussion. In: *Proceedings of 14th International Symposium on Computer Music Multidisciplinary Research*. Marseille, France: [s.n.], 2019. p. 640–651. Citado na página 31.

COSTALONGA, L. L.; PIMENTA, M. S.; MIRANDA, E. R. Understanding biomechanical constraints for modelling expressive performance: A guitar case study. *Journal of New Music Research*, Routledge, v. 48, n. 4, p. 331–351, 2019. Disponível em: <https://doi.org/10.1080/09298215.2019.1643892>. Citado na página 32.

COYLE, E.; GAINZA, M. Time signature detection by using a multi-resolution audio similarity matrix. In: *Audio Engineering Society Convention 122*. [s.n.], 2007. Disponível em: <http://www.aes.org/e-lib/browse.cfm?elib=14139>. Citado 3 vezes nas páginas 51, 52 e 53.

DELIÈGE, I.; WIGGINS, G. A. *Musical creativity: Multidisciplinary research in theory and practice*. [S.l.]: Psychology Press, 2006. Citado na página 31.

DOGANTAN-DACK, M. The body behind music: Precedents and prospects. *Psychology of Music*, v. 34, n. 4, p. 449–464, 2006. ISSN 17413087. Citado na página 63.

ELLIS, D. P. W. Beat tracking by dynamic programming. *Journal of New Music Research*, Routledge, v. 36, n. 1, p. 51–60, 2007. Disponível em: <https://doi.org/10.1080/09298210701653344>. Citado 4 vezes nas páginas 47, 48, 49 e 50.

ERICSSON, K. A.; KRAMPE, R. T.; TESCH-RÖMER, C. The role of deliberate practice in the acquisition of expert performance. *Psychological Review*, v. 100, n. 3, p. 363–406, 1993. ISSN 1939-1471. Disponível em: <http://doi.apa.org/getdoi.cfm?doi=10.1037/0033-295X.100.3.363>. Citado na página 32.

FOOTE, J. Visualizing music and audio using self-similarity. In: *Proceedings of the Seventh ACM International Conference on Multimedia (Part 1)*. New York, NY, USA: Association for Computing Machinery, 1999. (MULTIMEDIA '99), p. 77–80. ISBN 1581131518. Disponível em: <https://doi.org/10.1145/319463.319472>. Citado na página 51.

GANZA, V. H. de. *Estudos de Psicopedagogia Musical*. 3ª. ed. São Paulo, Brasil: Summus Editorial, 1988. Citado na página 32.

GOEBL, W.; PALMER, C. Synchronization of timing and motion among performing musicians. *Music Perception: An Interdisciplinary Journal*, University of California Press Journals, v. 26, n. 5, p. 427–438, 2009. ISSN 0730-7829. Disponível em: <https://mp.ucpress.edu/content/26/5/427>. Citado na página 40.

HAYKIN, S. *Redes neurais artificiais: princípios e prática*. Porto Alegre: Bookman, 2001. ISBN 9788573077186. Citado 4 vezes nas páginas 56, 57, 58 e 86.

HOFFMAN, G.; WEINBERG, G. Shimon: An interactive improvisational robotic marimba player. In: *CHI '10 Extended Abstracts on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2010. (CHI EA '10), p. 3097–3102. ISBN 9781605589305. Disponível em: <https://doi.org/10.1145/1753846.1753925>. Citado 3 vezes nas páginas 32, 39 e 41.

HUANG, X. et al. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. 1st. ed. USA: Prentice Hall PTR, 2001. ISBN 0130226165. Citado na página 46.

IBGE. *Censo Demográfico 2010 - Características Gerais da População, Religião, e Pessoas com Deficiência*. 2010. Disponível em: <http://biblioteca.ibge.gov.br/visualizacao/periodicos/94/cd{_}2010{_}religiao{_}defici>. Citado na página 32.

IEEE Standard for Floating-Point Arithmetic. *IEEE Std 754-2008*, p. 1–70, Aug 2008. ISSN null. Citado na página 53.

JORDà, S. Afasia: the ultimate homeric one-man-multimedia-band. In: *Proceedings of the International Conference on New Interfaces for Musical Expression*. Dublin, Ireland: [s.n.], 2002. p. 102–107. Disponível em: <http://www.nime.org/proceedings/2002/nime2002_102.pdf>. Citado 2 vezes nas páginas 38 e 39.

