Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Dispositivo de conversão de sinais sonoros para sinais táteis voltados a usuários surdos

Tito Marcos De Moraes Klautau

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Marcelo Grandi Mandelli

Brasília
2019

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Coordenador: Prof. Dr. Edison Ishikawa

Banca examinadora composta por:

Prof. Dr. Marcelo Grandi Mandelli (Orientador) — CIC/UnB
Prof. Dr. Marcus Vinicius Lamar — CIC/UnB
Prof.ª Dr.ª Carla Cavalcante Koike — CIC/UnB

CIP — Catalogação Internacional na Publicação

De Moraes Klautau, Tito Marcos.

Dispositivo de conversão de sinais sonoros para sinais táteis voltados a usuários surdos / Tito Marcos De Moraes Klautau. Brasília : UnB, 2019.
63 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2019.

1. inclusão de deficientes auditivos, 2. feedback háptico, 3. surdos

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Dispositivo de conversão de sinais sonoros para sinais táteis voltados a usuários surdos

Tito Marcos De Moraes Klautau

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Marcelo Grandi Mandelli (Orientador)
CIC/UnB

Prof. Dr. Marcus Vinicius Lamar
CIC/UnB

Prof.ª Dr.ª Carla Cavalcante Koike
CIC/UnB

Prof. Dr. Edison Ishikawa
Coordenador do Bacharelado em Ciência da Computação

Brasília, 11 de dezembro de 2019

Dedicatória

Dedico este trabalho à:

Meu falecido pai, Sebastião Dias Klautau;

Minha mãe, Maria Corina David de Moraes Klautau;

Minha namorada, Tereza Letícia Bezerra Niederauer;

estes que sempre dedicaram-se a ajudar-me na vida acadêmica e emocional;

Agradecimentos

Neste trabalho ofereço minhas sinceras gratitudes à:

Meus pais, vó Ana, vó Lelé(Nazaré) e tia Celeste, que me ajudaram a fundar uma base de conhecimentos importantes para minha vida na minha educação básica.

Meu pai por me incentivar do seus modo a perseguir uma educação superior e ser um grande apoio para conversas enquanto vivo.

Minha namorada Tereza por me aguentar em um tempo que não pode ser considerado um dos meus melhores.

Minha vó Delvair e minha mãe pelo suporte emocional que me deram nos momentos de stress.

Aos professores Mandelli e Lamar por persistirem em apoiar-me neste projeto.

Resumo

Este trabalho apresenta um protótipo de um dispositivo modular de baixo custo dedicado à conversão de ondas sonoras em sinais hápticos através de motores de vibração. A tradução dos sinais sonoros para táteis pode ser realizada em tempo real através de um aplicativo de celular conectado a um módulo de acionamento dos motores por bluetooth. Testes foram efetuados com um usuário deficiente auditivo para verificar a viabilidade do sistema. Com resultados positivos, o projeto mostra que uma implementação de um reprodutor de músicas portátil para surdos é viável e passível de produção comercial a baixo custo.

Palavras-chave: inclusão de deficientes auditivos, feedback háptico, surdos

Abstract

This thesis presents an prototipe of a low cost modular device used for translation of soundwaves into a haptic medium using vibration motors. The translation of soundwaves to tactile feedback can be calculated on live data by an app instaled on a mobile device, the app records in real time, applies the conversion algorithm on the soundwave data, and then sends the activation array of that snapshot by bluetooth to the controler module. The document also describes tests that were executed with a hearing-impaired user to determine the project viability. With positive results, the project shows that a frugal implementation of a device similar to a portable music players for the hearing-impaired is feasible and could be implemented to mass market production with more time of development.

Keywords: hearing imparied inclusion, haptic feedback, wearable device, deafs

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

2PD Two-point discrimination. 15, 42, 47

ASA Acoustical Society of America. 4

FFT Fast Fourier Trasform. 21, 24, 31

GPIO General-purpose input/output. 29, 34, 48

HFM Haptic Feedback Music. 21–23, 26–29, 47

JVM Java Virtual Machine. 15

PCM Pulse Code Modulation. 6, 7

PWM Pulse-Width Modulation. 42, 47

RIFF Resource Interchange File Format. 5, 6, 24

TPDT Two-Point discrimination threshold. 15

UnB Universidade de Brasília. 38

WAV Waveform Audio File Format. 6, 9, 21, 22, 24

WAVE Waveform Audio File Format. 3, 5, 6

Capítulo 1

Introdução

Considerada como a primeira arte, a música mostra a capacidade de ondas mecânicas transmitirem informações que devido a um ritmo acabam por invocar diversos sentimentos a quem por ventura os receba. Um dos momentos mais importantes do começo da afirmação de sua personalidade própria nos anos de formação de muitas pessoas vem por base da música. A definição de um estilo preferido, a busca de comunidades que tenham o mesmo gosto, o consumismo de materiais pertinentes a este. Enfim a influencia da música na cultura humana é inegável e todos aqueles expostos a ela acabam por serem parcialmente moldados pelos seus ritmos.

A deficiência auditiva nos tempos atuais não apresenta perda de função significativa a ponto de uma pessoa tornar-se isolado devido a esta. Porém mas um dos maiores detrimentos da falta de audição é a alienação perante essa grande parte da cultura musical e sonora das sociedades em que habitam. Mesmo com dificuldades, muitos surdos conseguem ter um bom entendimento sobre as artes musicais devido às propriedades mecânicas da propagação do som. Infelizmente mesmo para estes o limite do tato humano não permite um discernimento aceitável de altas frequências presentes na maioria das peças musicais.

1.1 Objetivo

O projeto tem como objetivo a prototipagem de um sistema capaz de traduzir as informações presentes numa peça sonora para um meio passível de interpretação somente pelo tato humano, a ser utilizado por surdos para percepção de sons ritmados em seu redor no espectro de frequências completo da audição humana. Especificadamente, este projeto tem como objetivo converter, em tempo real, uma peça sonora em sinais táteis através de motores de vibração. Para isto, um software será implementado para realizar essa con-

versão, o qual será interconectado com um módulo de hardware que acionará os motores de vibração de acordo com a peça sonora.

O projeto deve aderir a 3 pontos chaves para o protótipo ser útil ao usuário final, estes são:

- Acessibilidade perante o custo:

  O projeto deve possuir um baixo custo de reprodução e ser de fácil uso, o hardware deve ser simples e construído de materiais facilmente encontrados.

- Modularidade:

  Tanto o hardware quando o software devem ser implementados de forma modular, possibilitando a configuração do número de motores associados a este. Pode-se ter um número maior de motores para se ter uma fidelização maior na tradução do sinal sonoro de entrada com o sinal táctil de saída, ou então, um número menor destes pela falta de espaço para o posicionamento dos mesmos.

- Mobilidade

  O sistema deve ser um dispositivo móvel. O usuário deve conseguir utilizá-lo de forma confortável em suas atividades diárias.

1.2 Organização dos Capítulos

Este trabalho divide-se em 5 capítulos:

1. Introdução

   Este capitulo, onde se apresenta a ideia básica do projeto.

2. Base Teórica

   Neste capitulo apresenta-se a base teórica utilizada no projeto.

3. Implementação

   Este capitulo mostra como foi o desenvolvimento da implementação deste trabalho.

4. Testes

   Compreende os testes realizados com o protótipo e seus resultados.

5. Conclusão

   Apresenta as conclusões finais e os trabalhos futuros a serem desenvolvidos.

Capítulo 2

Base Teórica

Neste capitulo são apresentados os conceitos básicos utilizados neste projeto divididos nas seguintes seções:

1. Ondas Sonoras

   Breve apresentação de conceitos importantes sobre ondas sonoras que serão utilizados posteriormente.

2. Espectrograma

   Simples definição de um espectrograma com um exemplo utilizado no projeto.

3. RIFF WAVE sistema de arquivo

   Descrição do formato WAVE e de como este é organizado

4. Sistema Auditivo Humano

   Explicação sobre como o sistema auditivo humano compreende estímulos sonoros de diversas frequências.

5. Transformada Discreta de Fourier

   Definição da Transformada Discreta de Fourier, utilizada para o calculo de espectrogramas no projeto.

6. Hardware Utilizado no projeto

   Listagem e descrição das partes de hardware utilizadas no projeto.

7. Software Utilizado no projeto

   Listagem e descrição das linguagens e bibliotecas utilizadas no projeto.

8. Fenômenos de iteração táctil no corpo

   Listagem e descrição dos fenômenos táteis encontrados nos testes efetuados que acabaram por gerar complicações no posicionamento dos motores.

9. Pesquisas Correlacionadas

   Pesquisas e projetos que tem um tema em comum com o projeto que acabaram por servir de inspiração para soluções deste.

2.1 Ondas sonoras

2.1.1 Definição

Pela definição do dicionário Merrian Webster[20] onda sonora engloba toda onda de pressão longitudinal propagada em qualquer material independente de se constituir ou não som audível.

2.1.2 Propriedades Importantes

São propriedades de mensuração e diferenciação das ondas sonoras:

- Intensidade

  Também conhecida como amplitude, a intensidade da onda sonora representa a pressão gerada pela onda no meio em que se propaga.

- Frequência

  Calculada pelo inverso de seu comprimento de onda, a frequência de uma onda sonora permite a distinção de tons em acordes musicais, gerando assim uma tonalidade característica a onda.

- Timbre

  Definido pela *Acoustical Society of America* (ASA) [27] como "a propriedade passível de sensação auditiva que permite o ouvinte distinguir diferentes ondas sonoras parecidas que possuam a mesma intensidade e frequência mas geram sons divergentes", em si o timbre representa a diferença entre ondas com a mesma frequência, mas que chegaram a esta frequência devido a interferências de ondas diferentes, gerando assim envelopes(envoltória) divergentes.

2.2 Espectrograma

Um sinal sonoro complexo, tal como uma música ou o barulho de uma rua movimentada, pode ser facilmente representado por uma onda. Esta pode ser gerada pela interferência de ondas de menor complexidade, que estejam ocupando o mesmo meio e interagindo entre si pela propriedade de interferência, a qual enuncia que a amplitude no encontro de duas ou mais ondas do mesmo tipo em um determinado ponto resulta na soma vetorial de suas amplitudes. Assim partindo deste princípio qualquer onda sonora pode ser decomposta em uma soma de senoides de frequências diversas, pois em cada momento a soma das amplitudes destas geram a amplitude da onda original. O espectrograma em si é a representação da intensidade de ondas senoides puras que fazem parte da onda sonora original quando decompostas. Um exemplo de espectrograma pode ser visto na Figura 2.1 que mostra as amplitudes das ondas senoides que somadas geram a onda sonora representada na Figura 2.3 .O método mais comum usado para o calculo de espectrogramas é a transformada discreta de Fourier, abordada na seção 2.5 deste trabalho.

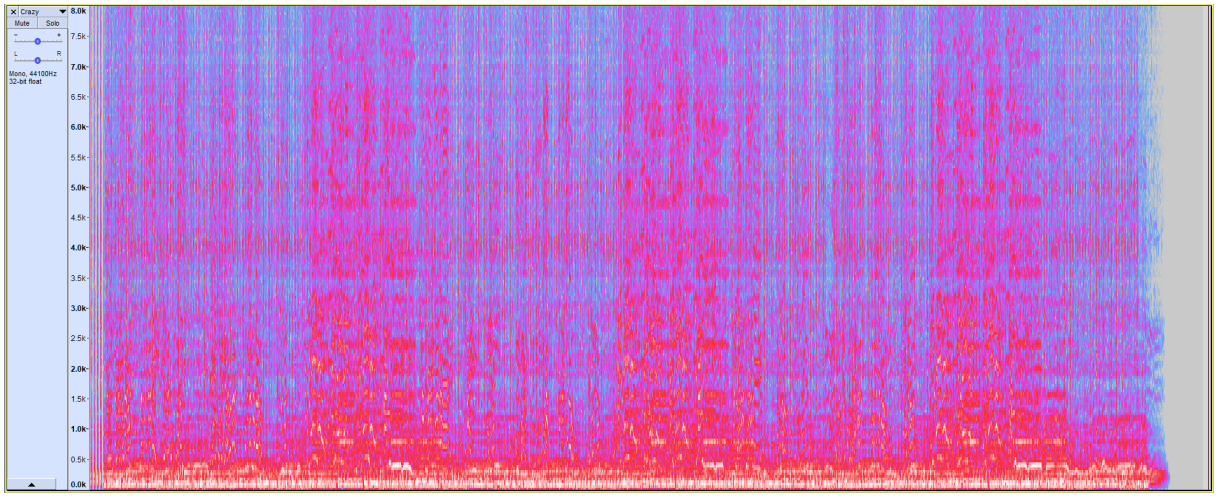

Figura 2.1: Gnarls Barkley - Crazy - Spectogram - 8000hz - Audacity.