KAPUR, A. A History of robotic Musical Instruments. In: *ICMC*. [S.l.: s.n.], 2005. Citado 2 vezes nas páginas 31 e 37.

KEHTARNAVAZ, N. Chapter 7 - frequency domain processing. In: KEHTARNAVAZ, N. (Ed.). *Digital Signal Processing System Design (Second Edition)*. Second edition. Burlington: Academic Press, 2008. p. 175 – 196. ISBN 978-0-12-374490-6. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9780123744906000076>. Citado 2 vezes nas páginas 45 e 46.

KELLER, D. et al. Convergent trends toward ubiquitous music. *Journal of New Music Research*, Routledge, v. 40, n. 3, p. 265–276, 2011. Disponível em: <https://doi.org/10.1080/09298215.2011.594514>. Citado na página 41.

KELLER, D.; LAZZARINI, V. Ecologically Grounded Creative Practices in Ubiquitous Music. *Organised Sound*, Cambridge University Press, v. 22, n. 1, p. 61–72, 2017. Citado na página 31.

KELLER, D.; LAZZARINI, V.; PIMENTA, M. S. (Ed.). *Ubiquitous Music*. 1. ed. [S.l.]: Springer International Publishing, 2014. ISBN 978-3-319-11152-0. Citado 3 vezes nas páginas 31, 41 e 63.

MAROUANI, H.; DAGENAIS, M. R. Internal clock drift estimation in computer clusters. *J. Comp. Sys., Netw., and Comm.*, Hindawi Limited, London, GBR, v. 2008, jan. 2008. ISSN 1687-7381. Disponível em: <https://doi.org/10.1155/2008/583162>. Citado na página 72.

MARTIN, J. et al. *Network Time Protocol Version 4: Protocol and Algorithms Specification*. RFC Editor, 2010. RFC 5905. (Request for Comments, 5905). Disponível em: <https://rfc-editor.org/rfc/rfc5905.txt>. Citado na página 54.

MED, B. *Teoria da Música*. 4ª edição. ed. Brasília, DF: Musimed, 1996. Citado na página 51.

MILLS, D. L. *Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI*. RFC Editor, 2006. RFC 4330. (Request for Comments, 4330). Disponível em: <https://rfc-editor.org/rfc/rfc4330.txt>. Citado na página 54.

MOOG, R. A. MIDI: Musical Instrument Digital Interface. *J. Audio Eng. Soc*, v. 34, n. 5, p. 394–404, 1986. Disponível em: <http://www.aes.org/e-lib/browse.cfm?elib=5267>. Citado 2 vezes nas páginas 38 e 54.

MOORE, B. C. J.; PETERS, R. W.; GLASBERG, B. R. Detection of temporal gaps in sinusoids: Effects of frequency and level. *The Journal of the Acoustical Society of America*, v. 93, n. 3, p. 1563–1570, 1993. ISSN 0001-4966. Disponível em: <http://asa.scitation.org/doi/10.1121/1.406815>. Citado na página 62.

NETTO, A. R. et al. Árvore das Memórias: Instalação Multimídia Interativa. In: *Proceedings of the 15th Brazilian Symposium on Computer Music*. Campinas: Unicamp, 2015. p. 76–83. Citado 2 vezes nas páginas 33 e 43.

PALMER, C. MUSIC PERFORMANCE. *Annual Review of Psychology*, v. 48, n. 1, p. 115–138, 1997. ISSN 0066-4308. Disponível em: <http://www.annualreviews.org/doi/10.1146/annurev.psych.48.1.115>. Citado na página 62.

PEREZ, F. *The Complete Acoustic Lap Steel Guitar Method*. Fenton: Mel Bay Publications, 2016. ISBN 0-7866-9591-9. Citado na página 113.

PIMENTA, M. et al. Música Ubíqua: Suporte para atividades musicais em dispositivos móveis. *ScientiaTec*, v. 2, p. 61–74, 2015. Citado 2 vezes nas páginas 31 e 41.

PISONI, D. B. Identification and discrimination of the relative onset time of two component tones: implications for voicing perception in stops. *The Journal of the Acoustical Society of America*, v. 61, n. May 2013, p. 1352–1361, 1977. ISSN 00014966. Citado na página 62.

Rabiner, L. R.; Schafer, R. W. *An Introduction to Digital Speech Processing*. now, 2007. 33-37 p. Disponível em: <https://ieeexplore.ieee.org/document/8187052>. Citado na página 45.

ROSENBLATT, F. *The perceptron: A probabilistic model for information storage and organization in the brain*. US: American Psychological Association, 1958. 386–408 p. Citado na página 58.

SINGER, E.; FEDDERSEN, J. Lemur: Robotic musical instruments. In: *Proceedings of the 12th Annual ACM International Conference on Multimedia*. New York, NY, USA: Association for Computing Machinery, 2004. (MULTIMEDIA '04), p. 184–185. ISBN 1581138938. Disponível em: <https://doi.org/10.1145/1027527.1027569>. Citado 2 vezes nas páginas 38 e 39.

SINGER, E.; LARKE, K.; BIANCIARDI, D. Lemur guitarbot: Midi robotic string instrument. In: *Proceedings of the International Conference on New Interfaces for Musical Expression*. Montreal, Canada: [s.n.], 2003. p. 188–191. Disponível em: <http://www.nime.org/proceedings/2003/nime2003_188.pdf>. Citado 2 vezes nas páginas 38 e 39.

STEVENS, S. S.; VOLKMANN, J. The relation of pitch to frequency: A revised scale. *The American Journal of Psychology*, University of Illinois Press, v. 53, n. 3, p. 329–353, 1940. ISSN 00029556. Disponível em: <http://www.jstor.org/stable/1417526>. Citado na página 45.

TURCHET, L. Smart Mandolin: autobiographical design, implementation, use cases, and lessons learned. In: *Proceedings of Audio Mostly Conference*. [s.n.], 2018. p. 13:1—-13:7. Disponível em: <http://doi.acm.org/10.1145/3243274.3243280>. Citado na página 44.

TURCHET, L. et al. Internet of Musical Things: Vision and Challenges. In: *IEEE Access*. [S.l.: s.n.], 2018. v. 6, p. 61994–62017. ISSN 2169-3536. Citado 3 vezes nas páginas 31, 42 e 43.

TURCHET, L.; MCPHERSON, A.; BARTHET, M. Real-time hit classification in a Smart Cajón. *Frontiers in ICT*, Frontiers, v. 5, p. 16, 2018. Citado na página 44.

TURCHET, L.; MCPHERSON, A.; FISCHIONE, C. Smart instruments: Towards an ecosystem of interoperable devices connecting performers and audiences. In: *Proceedings of the Sound and Music Computing Conference*. Hamburg: [s.n.], 2016. p. 498–505. Citado na página 44.

WEINBERG, G.; DRISCOLL, S. Toward robotic musicianship. *Computer Music Journal*, v. 30, n. 4, p. 28–45, 2006. Disponível em: <https://doi.org/10.1162/comj.2006.30.4.28>. Citado na página 40.

WEISER, M. The Computer for the 21 st Century. *Scientific American*, Scientific American, a division of Nature America, Inc., v. 265, n. 3, p. 94–105, 1991. ISSN 00368733, 19467087. Disponível em: <http://www.jstor.org/stable/24938718>. Citado 2 vezes nas páginas 31 e 41.

WRIGHT, M.; FREED, A. Open sound control: A new protocol for communicating with sound synthesizers. In: INTERNATIONAL COMPUTER MUSIC ASSOCIATION (ICMA). *International Computer Music Conference (ICMC)*. Thessaloniki, Hellas:

International Computer Music Association (ICMA), 1997. p. 101–104. Disponível em: <http://cnmat.berkeley.edu/http://hdl.handle.net/2027/spo.bbp2372.1997.033/open_sound_control_new_protocol_communicating_sound_synthesizers>. Citado na página 53.

ZANETTI, H. A. P.; BONACIN, R. Uma metodologia baseada em semiótica para elaboração e análise de práticas de ensino de programação com robótica pedagógica. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, v. 25, n. 1, p. 1233, 2014. ISSN 2316-6533. Citado na página 32.

Apêndices

APÊNDICE A – Protótipo Lap Steel Guitar RoboMus

O *lap steel guitar* robótico é um protótipo RoboMus, ainda em desenvolvimento, que envolve a robotização de um *lap steel guitar* que também é conhecido como guitarra havaiana (Figura 43). Esse instrumento normalmente é tocado deslizando um dispositivo de metal ou vidro (*slide*) sobre as cordas, gerando um som característico com *glissando* [1] (PEREZ, 2016).