2.3 RIFF WAVE Format File

Iniciativa conjunta entre as empresas Microsoft e IBM o formato *Resource Interchange File Format*(RIFF) foi anunciado em agosto de 1991[5] com o intuito de servir como um formato genérico de armazenamento para músicas, imagens, vídeos e multimídia em geral. Adotado por sua simplicidade, este é utilizado predominantemente para arquivos sem compressão. Em seu arquivo de especificação é dado como exemplo o formato *Waveform Audio File Format*(WAVE) (Formato de arquivo para ondas de áudio), também conhecido

frequentemente pela sigla WAV devido a terminação de seu seus arquivos serem .wav, este é uma implementação do formato genérico RIFF. Como prova de sua versatilidade o sistema RIFF de armazenamento encontra novos usos em projetos atuais, como o formato WebP[16] para armazenamento de imagens sem compressão voltado à web, desenvolvido em 2010 pelo Google.

2.3.1 Descrição formato WAV

Revisitado em 1992-94 para a adição de novos tipos de *chunks*[7] (blocos) e em 2002 para a adição de novos métodos de organização de áudios com mais de um canal[8], atualmente o arquivo WAV possui a formação básica[23] para arquivos sem compressão, *Pulse-Code Modulation*(PCM) (método utilizado para a representação digitalizada de sinais analógicos). A Figura 2.2 apresenta a organização do arquivo WAV.

- ChunkID

  Como o arquivo WAVE implementa o formato RIFF este sempre é iniciado com o *ChunkID*, quatro bytes contendo as letras "RIFF" em ASCII (0x52494646 em hexadecimal). Mesmo o formato RIFF armazenando dados em *Little-Endian* todo campo que descrevem *chunks* são armazenados em *Big-Endian*.

- ChunkSize

  Contém o número de bytes presentes no arquivo a partir deste *chunk*. Este serve para controle básico da leitura e é utilizado para verificação de perda de dados caso o arquivo tenha tamanho menor do que o informado. O número de bytes completo do arquivo é *ChunkSize* + 8, pois este não conta os 4 bytes de si próprio e do *chunkID* que o precede.

- Format

  Ultima parte do metadado geral do formato RIFF, este consiste em 4 caracteres em ASCII que identificam o formato a seguir do arquivo, no caso do formato WAVE este é composto pela string "WAVE" (0x57415645 em hexadecimal).

- Subchunk1ID

  Indica o começo do primeiro *subchunk* (bloco contido em bloco maior) do formato WAVE PCM, composto pela string "fmt" em ASCII (0x666d7420 em hexadecimal) este precede as informações de metadado do arquivo.

- Subchunk1Size

  Igual *ChunkSize* este representa o número de bytes presente no *Subchunk1*, para PCM (arquivo sem compressão) este valor é sempre 16.

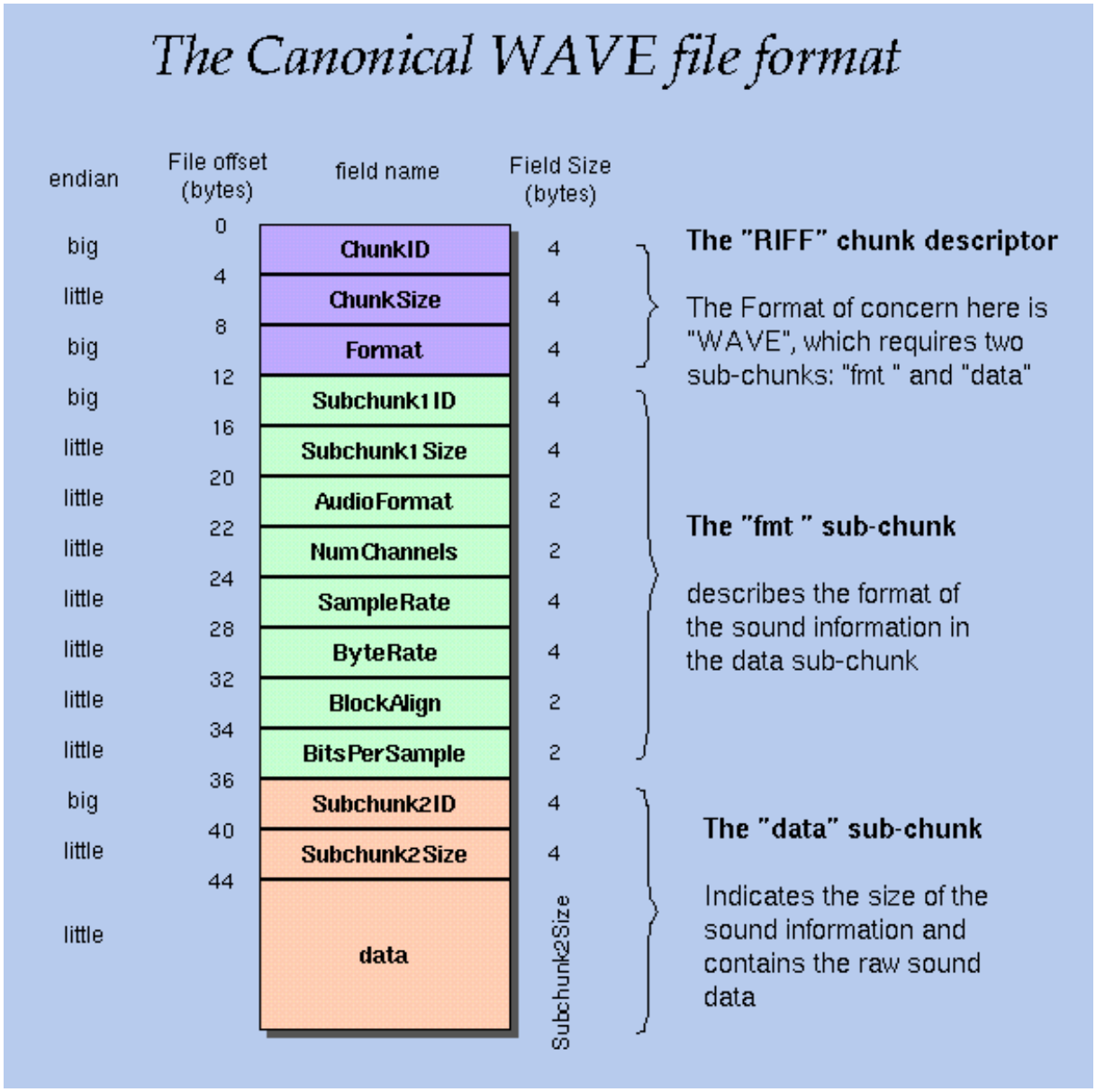

Figura 2.2: Exemplo da estrutura de um arquivo WAVE (Fonte: [33]).

- AudioFormat

  Representa o nível de compressão do arquivo, para PCM sempre será 1.

- NumChannels

  Número de canais de áudio presente, 1 = Mono, 2 = stereo, ... , 6 = 5.1 surround sound, ...

- SampleRate

  O número de *samples* (amostras) por segundo presente no arquivo, a maioria dos arquivos utiliza um *Sample Rate* (taxa de amostragem) de 44.1 kHz, colocada como o padrão da industria para este tipo de arquivo. Mesmo a maioria dos arquivos utilizarem-se de uma taxa de amostragem padrão, existe também a possibilidade de capturar-se áudio a uma taxa maior, consequentemente com suas vantagens e desvantagens[25]:

  Vantagens:

  1. maior fidelidade a onda inicial;
  2. melhor cálculo de harmônicas.

  Desvantagens:

  1. grande acréscimo no tamanho do arquivo;
  2. incompatibilidade com a maior parte dos hardwares e softwares (*hardcoded* para o padrão da industria)

  .

- ByteRate

  Número de bytes necessários para armazenar um segundo de reprodução do arquivo. (*SampleRate* * *NumChannels* * *BitsPerSample*/8)

- BlockAlign

  Número de bytes em somente uma amostra, utilizado para facilitar a leitura dos blocos no *subchunk Data*. (*NumChannels* * *BitsPerSample*/8)

- BitsPerSample

  Número de Bits por amostra de cada canal. O padrão adotado pela industria é de 16 bits, mas utilizar quantidade de bits maiores permite uma imagem mais realista da amplitude conforme o tempo do sinal inicial, isto também possui suas vantagens e desvantagens[25].

- Subchunk2ID

  Como exemplificado no *Subchunk1ID*, este contém a string ”data” em ASCII (0x64617461 em hexadecimal) e simboliza o começo do *subchunck* que contém o conteúdo real do arquivo.

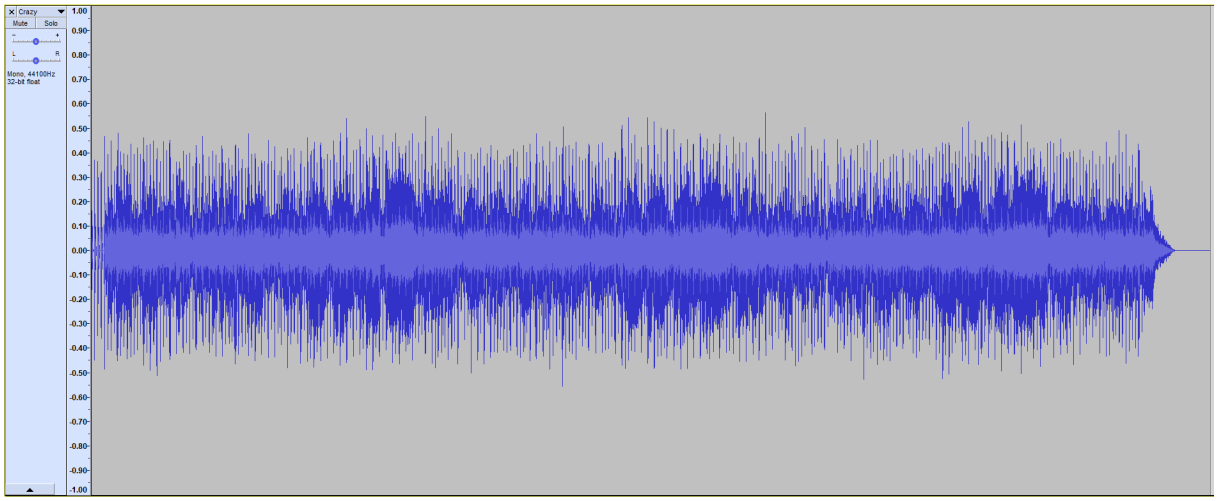

Figura 2.3: Gnarls Barkley - Crazy - WaveForm extraída do arquivo WAV.

- Subchunk2Size

  Número de bytes sequencias contendo os dados do arquivo. (*NumSamples* * *NumChannels* * *BitsPerSample*/8)

- Data

  O *subchunck* que contém a *waveform* do sinal. Este campo apresenta conjuntos de amostras em ordem crescente ao tempo associado a elas, cada conjunto possui as samples de todos os canais naquele determinado tempo.

Também existe o formato RIFX, este tem sua composição de chunks espelhada na do formato RIFF, contudo todos os dados são armazenados em **Big-Endian**, este foi criado para servir aos hardwares implementados pela Motorola na década de 70.

A informação contida no arquivo WAV é uma função de amplitude/tempo do som capturado, como pode ser visto na Figura 2.3 que apresenta um exemplo de forma de onda de uma música contida num arquivo WAV, conforme indicado no nome do formato. A Figura 2.3 apresenta valor de amplitude em uma escala de faixa igual a $-2^{BitsPerSample}$ a $2^{BitsPerSample} - 1$, o complemento de dois formado pela quantidade de bits por amostra armazenada no metadado do arquivo.

2.4 Sistema Auditivo Humano

Composto de 3 seções, o ouvido humano é um sistema especializado em receber ondas sonoras compostas de variações de pressão do ambiente. Para o escopo deste trabalho, consideraremos apenas o ar atmosférico como meio. Transforma as ondas sonoras em

sinais suficientemente claros, eliminando possíveis ruídos, permite o reconhecimento de sons familiares.[28]

O canal auditivo e o tímpano formam a parte responsável pela audição da seção do sistema em contato direto com o ambiente. O canal auditivo consiste em um duto aberto em uma extremidade e fechado pelo tímpano em outra. O tímpano é a membrana responsável pela captura das variações na pressão do ar presente no canal geradas pelas ondas sonoras.