Figura 43 – Foto de uma *Lap Steel Guitar*

Fonte: (PEREZ, 2016).

O *lap steel guitar* robótico é um protótipo com nove cordas em afinações diferentes do padrão, seguindo as afinações: G4, F4, E4, C4, A3, F3, D3, A2 e D2. Essa afinação foi pensada para propiciar a formação de acordes de primeiro grau maior e menor com as cordas soltas ou com a barra de *slide* posicionada em alguma casa. Dessa forma, simplifica-se a demanda mecânica do robô.

A estrutura física do robô foi projetada para suportar as seguintes ações: excitação/abafamento das cordas de forma independente, movimentação da barra de *slide* com precisão e pressionamento das cordas.

[1] Transição suave entre alturas tonais (frequências)

Para excitação das cordas foram usados 9 servomotores, sendo fixado uma palheta em cada motor para tocar uma respectiva corda. Já para abafar as cordas foram usados 9 servomotores com espuma porosa nas pontas. Esses servomotores foram posicionados acima das cordas e próximos ao ponto de captação como é mostrado nas Figuras 44, 45 (a) e (b). Já para o dispositivo deslisante utilizou-se um motor de passo com correia dentada para movimentar a barra de *slide* sobre as cordas, e o controle da barra foi feito por dois servomotores como mostra a Figura 44 e 45 (c). Além disso, todo o controle foi feito com a estrutura definida na Figura 19 (a).

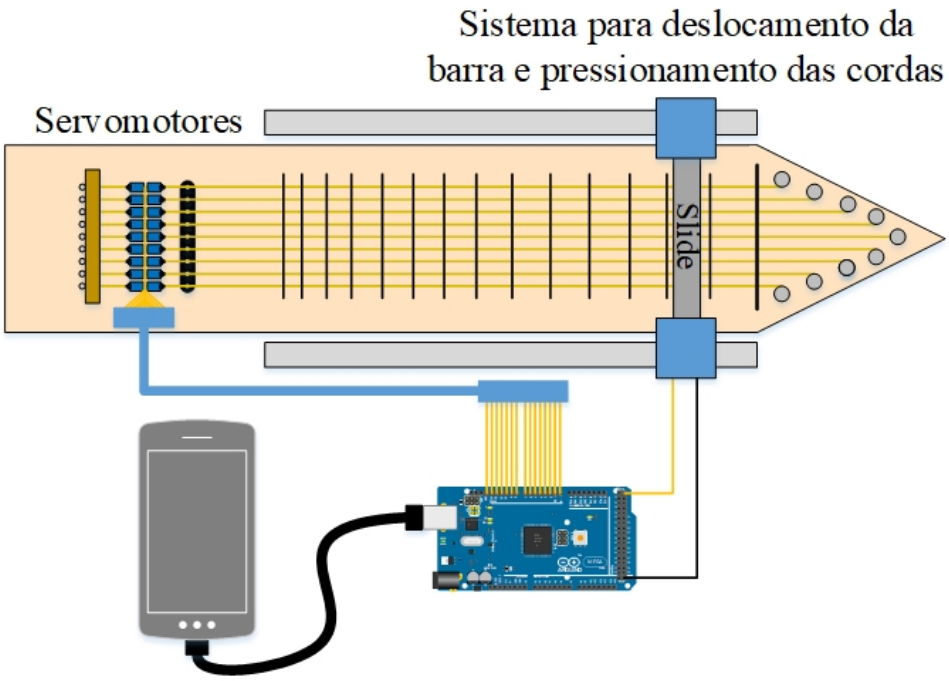

Figura 44 – Estrutura do *lap steel guitar* robótico

Fonte: Autor.

A.1 Formato das Mensagens

Para o *lap steel guitar* robótico define-se ações complementares às ações pré definidas para instrumentos trasteados, onde a informação de tempo de execução das mensagens será abstraída dos quadros apresentados a seguir, pois o *bundle* contém um campo para essa finalidade. Portanto, as ações propostas são: *slide*; mover barra na vertical e tocar cordas.