Considerando que sua fisiologia, corresponde a um tubo fechado em uma extremidade com uma membrana, o canal auditivo gera ressonância. Esta faz com que certas frequências gerem uma amplitude de movimento maior no tímpano comparado com outras com a mesma amplitude inicial, pois sua energia sonora foi ampliada por sua passagem pelo canal. Este comportamento define a faixa de frequências em que o ouvido humano consegue identificar ondas sonoras, entre 20Hz e 20kHz. Sons distantes dessa faixa acabam por não produzir uma força considerável no tímpano e por consequência são ignorados pela audição humana.

A faixa de frequências que ressoa pelo canal auditivo pode ser calculada devido a sua forma, um tubo com uma extremidade fechada. A fórmula da frequência de ressonância Fn nesta forma geométrica é apresentada na Equação 2.1, sendo **v** a velocidade do som no meio presente no interior do tubo (neste caso ar, 330 m/s), **L** o comprimento do tubo (2.5cm em média para um adulto) e **n** a harmônica (neste caso somente harmônicas impares são geradas, 1;3;5...).

$$F_n = n \times v/(4 \times L) \tag{2.1}$$

Temos que a frequência de maior ressonância é 3300Hz, como pode ser verificado na Figura 2.4. A Figura denota as intensidades necessárias por faixa de frequência para estas serem percebidas pelo sistema auditivo humano. O limite mínimo percebido pelo ouvido humano ideal é de 20Hz, e o limite máximo é de 20kHz, sendo estes limites passíveis de degradação por idade e outros fatores. Cada linha escura representa as "*isophone curves*" (curvas isofônicas) de nível de intensidade em Phons do número logo acima. Cada curva desta representa para o ouvido humano um som de mesma intensidade. Phon é uma unidade de medida utilizada para medir a intensidade (*loudness*) de uma onda senoidal pura em relação a audição humana, 1 phon equivale a percepção de uma onda senoide em 1kHz a 1 dB. Como pode-se ver na Figura 2.4 a curva isofônica mais inferior intercede nos pontos 60Hz/50dB e 1kHz/0dB, assim podemos perceber que um sinal sonoro puro em 60Hz com volume de 50dB seria imperceptível ao ouvido humano do mesmo modo que um sinal de 1kHz a um volume de 0dB, e um sinal com volume de 70 dB com frequência de 70Hz seria comparado ao som de uma conversa calma, enquanto um sinal em 2kHz com a

mesma amplitude teria a percepção de volume comparada a de um carro em movimento. A frequência em que o ouvido humano é mais sensível é de 3300 Hz, ou seja, nesta frequência a capacidade de percepção de som ocorre até em ínfimas intensidades.

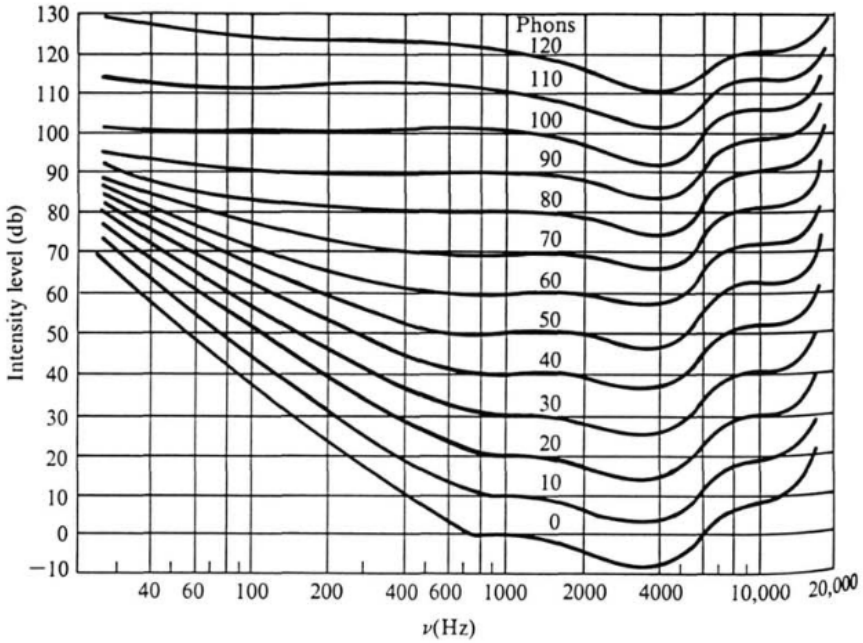

Figura 2.4: Intensidade/Frequência da audição humana (Fonte: [28]).

2.5 Transformada Discreta de Fourier

Apresentada em 1822 no livro *Theorie analytique de la chaleur* por Jean Baptiste Joseph Fourier[15], a ideia da Transformada de Fourier, apresentada na Equação 2.2, propõe que qualquer sinal contínuo no domínio do tempo pode ser representado no domínio de frequência por uma soma de ondas senoidais puras.

$$\mathcal{F}(h) = \int_{\infty}^{\infty} f(x)\mathrm{e}^{-2\pi ixh}dx \tag{2.2}$$

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{i2\pi}{N}kn} \tag{2.3}$$

A transformada discreta de Fourier, apresentada na Equação 2.3, como o nome indica, apresenta o mesmo funcionamento da transformada de Fourier mas aplicada em um espaço discreto. Como toda representação de um espaço continuo na computação obri-

gatoriamente deve ser um conjunto de amostras discretas esta forma da transformada de Fourier acaba por ser a utilizada para análise computacional de ondas.

2.6 Hardware utilizado no projeto

2.6.1 Raspberry Pi

Raspberry pi é um computador *single-board* (de placa única) de baixo custo criado e produzido pela Raspberry PI Foundation. Incrivelmente versátil, este é utilizado em projetos de varias complexidades.

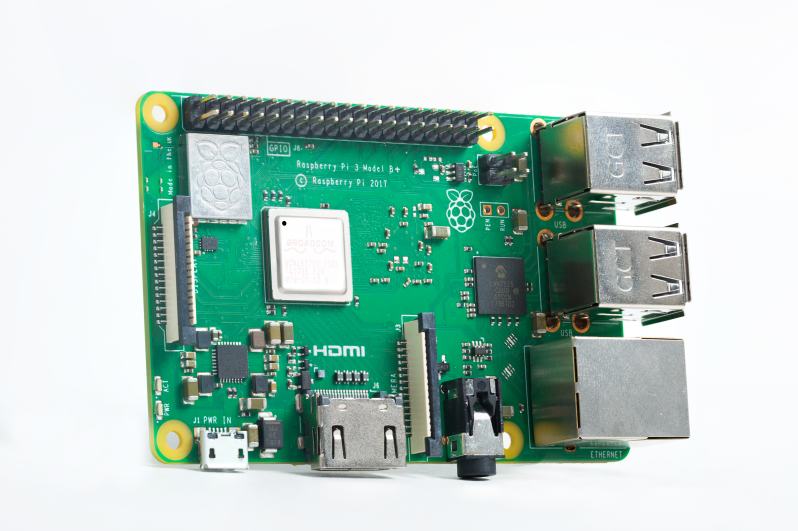

Figura 2.5: Raspberry Pi modelo 3B+.

Desde sua primeira iteração este vem sido adotado como poderosa ferramenta no ensino de eletrônica para alunos de ensino básico[13]. Sua adoção em vez de outros métodos como arduino se devem aos programas de *outreach* (alcance de novos usuários) desenvolvidos pela própria Raspberry PI Foundation, sua capacidade de apresentar uma interface gráfica amigável, possuir sistema operacional integrado (gerando assim um sistema fechado que independe de hardware externo para controle e programação da placa) e por fim a enorme comunidade que acaba por gerar diversos tutoriais e fornecer suporte a usuários que por ventura acabem necessitando.

Disponível em 5 famílias que englobam 11 modelos distintos, os modelos mais novos também se diferenciam da concorrência devido a diversos módulos inclusos diretamente na placa. Estes novos modelos por exemplo possuem Bluetooth, Wi-fi, saída hdmi e saída de áudio integrados [14].

2.6.2 Circuito integado sn74ls164

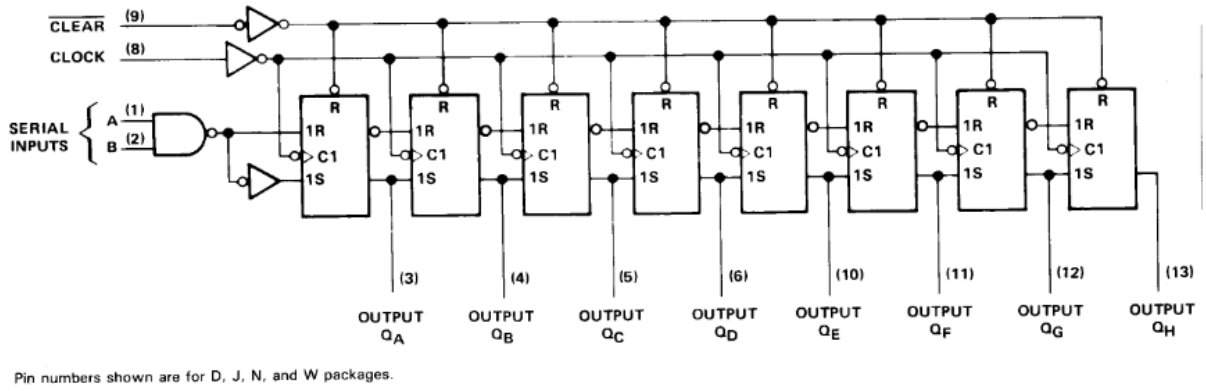

Figura 2.6: Diagrama lógico do chip sn74ls164 (Fonte: [21]).

O circuito integrado de número de série sn74ls164 é um registrador de deslocamento de 8 bits em saída paralela, como pode ser visto na Figura 2.6. Este é composto de 8 flipflops e a entrada de dados é efetuada pela interface serial composta das entradas A e B, podendo utilizar qualquer uma das duas como flag para indicar a entrada de novos dados enquanto a outra seta o estado do primeiro flipflop na descida do sinal do clock(8). Este também possui uma entrada de clear assíncrono, que não depende do *clock* para ser ativado.

2.6.3 Transistor 2N2222

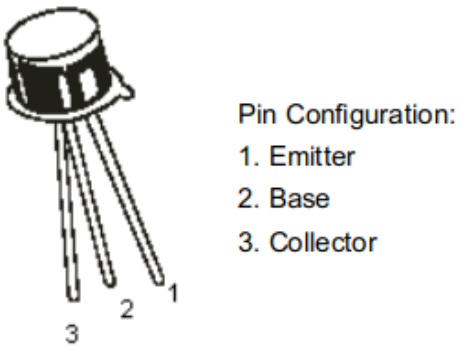

Figura 2.7: Pinagem de um transistor 2N222 (Fonte: [26]).

O transistor NPN de serial 2N2222[26] é comumente utilizado para controle e ampliação de baixas potencias em circuitos com tensões e correntes medianas. Possuindo 3 conectores base, coletor e emissor, vide Figura 2.7. O transistor determina a corrente entre o colector e o emissor com referência na diferença de potencial entre a base e o emissor, permitindo assim controlar uma corrente de maior potência.

2.7 Software Utilizado no projeto

2.7.1 Linguagem Python

Desenvolvido em 1989-1990 por Guido van Rossum[34], Python se destaca perante outras linguagens de programação pela facilidade de seu aprendizado, seu foco em direcionar fortemente a produção de códigos simples e legíveis, e na produção ágil de desenvolvimento e aplicação de seu código.

A obrigatoriedade do uso de indentação, tipagem dinâmica, estruturas de dados de alto nível nativas, portabilidade massiva de código e uma comunidade ativa voltada a popularização deste fazem de Python uma das linguagens de maior crescimento atualmente, ocupando a quarta posição em engajamento de usuários na maior comunidade online de programadores voltada a disseminação do conhecimento[31]. Já para usuários mais experientes, a facilidade de colocar uma ideia em prova com a utilização deste torna Python uma das melhores ferramentas para testes de conceitos rápidos.

2.7.2 Typescript

Apresentado em outubro de 2012, Typescript foi desenvolvido pela Microsoft[24] atendendo à demanda de desenvolvedores que necessitam utilizar o amplo alcance de dispositivos que Javascript possui, mas necessitam de uma linguagem mais sólida para um projeto de larga escala. Em si, Typescript somente implementa tipagem estática, classes e módulos sobre Javascript.

Atualmente a utilização de Typescrypt vem aumentando comparada a estagnação de sua linguagem base. Com um número crescente de novos projetos preferindo a utilização desta, é possível que Javascript venha a ser substituído como a linguagem de maior utilização pelo seu sucessor direto[9]. Tendo como a única desvantagem real perante javascript, a necessidade de uma pré-compilação, que tem custo e tempo ínfimos perante a vantagem de um código mais otimizado e drasticamente mais fácil de produzir e compreender, não é surpresa que vários *frameworks* voltados a desenvolvimentos web e mobile (ex.: Ionic) venham a suporta-lo[6].

2.7.3 Linguagem Java

Desenvolvida no começo da década de 90 na Sun Microsystems pelo time de James Gosling[30], Java teve como força para seu desenvolvimento a necessidade de uma linguagem de sintaxe parecida a C++ mas com portabilidade não oferecida por esta. Java é uma linguagem compilada, mas esta compilação sempre gera o mesmo código em bytecode independente da arquitetura do sistema utilizado para a compilação. A execução de bytecode java sempre ocorre em uma JVM (máquina virtual Java), a qual tem como responsabilidade transformar as instruções do bytecode em instruções da arquitetura em que se encontra, algumas vezes recompilando o programa durante a execução para linguagem nativa.

Mesmo sendo proprietária Java tornou-se a linguagem padrão para várias aplicações, incluindo o desenvolvimento de aplicativos para o sistema operacional Android, porém o uso novas linguagens como Kotlin ameaçam esta soberania.[19]

2.7.4 Ionic Framework

Desde sua versão inicial em 2013, Ionic[4] aposta que o futuro do desenvolvimento de aplicativos mobiles será baseado na utilização de browsers nativos aos aparelhos para a comunicação com o usuário. Utilizando-se da implementação do HTML5 na maioria dos dispositivos mobiles, independente de fabricante ou sistema operacional, e de todos os navegadores web presentes no mercado, Ionic consegue uma ampla portabilidade de código porém, essa portabilidade ocasiona em um processamento mais demorado do que aplicativos projetados diretamente para o OS em questão. Mesmo com esta perda de velocidade, o Ionic framework ainda tem acesso a todas as chamadas nativas dos OS mobiles mais utilizados, tornando-se assim uma atrativa opção para desenvolvedores que não necessitem de um poder de processamento maior diretamente no dispositivo.

2.8 Fenômenos de iteração táctil no corpo

2.8.1 2PD, TPDT

*Two-point discrimination*2PD (Discriminação de dois pontos) é o nome dado para o evento em que o corpo humano consegue distinguir dois objetos em contato com a pele como distintos. *Two-Point discrimination threshold*TPDT representa a distância miníma dentro da normalidade em que os dois objetos possam ser distintamente reconhecidos. O TPDT varia dependendo do local no corpo, vide Figura 2.8, devido à diferença da quantidade de Corpúsculo de Meissner[32], conjuntos de células receptoras de estímulos tácteis.

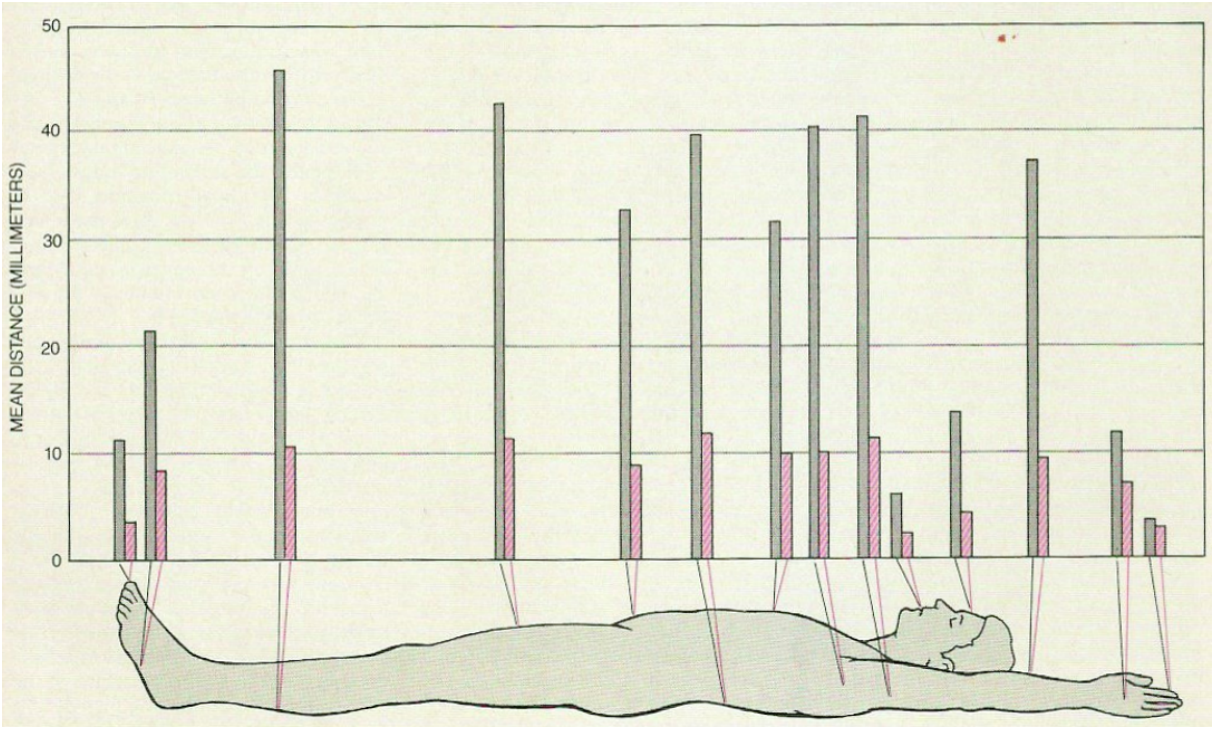

Figura 2.8: 2PD em diversas partes do corpo, as barras cinzas representam o espaço mínimo entre dois pontos de contato onde estes podem ser diferenciados. As barras vermelhas representam a distancia média entre o ponto sentido e um real ponto de contato. (Fonte: [1]).

2.8.2 O Fenômeno Saltatory

Responsável pela "Cutaneous rabbit illusion" o fenômeno Saltatory[1] consiste na sensação falsa de um movimento sobre a pele quando pontos distintos com uma distancia considerável entre eles recebem estímulos ritmados.

2.8.3 O Efeito Tau

O efeito Tau consiste da percepção falsa de uma distância entre dois estímulos ritmizados separados por uma distancia constante mas em frequências distintas, sendo a percepção diretamente proporcional ao tempo entre o acionamento dos dois estímulos.[18]

2.8.4 O Fenômeno Phi

Proposto em 1912 por Max Whertheimer[35] e depois melhor especificado por Edwing Boring[11], o fenômeno Phi consiste de uma sensação de movimento gerado por duas fontes estacionárias pulsando a frequências relativamente altas. Primeiramente encontrado no campo visual, este fenômeno apresenta-se também no espaço tátil, gerando em fontes de

estímulos próximos uma área de movimentos fantasmas que mascaram os locais reais onde se encontram as fontes.

2.9 Pesquisas Semelhantes

2.9.1 Sound Shirt Project

Figura 2.9: Imagem promocional da Sound Shirt (Fonte: [3]).

A *Sound Shirt* (camisa sonora)[17] representada na Figura 2.9, criada pela empresa Cute Circuit [2] consiste de uma camisa contendo terminações hápticas embutidas no próprio tecido e utilizada com software proprietário para o controle do hardware. O preço atual de um dev-kit do projeto é de 3.000,00 GBP[3] (16.457 BRL em 28/11/2019), ou seja de alto custo para um consumidor final no presente momento.

O projeto *Sound Shirt* demonstra a viabilidade do produto final na parte de hardware, tornando-se assim a maior dificuldade, a criação de um hardware de baixo custo capaz gerar uma sensação condizente com a música.

2.9.2 A Comparative Study of Phoneme- and Word-BasedLearning of English Words Presented to the Skin

Apresentado no EuroHaptcis 2018[10], o estudo consiste no teste de dois métodos de ensino para o reconhecimento da fala em um sistema háptico adaptado retratado na Figura 2.10.

O primeiro método baseado na distinção de símbolos fonéticos e depois na formação das palavras utilizando estes, enquanto outro método voltado ao reconhecimento direto das palavras[22].

Ao final, o estudo em questão mostra a necessidade de um estudo sobre como melhor gerir o acionamento dos motores conforme a entrada, para que um usuário treinado consiga especificar o tipo de música reproduzida e venha a criar um entendimento completo sobre os diferentes ritmos e estilos.

Figura 2.10: Hardware utilizado no estudo comparativo de fonemas (Fonte: [22]).

2.9.3 An Enhanced Musical Experience for the Deaf: Design and Evaluation of a Music Display and a Haptic Chair

Testada em 43 deficientes auditivos participantes, o projeto da *Haptic Chair* consiste em uma cadeira contendo alto-falantes de vibração embutidos, que pode ser vista na Figura 2.11, e uma tela contendo vídeos pertinentes a experiencia sonora a ser reproduzida[29]. A cadeira utiliza mais de um alto-falante de vibração. Assim, com a vibração dos alto-falantes se gera um sinal táctil da música que o participante sente pelos braços desta. Ao final, todos os 43 participantes aprovaram a utilização da cadeira e se mostraram dispostos a utilizá-la novamente.

Com um número considerável de testes, o estudo mostra que, do ponto de vista do potencial consumidor, um dispositivo de tradução para vias tácteis de sinais sonoros melódicos é viável, dependendo somente da implementação ter um custo acessível.

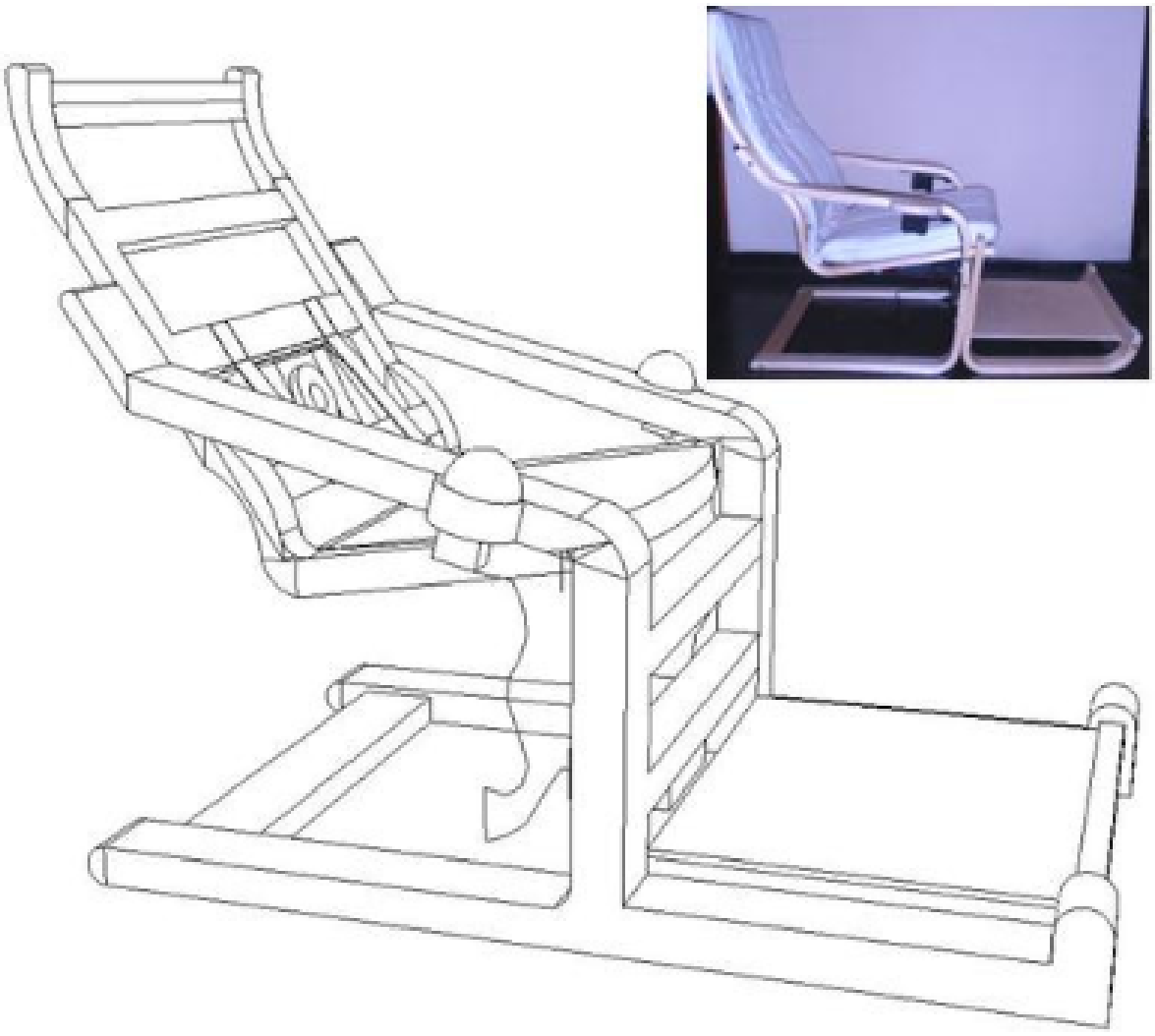

Figura 2.11: Haptic Chair utilizada para testes (Fonte: [29]).