A ação de *slide* representa deslizar a barra pressionada levemente contra as cordas. O seu formato é definido no Quadro 15 e seus parâmetros são:

- identificador: identificador da mensagem;

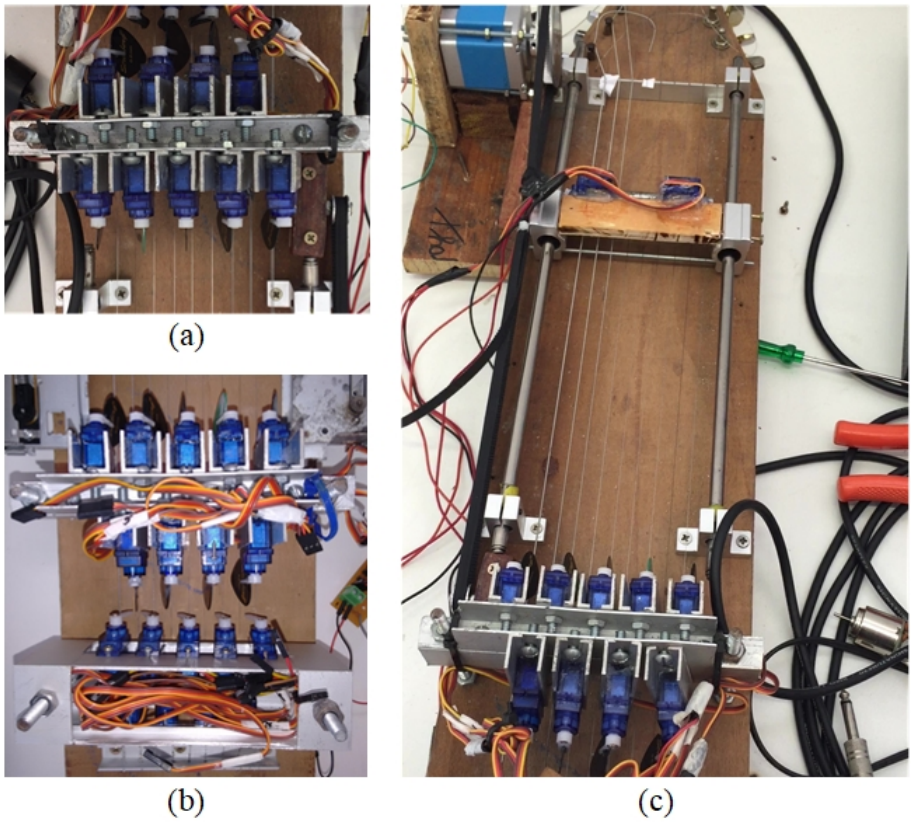

Figura 45 – Fotos do *lap steel guitar* robótico: (a) dispositivo de excitação das cordas, (b) dispositivo de excitação e abafamento das cordas e (c) foto do robô completo

Fonte: Autor.

- posição inicial: posição definida pelo número do traste onde a barra deve começar a executar o movimento;
- posição final: posição definida pelo número do traste onde a barra deve finalizar a execução do movimento.

Quadro 15 – Formato da mensagem de slide

Componente	**Tipo**
/Endereço OSC do robô/Slide	Endereço OSC
Identificador	Inteiro
Posição (traste) inicial	Inteiro
Posição (traste) final	Inteiro

A ação de tocar cordas consiste em posicionar a barra em algum traste específico e tocar cordas selecionadas juntas. Para isso, o formato da ação é definido no Quadro 16, onde os parâmetros são:

- identificador: identificador da mensagem;
- traste: o número da casa em que a barra será pressionada ou zero para cordas soltas;
- cordas: um vetor de nove posições representando as cordas, onde o número da posição representa uma corda e o conteúdo dessa posição determina o estado, sendo 1 para tocar e 0 para não tocar. Por exemplo, o vetor 001010000 indica que apenas as cordas 3 e 5 serão tocadas.

Quadro 16 – Formato da mensagem para tocar cordas

Componente	**Tipo**
/Endereço OSC do robô/TocarCordas	Endereço OSC
Identificador	Inteiro
Traste	Inteiro
Cordas	*String*

A ação de silenciar cordas consiste em abafar um certo conjunto de cordas. O formato da ação é definido no Quadro 17, onde os parâmetro são:

- identificador: identificador da mensagem;
- cordas: um vetor de nove posições representando as cordas, onde o número da posição representa uma corda e o conteúdo do vetor na mesma posição representa o seu estado, 1 para abafar e 0 para não abafar.

Quadro 17 – Formato da mensagem para silenciar cordas

Componente	**Tipo**
/Endereço OSC do robo/SilenciarCordas	Endereço OSC
Identificador	Inteiro
Cordas	*String*