Capítulo 3

Implementação

Este trabalho tem como objetivo a criação de um sistema modular de baixo custo e fácil reprodução que consiga transformar um sinal sonoro musical em um sinal háptico equivalente.

O projeto se divide em duas partes: a criação de um software que gere um sinal capaz de ser reproduzido por uma saída física que permita a sensação táctil; e de um circuito modular de baixo custo que atenda aos requisitos para a reprodução do sinal.

Neste capitulo é tratado o processo de implementação do projeto, abrangendo o software que vem a ser explicado na seção 3.1, e o hardware construído para o projeto que é apresentado na seção 3.2.

3.1 Software

A implementação do software passou por 4 versões, a primeira somente foi um teste de protótipo não documentado que evoluiu à segunda versão quando foi implementada a integração ao hardware e a criação do arquivo HFM. A terceira versão foi o primeiro teste de processamento em tempo real em uma plataforma mobile, implementado em TypeScript utilizando o *framework* Ionic[4] acabou abandonada pela necessidade de um processamento mais rápido do que o *framework* oferece. A quarta e atual versão foi implementada em Java focando somente dispositivos Android. Assim, com o número de dispositivos atendidos menor foi possível implementar um processamento em tempo real. O planejamento de software do projeto aderiu aos seguintes pontos chaves:

- Modularidade:

  O projeto deve aceitar a expansão de novas funcionalidades e um número maior de saídas terminais facilmente e sem a necessidade de mudanças maiores em sua base.

- Custo:

  Efetuar a construção do hardware com um custo mínimo é de suma importância para eventuais reproduções do projeto e expansão de novos módulos.

- Fidelidade:

  A saída táctil deve ser mais fiel possível à entrada. Duas ondas sonoras distintas o suficiente para serem consideradas claramente diferentes ao sistema auditivo humano devem ser reproduzidas no sistema háptico de modo que possam ser diferenciadas uma da outra.

3.1.1 Primeira versão do software

Implementada em python, a primeira versão tinha como finalidade firmar o aprendizado sobre ondas sonoras e como transformá-las em um sinal forte suficiente para a percepção táctil humana. Foi decidido a utilização da *Fast Fourier Trasform* FFT aplicada ao sinal sonoro para a estimação do espectrograma e a análise da magnitude das ondas senoidais de diferentes frequências para a ativação dos diversos motores que gerariam o sinal táctil para o usuário. Projetado para utilização diretamente no Raspberry-PI, o sistema tinha como entrada um arquivo WAV e saída direto aos motores, com uma implementação rápida esta versão exemplificou os maiores problemas que viriam a ser tratados nas próximas:

- Necessidade de pré-processamento:

  Tendo em vista a característica simples desta primeira implementação o modo que o cálculo da FFT foi tratado não permitia um processamento rápido o suficiente para a saída de dados no mesmo fluxo de entrada, assim tornando inviável o processamento em tempo real, e implicando a necessidade de um pré-processamento para arquivos WAV. Este problema veio a ser resolvido em versões futuras do projeto revisando-se o modo que o cálculo da FFT é efetuado utilizando-se um banco de constantes pré-calculadas.

- Necessidade de reprocessamento:

  A falta de um sistema de arquivo para o armazenamento em disco da saída acarretava na necessidade de reprocessamento toda vez que um arquivo WAV fosse carregado, mesmo se este já tivesse sido calculado em reprodução anterior. Em futuras versões do projeto o tipo de arquivo HFM foi projetado e implementado para sanar esta deficiência e possibilitar a criação de peças especificas para o hardware.

- Faixas de cálculo de frequência imutáveis:

  Com as 16 faixas de frequências de 500Hz de 0 até 8000, este modelo se tornou útil para a verificação da transformação de um sinal sonoro de alta frequência e baixa amplitude (uma nota alta de volume baixo que não seria capaz de gerar uma força sobre a pele humana para ser sentida). Em contra partida, este ainda não fazia nada para a simbolizar os timbres do sinal dado. O problema apresentado neste item pôde ser parcialmente solucionado dando ao usuário a possibilidade de modificar as faixas de frequência de acionamento de cada motor, assim com faixas de frequência direcionadas à frequências especificas e com a união de faixas podemos gerar um sinal mais fiel a amplitude de instrumentos e a voz humana.

- Processamento diretamente no controlador do hardware:

  O processamento do sinal sonoro diretamente no hardware de controle dos motores apresenta dois problemas: o primeiro é a diminuição de portabilidade, pois o processamento exige hardware maior do que os encontrados em arduinos simples; e a perda de modularidade, quando o sistema inteiro depende de um hardware único este torna-se preso aquela implementação. A resolução de ambos acaba por exigir *trade-offs*: portar o processamento para um hardware de maior processamento e de fácil acesso como um celular aumenta a mobilidade e em parte a modularidade, mas também aumenta a complexidade exigindo um modo de comunicação sem fio (neste caso Bluetooth) entre este e os motores.

3.1.2 Segunda versão do software

Construída sobre a primeira, a segunda versão[1] visava elaborar um método simples de solucionar o reprocessamento dos arquivos WAV sem uma grande reestruturação do código do processamento em si. Dessa forma foi criado um novo tipo de arquivo chamado, Haptic Feedback Music (HFM). O conjunto de softwares divide-se em 2 programas:

O primeiro definido como "Conversor WAV-HFM" tem como função receber um arquivo de extensão WAV e converte-lo em Haptic Feedback Music (HFM), consistindo dos seguintes arquivos:

- wavEditor.py
- hfmEditor.py
- wav2hfm.py

[1]Repositório :*https://bitbucket.org/TMKlautau/tcc-version-2/src/master/*.

O segundo definido como ”Software de reprodução de HFM” recebe um arquivo HFM e aciona os terminais hápticos nos padrões lidos, este é composto pelos seguintes arquivos:

- hfmHardware.py
- hfmPlayer.py

Ambos, em conjunto, geram pela primeira vez no projeto uma conversão real de um arquivo que representa um sinal sonoro e a sua representação através de um conjunto de sinais tácteis, que possam ser comparados e identificados como similares quando executados em paralelo. A tradução em tempo real ainda era inviável pois neste momento o software necessitava de pré-processamento da entrada.

Escolha da linguagem python

Todo o código da segunda versão do projeto foi elaborado em Python, a escolha desta linguagem deve-se a 4 importantes características:

- Portabilidade
- Integração com o hardware
- Facilidade de leitura
- Extensiva Standard Library

No quesito portabilidade, Python se sobressai sobre outras linguagens pela grande quantidade de bibliotecas funcionais tanto em sistemas Windows quanto linux. Isso é importante já que o controlador de hardware utilizado usa uma distro Linux (Raspbian), enquanto o desenvolvimento do software ocorreu em um sistema Windows e este abrange um público maior.

Em relação a integração com o hardware, Python novamente se destaca por ter um grande apoio da comunidade em torno do hardware utilizado (Raspbery Pi), devido ao baixo custo do projeto e a rede de suporte existente tanto ao hardware como à linguagem, podendo ser facilmente replicado e estendido por outros que por ventura desejarem.

Python também oferece uma facilidade de compreensão de código, permitindo assim um código de fácil manutenção e expansão.

Por final, também se sobressai sobre outras linguagens pela extensiva biblioteca considerada como padrão, permitindo assim que esta versão tenha somente duas dependências externas aquelas já incluídas com a instalação normal do interpretador: RPi.GPIO e numpy, sendo esta última possível de não ser utilizada com uma reescrita do código.

wavEditor.py

O arquivo wavEditor contempla as classes WavFile e Spectogram. A classe WavFile recebe o diretório para o arquivo WAV de entrada e extrai suas informações para fácil acesso. Foi escolhido a extensão WAV pela facilidade da implementação de um leitor, pois esta não apresenta compressão e sua organização por sistema RIFF o faz de uma simplicidade única juntamente com a facilidade de ignorar-se meta-dados presentes no arquivo que não seria de utilidade para esta versão inicial do projeto. A classe peca em performance, podendo ser otimizada, mas segue as especificações do projeto onde, excluindo-se estruturas de dados básicas, todo o processo é apresentado em fácil leitura.

A classe Spectogram recebe uma instância da classe WavFile e opcionalmente um inteiro descrito como *refreshRateCeilingIn*, o qual representa o maior valor possível de espectrogramas calculados por segundo pelo algoritmo, que no final correlaciona diretamente à frequência das saídas táteis. Para o cálculo do espetrograma foi utilizado o algoritmo FFT de menor complexidade que atendia aos requisitos (Cooley–Tukey). Como o algoritmo exige um segmento de amostras cujo tamanho seja uma potência de 2, se calcula o número mínimo de valores em cada segmento que este será aplicado dividindo-se a quantidade de amostras por segundo pela quantidade máxima de segmentos por segundo. Após obter-se o tamanho mínimo podemos chegar facilmente a maior potência de 2 que mais se aproxima deste, possuindo então um tamanho de segmento que atenda aos requisitos do algoritmo Cooley–Tukey e ofereça uma frequência aceitável ao produto final.

Aplicando o cálculo da FFT sobre as amostras do objeto WAV conseguimos uma matriz contendo os valores da amplitude de cada frequência com relação ao tempo que somadas geram o sinal inicial como pode ser visto na Figura 3.1.

Como o projeto foi escrito com o intuito de modularização este calcula o espectro de frequências completo do sinal fornecido, no caso da Figura 3.1 cuja frequência de amostragem do sinal inicial é de 44.1kHz (frequência padrão dos arquivos encontrados em cds de música comercial) esta corresponde a um espectrograma cuja janela de frequências vai de 0 até 22.05kHz.

Como estamos tratando de uma peça musical, a faixa de frequências que realmente é utilizada consiste da faixa de melhor sensitividade do ouvido humano, de 20Hz à 8kHz como foi apresentado na seção 2.4. Assim o método *cutToFreq* verifica quais amostras pertencem até a faixa indicada (cada amostra representa uma faixa de frequências correspondente a $44100i/2048$, sendo i o índice da amostra) e reorganiza a matriz, neste caso o espectrograma desejado é apresentado na Figura 3.2. A veracidade desta pode ser verificada pela sua similaridade com o apresentado em programas de edição de áudio do mercado, vide Figura 3.2 e Figura 2.1

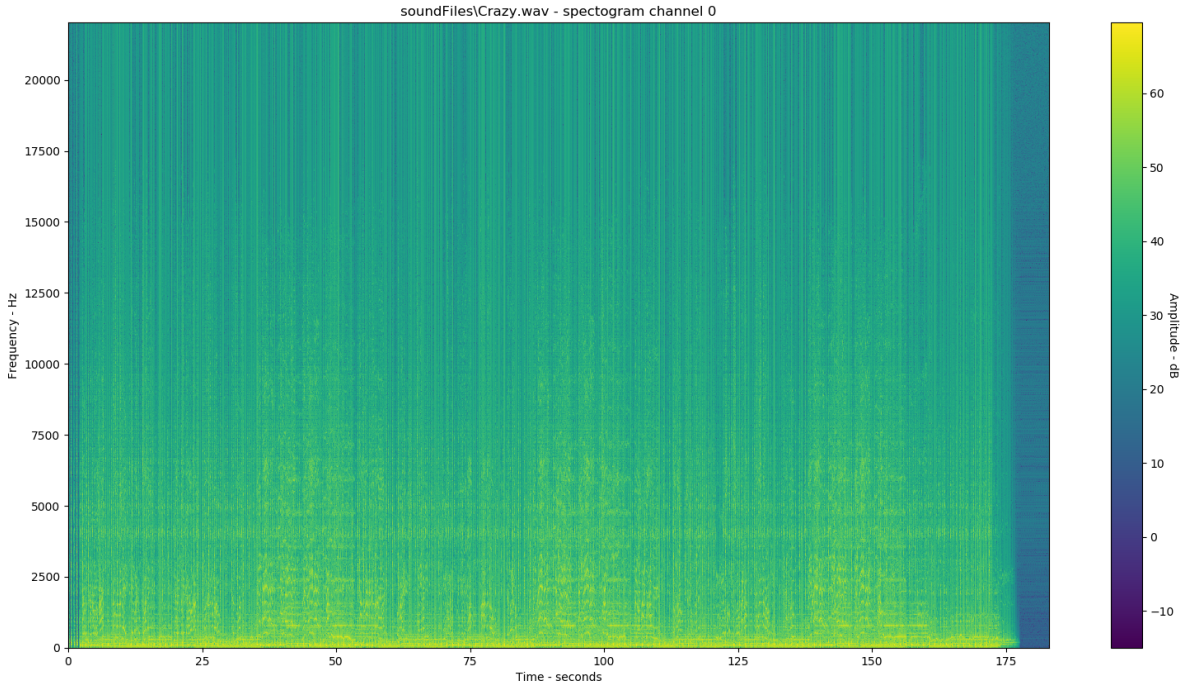

Figura 3.1: Gnarls Barkley - Crazy - Spectogram - limite 20kHz.

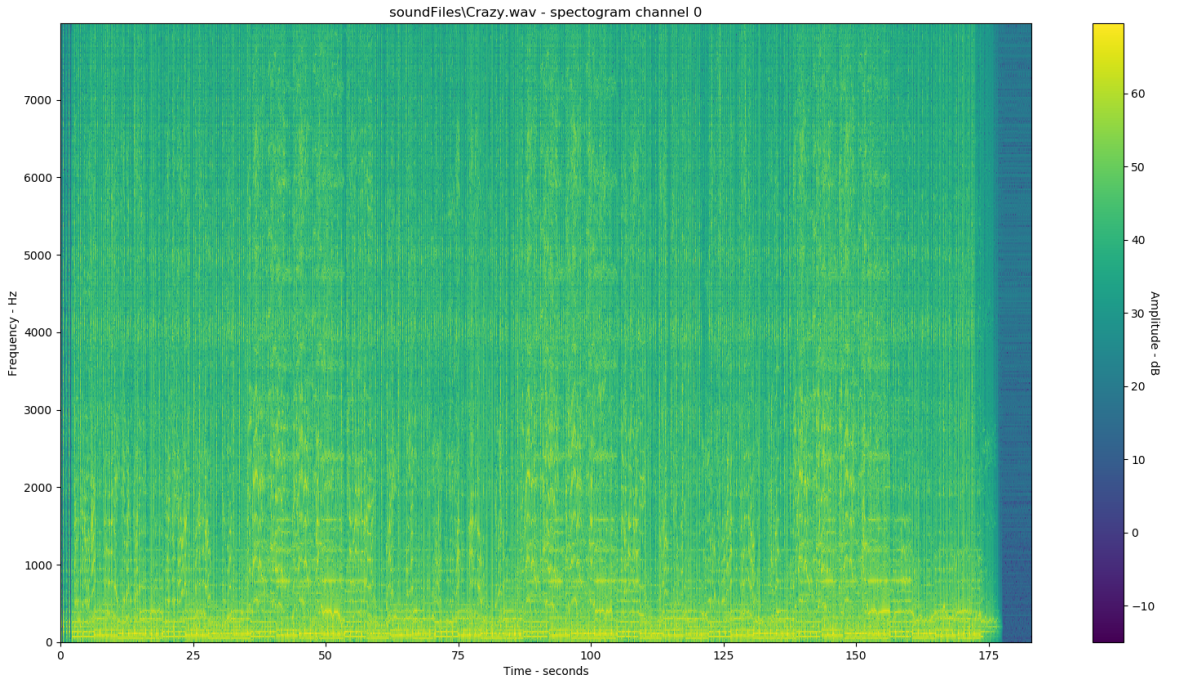

Figura 3.2: Gnarls Barkley - Crazy - Spectogram - limite 8kHz.

Com o espectrograma calculado, temos um conjunto de dados que facilmente podemos adaptar para um sistema onde faixas de frequências sejam correspondentes a saídas

físicas que produzam um sinal capaz de ser identificado pelo tato humano. Para a versão atual do projeto foi escolhido pequenos vibradores encontrados em celulares devido ao seu baixo custo e facilidade de aquisição. O código foi escrito para funcionar com uma quantidade de saídas múltipla de 8, nessa versão foram utilizadas 16 saídas por canal de áudio apresentado, cada saída apresenta uma correlação com uma faixa de 500Hz do espectrograma.

Identificado as faixas podemos definir um limite para o acionamento da saída física, para isso é possível modificar a intensidade da vibração variando a quantidade de corrente que é alimentada, mas esta opção aumenta a complexidade do circuito a ponto de sua implementação ser inviável nesta versão pela necessidade de um maior de processamento. Também é possível agrupar um ou mais motores em uma determinada faixa. Agrupando 2 geramos 3 tipos de intensidade, mas em troca ou aumentaríamos o número de motores ou aumentaríamos a faixa responsável pela ativação, assim perdendo fidelidade ao sinal original. Então a melhor opção, no caso de se aumentar o número de terminações, era simplesmente continuar alimentando cada motor com uma quantidade de corrente constante quando ativado e desativar quando a faixa de frequência não mostrar amplitude suficiente.

Assim, com 16 terminações correspondendo a faixa de 0 até 8kHz atribuímos 500Hz para cada. Além disso para uma primeira avaliação definimos a amplitude de corte em 50dB, assim fazemos uma média simples de todos os valores da faixa, e caso este apresente-se maior que o valor de corte para aquela posição no tempo, o motor associado estará ligado, e caso apresente-se menor, desligado. Com essa primeira avaliação o conjunto de 16 saídas comporta-se como apresentado na Figura 3.3, na qual pode-se verificar a similaridade com a Figura 3.2. Podemos verificar os 3 picos nas frequências acimas de 2kHz representados pelo som dos instrumentos de corda de altura aguda na música nos segundos 36, 87 e 139, também podemos verificar a presença ou não da voz do cantor na faixa de frequências correspondente a 1kHz até 2kHz, e a presença da percussão e instrumentos de corda grave que fornecem o ritmo da música em todo seu tempo na faixa de frequências correspondente a 20Hz até 1kHz.

hfmEditor.py

O arquivo hfmEditor consiste somente da classe HFMFile, que foi criada para manipulação de arquivos da extensão HFM.

Contendo quatro métodos simples, a classe HFMFile contém toda a informação necessária para a criação do arquivo HFM ou para a reprodução no hardware especifico. O método *initFromFile* recebe um arquivo HFM e carrega seus dados O método *populateData* recebe os dados diretamente e os inicializa no objeto, o *saveToFile* cria e salva

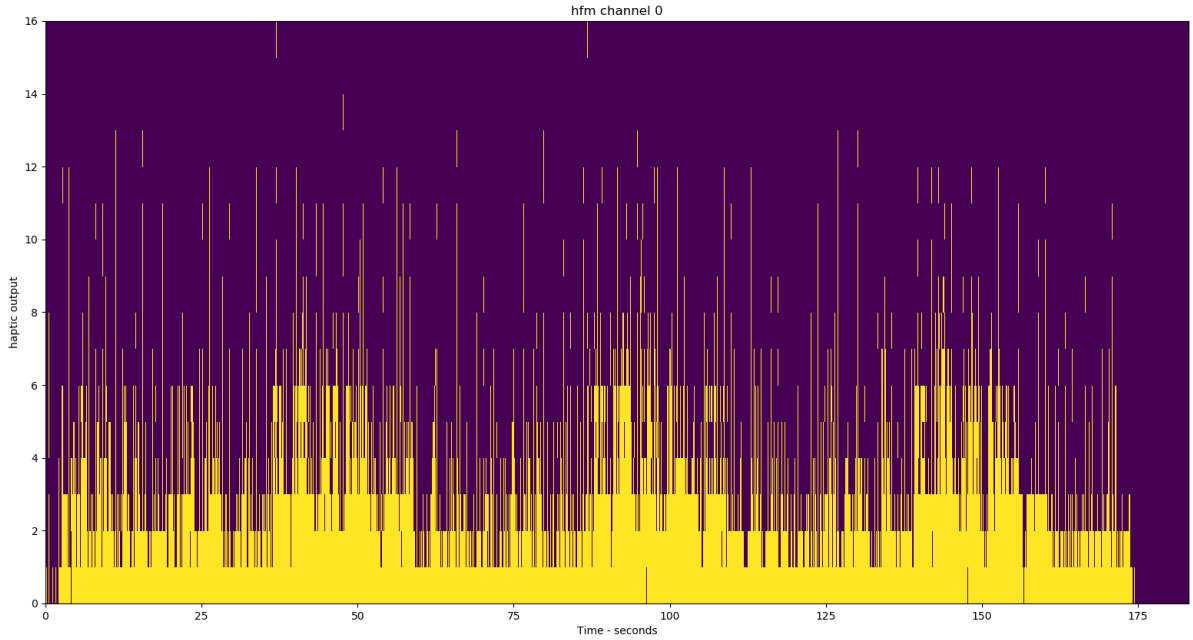

Figura 3.3: Gnarls Barkley - Crazy - Corte em 50dB.

Tabela 3.1: Estrutura do arquivo HFM.

Offset	**Nome**	**Size**	**Conteúdo**
0	Format	4	Apresenta as letras HFFM em ASCII
4	Meta Subchunck Descriptor	4	Apresenta as letras meta em ASCII
8	NumChannels	2	Número de canais de áudio
10	Frequency	8	Frequência em Hz do arquivo
18	HapticOutNum	2	Número de outputs por canal
20	Data Subchunk Descriptor	4	Apresenta as letras data em ASCII
24	Data Size	4	Número de segmentos a seguir
28	Segment Data	*	Data da tabela de segmentos

o arquivo HFM, e *dumpInfo* apresenta no *prompt* todas as informações pertinentes ao objeto.

Arquivos HFM

Com o cálculo das ativações das terminações táteis efetuado, é gerado um arquivo HFM que descreve qual das terminações táteis é ativada em cada instante de tempo. O formato do arquivo HFM é derivado do modo de armazenamento RIFF mostrado anteriormente, mas levemente modificado (por utilizar notação big-endian, este seria mais comparável ao modo RIFX). A estrutura do arquivo é apresentada na Tabela 3.1, a qual apresenta propriedades dos *subchunks*. O conteúdo de cada *subchunk* é detalhado a seguir:

- Chunk Descriptor

  Consiste das letras HFFM em ASCII (0x4846464D) e representa que o arquivo é realmente do formato HFM. Caso não esteja presente não é necessário continuar e se retorna um erro ao tentar ler arquivos de outros formatos

- Meta Subchunck Descriptor

  Consiste da palavra meta em ASCII (0x6D657461), representa o inicio do *subchunk* contendo metadados

- NumChannels

  Representa o número de canais de áudio presente no arquivo. Cada canal é tratado diferentemente e necessita um conjunto de saídas. Normalmente arquivos wav possuem um (mono) ou dois (stereo) canais, mas alguns formatos possuem mais, como o 5.1 surround sound que apresenta cinco. Mesmo possuindo mais de um canal a maioria dos arquivos stereo possui somente a mesma informação duplicada em ambos ou ligeiramente modificada, assim mesmo utilizando-se somente um módulo de saídas ainda é possível de utilizar-se a maior parte de arquivos stereos.

- Frequency

  Armazena em *float* a frequência em Hz do do arquivo, isto é, quantas vezes por segundo as saídas terão seu valor modificado.

- HapticOutNum

  Contém o número de saídas esperado por canal para a reprodução do arquivo. Como cada módulo projeto consiste de 8 saídas, a quantidade de módulos por canal necessário é o menor múltiplo de oito que seja maior que o número de saídas.

- Data Subchunk Descriptor

  Consiste da palavra "data" em ASCII (0x6D657461), representa o inicio do *subchunk* contendo os dados reais do arquivo

- Data Size

  Este é o número de colunas na tabela contendo os dados do arquivo (segmentos) multiplicado pelo número de canais. Como cada saída ocupa um bit por segmento, cada módulo contendo 8 saídas ocupa um byte. Assim, o número real de bytes que o campo de dados ocupa é representado pela multiplicação de *Data Size* pelo número de módulos necessários para a reprodução de todos os canais do arquivo(vide *hapticOutNum*).

- Segment Data

  Os dados reais do arquivo, consiste dos valores das saídas organizadas sequencialmente por canal e tempo.

wav2hfm.py

O executável que utiliza-se das classes apresentadas em wavEditor.py e hfmEditor.py, aceita os seguintes parametros:

- -f (File To Open)

  Recebe o diretório para o arquivo WAV a ser convertido.

- -o (Number Of Outputs)

  Recebe o número de saídas presente no hardware onde o arquivo HFM será reproduzido.

- -r (Refresh Rate Ceiling)

  Recebe o valor máximo da frequência final do arquivo HFM.

- -g (Show Graph)

  Flag que se setada gera o gráfico do espectrograma e do arquivo HFM. Exemplos de gráficos gerados podem ser vistos nas Figuras 3.2 e 3.3

- -i (Dump Info)

  Flag que se setada mostra informações dos arquivos após conversão.

- -h (Help)

  Flag que se setada mostra a mensagem de ajuda contendo todos os parâmetros possíveis e termina a execução.

hfmHardware.py:

Este arquivo contém uma pequena biblioteca de funções utilizadas para controlar os dois módulos de motores utilizados nesta versão, utilizando os pinos *General-purpose input/output*(GPIO) presentes no Raspberry-PI.

hfmPlayer.py:

Executável que recebe um arquivo HFM e, utilizando-se das ferramentas implementadas no arquivo hfmHardware.py, aciona as saídas de forma ritmada.

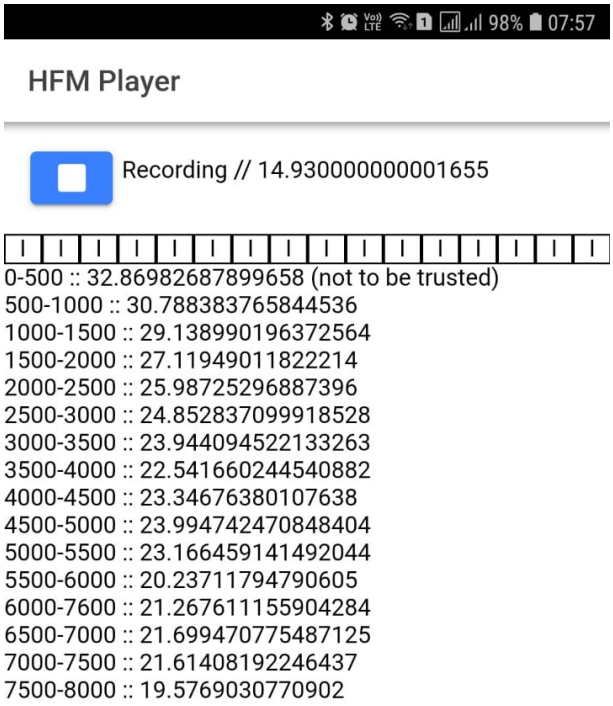

Figura 3.4: UI básica no momento da captação em tempo real da terceira versão do software.

3.1.3 Terceira versão do software

A terceira versão[2] do software visava passar o processamento ao celular e, utilizando-se do microfone embutido neste, realizar o processamento em tempo real dos sinais sonoros ao redor do usuário. Visando a portabilidade do código o primeiro protótipo utilizou-se do framework IONIC[4], pois este podia ser compilado tanto para IOS, Android, ou utilizado em *browsers* que tivessem permissão de acesso a um microfone. Esta versão foi rapidamente abandonada quando verificou-se que o framework IONIC não permitia uma velocidade de processamento aceitável para o cálculo da FFT em tempo real na precisão necessária. Para que a ativação dos motores tenha um *delay* mínimo e ocorra na mesma velocidade que o sinal sonoro captado pelo microfone o espectrograma teve de ser construído com 32 pontos sobre uma captação de 16kHz, o que gerava uma impossibilidade da utilização de janelas menores que 500Hz, pois tornava o resultado para qualquer frequência abaixo disto não confiável. Mesmo com estes problemas esta versão serviu de início ao desenvolvimento do processamento em tempo real, a separação do hardware dedicado ao controle dos motores e o dedicado a captação do sinal sonoro e processamento, o foco ao desenvolvimento do projeto para sistemas mobile (mesmo que neste momento com uma UI básica como pode ser visto na Figura 3.4).

[2]Repositório :*https://bitbucket.org/TMKlautau/tcc-version-3/src/master/*.

3.1.4 Quarta versão do software

Quarta[3] e última até o presente momento, esta versão soluciona o problema das faixas de cálculo de frequência (permitindo um controle maior perante a faixa de cada um dos motores), e o problema do cálculo em tempo real. Utilizando-se da IDE Android Studio portou-se a terceira versão somente para android, em contra partida limita-se assim o número de aparelhos alcançados. Porém o ganho de velocidade de processamento sem utilizar-se o envelopamento do Ionic permite que se capture o microfone a 44100Hz e se aplique a FFT em janelas de 4096 amostras, permitindo assim uma diferença de somente 10,76Hz entre cada medida.

Estrutura do código

Como esta versão foi implementada em java para o sistema Android sua estrutura lógica segue espelhada na construída para a segunda versão. Em suma o código é divido nos módulos lógicos e fragmentos apresentados a seguir. Na versão atual do software existem 3 módulos de grande importância, estes são:

- Bluetooth:

  Composto pelo arquivo BluetoothControl.java. O módulo de bluetooth é implementado utilizando-se das APIs para a utilização do hardware de bluetooth do aparelho para se conectar com o sistema de controle dos motores de vibração sem a necessidade de fios.

- Microphone:

  Contendo somente o arquivo MicControl.java este módulo foi implementado com a função de utilizar-se das APIs de controle do microfone do dispositivo móvel para a captação em tempo real dos dados de entrada para o acionamento dos motores.

- HfmLogic:

  Possuindo 3 arquivos, Complex.java, FFTControl.java e HapticsControl.java, o módulo HfmLogic apresenta a implementação do cálculo do espectrograma de um sinal de entrada utilizando-se do cálculo da FFT pelo algoritmo Cooley–Tukey como foi explicado na implementação da segunda versão do software.

O usuário nesta versão do software controla o estado dos motores a partir dos seguintes fragmentos:

[3]Repositório :*https://bitbucket.org/TMKlautau/tcc-version-4.2/src/master/.*

Figura 3.5: UI básica no momento da captação em tempo real da quarta versão do software.

- RealTimeProcessingFragment

  Fragmento independente que serve como UI para a captura de áudio, contém informações sobre o estado da conexão com o sistema de controle dos motores e o estado de ativação destes, um botão de acionamento da captura de áudio em tempo real e outro para o fragmento hapticsSettingsScreen. Vide Figura 3.5.

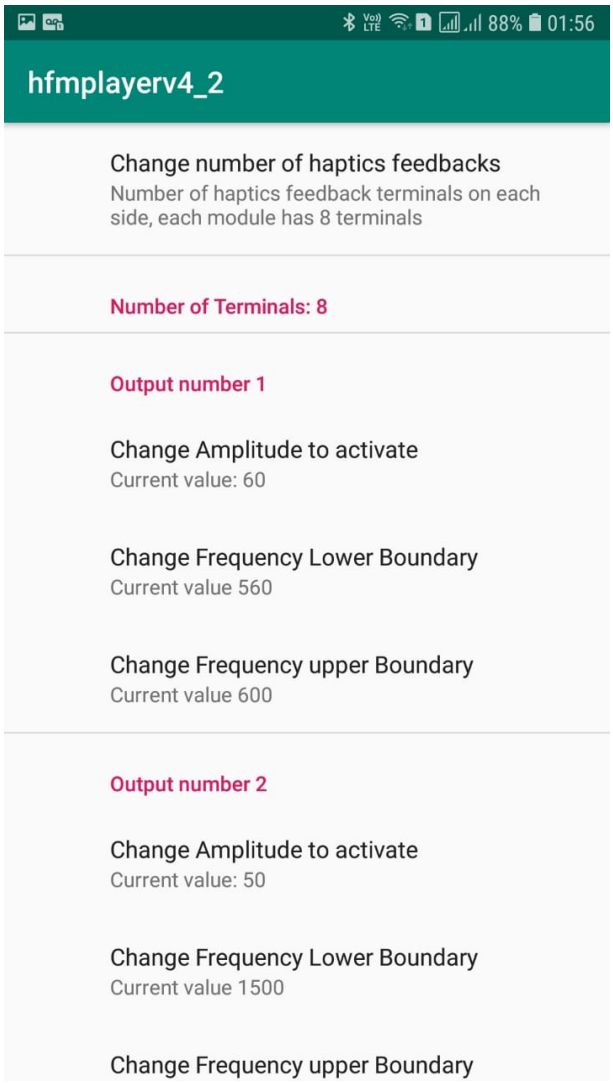

Figura 3.6: UI básica usada para a definição das configurações dos motores na quarta versão do software.

- HapticsSettingsScreen

  Implementado no módulo HFMLogic, o fragmento HapticsSettingsScreen consiste nas configurações pertinentes ao cálculo dos arrays de ativações dos motores. Neste fragmento o usuário define manualmente a quantidade de motores que estão ligados ao sistema e qual o espectro de frequências e a intensidade que a ativação de cada um terá como base para o cálculo. Vide Figura 3.6.

3.2 Hardware:

O hardware constitue de um circuito para acionamento dos motores de vibração, o qual é interfaceado pela placa Raspberry Pi 3B. Nas duas primeiras versões do software deste projeto o mesmo era executado em sua totalidade nesta placa. Nas versões posteriores o software é parcialmente executado no dispositivo móvel (processamento do sinal sonoro) e parcialmente na placa (acionando os motores de vibração através do controle de GPIO). O circuito de acionamento dos motores de vibração, visto na Figura 3.7, consiste em um conjunto de módulos de motores, cada um responsável pelo acionamento de um motor de vibração, representado por um círculo contendo a letra M na Figura 3.8. Cada módulo contém um flip-flop ligado a base de um transistor 2n2222 [26] que controla o fluxo de corrente para um motor de vibração.

Figura 3.7: Motor de vibração 0834.

Para melhor uso de espaço nas *breadboards* e uma construção modular, o protótipo lida com junções de 8 módulos de motores ligados a um módulo de controle composto de um chip sn74ls164 [21], este por sua vez consiste de 8 flip-flops em série como mostrado na Figura 2.6.

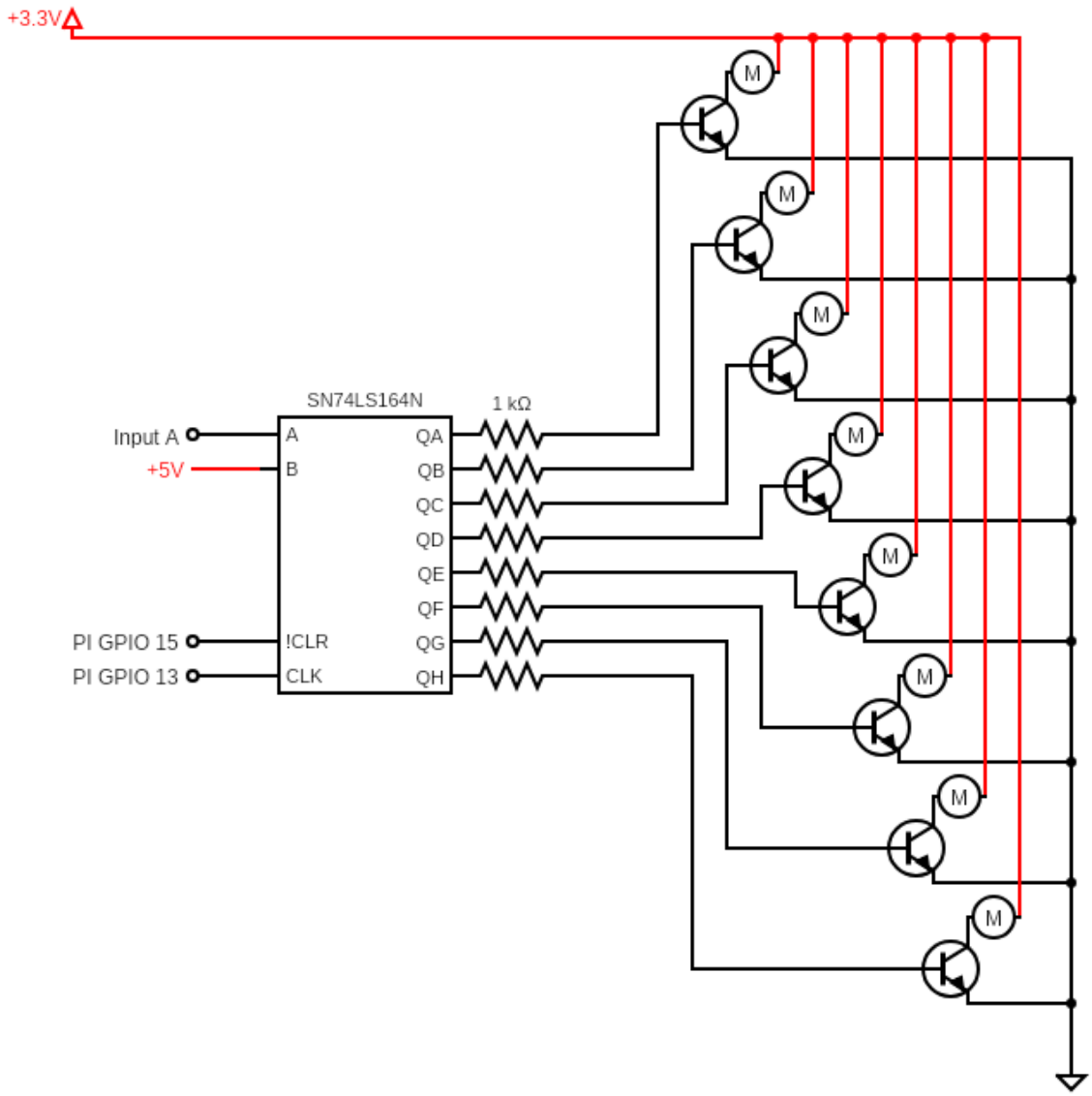

Figura 3.8: Esquemático do módulo de controle.

3.2.1 Componentes utilizados no protótipo

- O motor de vibração usado mostrado na Figura 3.7 possui número de série 0834 e as especificações técnicas pertinentes estão listadas na Tabela 3.2
- O transistor 2n2222 teve suas informações detalhadas no Capítulo 2.
- O circuito integrado de número de série sn74ls164 teve suas informações detalhadas no Capítulo 2.

Tabela 3.2: Especificações do Motor 0834 (Fonte: [12]).

Model Number	BM0834
voltage range	2.5 V - 4.0 V DC
Direction of rotation	Any direction
Speed(RPM)	9000-12500rpm
Starting current(A)	90ma
Continuous Current(A)	70ma

Figura 3.9: Fonte de Alimentação 5V/3,3V para *Protoboard*.

3.2.2 Sistema de alimentação do protótipo

Um dos maiores problemas encontrados no desenvolvimento do projeto foi a necessidade de um sistema de alimentação portátil que ofereça a corrente necessária para a ativação simultânea de todos os motores utilizados. Para o primeiro teste foi utilizado uma fonte simples ligada diretamente à *protoboard* mostrada na Figura 3.9. A fonte usada possui corrente máxima de saída menor que 700mA a 3,3V. Quando acionados os 16 motores estes exigiam 90mA para inicio de operação e 70mA em operação contínua. O acionamento de diversos motores simultaneamente acabava por gerar uma vibração funcional, mas de menor intensidade do que a possível.

Após o primeiro teste foi testado a alimentação diretamente por uma fonte de carregamento de celulares que em tese emitia 2,1A a 5V diminuídos a 3.3 com 2 diodos colocados em série. Com 16 motores ligados simultaneamente a corrente máxima exigida seria de 1,44A. O carregador teve uma performance melhor do que a fonte, mas este ainda não se mostrou capaz de alimentar 16 motores simultaneamente.

Por final, devido a necessidade da portabilidade do projeto foi testado um powerbank da marca PINENG cuja saída oferecia 2,1A a 5v. Esta solução mostrou-se infrutífera pelo fato deste desligar-se sozinho após um tempo, talvez pela necessidade de ter alguma

Tabela 3.3: Número de componentes utilizados para o primeiro protótipo.

Motor de vibração BM0834	16
Transistor 2n2222	16
Raspberry pi	1
Fonte de alimentação para Protoboard	1
sn74ls164	2

voltagem nos terminais de passagem de dados da porta USB para sinalização de que existe um dispositivo conectado.

3.2.3 Versões do hardware utilizadas no projeto

O primeiro protótipo funcional foi projetado para testes com 16 módulos de saída háptica integrados a 2 módulos de controle, o que permitia a reprodução de arquivos de músicas na segunda versão do software. Com corte de faixas de frequência em 500Hz por saída e arquivos stereos com corte de 1kHz por saída, mesmo que funcional esta versão não permitia uma fácil identificação da música. No caso de arquivos stereos, as faixas de frequência tão grandes mapeadas em somente uma saída geravam uma infidelidade não aceitável com o arquivo original. Como primeira tentativa de solução foram montados 2 outros módulos de controle e 16 novos módulos de saída háptica. Assim com cada canal do arquivo stereo possuindo 16 motores de vibração cada um poderia ser mapeado a uma faixa de 500 Hz. O número de componentes utilizados para esta primeira versão encontra-se na Tabela 3.3. A diferença para a versão utilizada na segunda versão do protótipo foi a redução de 16 motores por hemisfério para 8.

Capítulo 4

Testes

Este capítulo apresenta os testes realizados para a validação deste trabalho e a estimativa de custo do protótipo.

4.1 Testes

Nesta seção são apresentados os dois testes efetuados até o presente momento e seus resultados, ambos foram efetuados com um deficiente auditivo na Universidade de Brasília (UnB).

O primeiro teste é apresentado na Seção 4.1.1, já o segundo teste é mostrado na Seção 4.1.2.

4.1.1 Primeiro teste

Este teste teve como objetivo validar o primeiro protótipo do hardware desenvolvido neste trabalho em um potencial usuário final. Para isto, foi testado o posicionamento dos motores de vibração sobre o corpo do usuário com o intuito de verificar as sensações do mesmo.

Neste teste foram usados 16 motores de vibração, os quais foram posicionados exclusivamente no braço e antebraço do usuário. Estes 16 motores foram distribuídos conforme pode ser visto na Figura 4.1. Efetuou-se primeiramente um teste simples posicionando os motores em pares simétricos em relação ao centro do braço, com o objetivo de verificar se o sinal gerado por estes seria identificado na posição correta de cada um e, dessa forma, descobrir o melhor posicionamento e o número necessário de motores em cada braço.

O teste consistiu de 41 ativações de conjuntos de motores, com resultados apresentados na Tabela 4.1. A primeira coluna da tabela apresenta o identificador do teste ativações. Cada teste de ativação consiste no acionamento de um conjunto de motores, descritos na

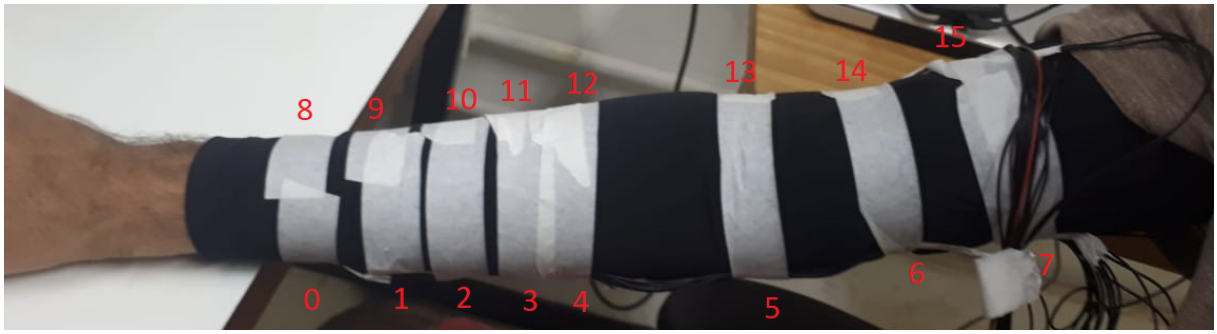

Figura 4.1: Posição dos outputs no primeiro teste de hardware.

coluna ”Motores Testados”. Após a ativação dos motores foi instruído ao participante à informar quais motores este sentia vibrar. Estas respostas por sua vez encontram-se na coluna ”Motores Sentidos” na mesma linha correspondente aos ativados.

Resultados

Com base nos resultados presentes na Tabela 4.1 verificamos os seguintes problemas com este posicionamento dos motores:

- Os motores de 8 a 12 e de 0 a 4 apresentam fenômeno Phi:

  Devido a suas proximidades estes motores acabam por mascarar uns aos outros. Dessa forma, verificou-se que a presença de mais de um motor em uma área de pouca sensibilidade no melhor dos casos tornou-se irrelevante, ou no pior dos casos, apresentou um detrimento a fidelidade do sinal a ser reproduzido. Isso pode ser constatado nos testes de de ativação 16 e 30 na Tabela 4.1.

- A ativação de mais de 2 motores ao mesmo tempo não gera o resultado esperado:

  Esse resultado foi possivelmente causado por uma falha de projeto no sistema de alimentação dos motores. Nesta primeira versão, com a ativação de muitos motores ao mesmo tempo, a intensidade de vibração acaba por ser reduzida. Por exemplo, foi constada que a ativação de 3 ou mais motores simultaneamente mascara o sinal da maioria destes. Isso pode ser constatado nos testes de de ativação 16, 18 e 32 na Tabela 4.1.

Por final o primeiro teste permitiu uma visualização real do projeto, permitindo assim verificar os problemas inerentes ao hardware utilizado e os problemas gerados pelo posicionamento equivocado dos terminais no braço e antebraço.

Tabela 4.1: Resultados do primeiro teste do hardware.

ID de Ativação	**Motores Testados**	**Motores Sentidos**
1	15	8
2	14	13
3	1	1
4	8	8
5	3	5
6	0	0
7	4	6
8	7	7
9	6	6
10	7	7
11	4	4/5
12	11	11
13	10	10
14	3	3
15	7/15	10/15
16	3/5/7/9/11	10
17	3/6	4/6
18	8/13/15	12/15
19	8/14	12/14
20	8/12	8/12
21	8	8/12
22	9	8/12
23	10	8/12
24	8/9	9/10
25	9/10	9/10
26	8/12	8
27	8/11	8/10
28	0/2	0
29	0/1	0/1
30	3/4	3
31	0/15	0
32	0/14/15	14/15
33	0/15	15
34	9/15	8
35	0/15	0
36	1/15	1
37	2/15	4
38	3/15	4/15
39	7/15	7
40	6/13	6/13
41	5/13	6

4.1.2 Segundo teste

Este teste teve como objetivo validar o protótipo deste trabalho em que o software é portado para um aplicativo de celular, e a iteração com o hardware é realizada por meio da tecnologia bluetooth. Dessa forma, buscou-se validar a portabilidade do cálculo da matriz de ativações dos motores no celular do usuário, e a utilização do microfone do celular para a captação e processamento de sinais sonoros em tempo real.

Além disso, o segundo conjunto de testes teve como função verificar novos posicionamentos nos motores, e como estes são afetados pelos fenômenos de iteração táctil no corpo apresentados nos primeiros testes. Nesse teste foram utilizados apenas 8 motores de vibração dispostos conforme as Figuras 4.3 e 4.4. O segundo dia de testes foi composto dos seguinte 5 experimentos:

- Verificação de intensidade

  Nesse experimento foi verificado se os problemas encontrados anteriormente na alimentação do sistema persistiam, fazendo com que os motores não apresentem força necessária para efetuar um sentimento satisfatório na pele do usuário. Com a expansão do número de motores além dos 8 testados, pode ser necessário uma modificação no sistema de alimentação do protótipo. Em tese o protótipo neste estado é passível de alimentação por um *powerbank* (carregador portátil para celulares facilmente encontrado no mercado).

- Verificação de 2PD

  A verificação de 2PD consiste na ativação conjunta dos motores em proximidade para definir se a distância escolhida entre estes deveria permitir uma boa diferenciação das ativações destes. Neste novo teste de posicionamento de motores tentou-se respeitar a distância miníma para a distinção entre dois estímulos[1] nas suas determinadas localizações.

- Verificação de Saltatory Phenomenon

  Foi necessária a verificação da possibilidade de a frequência da mudança do sinal gerar a ilusão de nome Cutaneous rabbit gerada pelo fenômeno saltatory.

- Verificação do funcionamento com uma música pré-definida

  Teste para melhor calibração de frequências da música Phantom of the Opera - Sierra Boggess e Ramin Karimloo (Classic BRIT Awards 2012) cujo espectrograma pode ser visto na Figura 4.2

- Verificação do funcionamento com música livre

  A última parte do teste consistiu no teste do protótipo com músicas escolhidas pelos participantes.

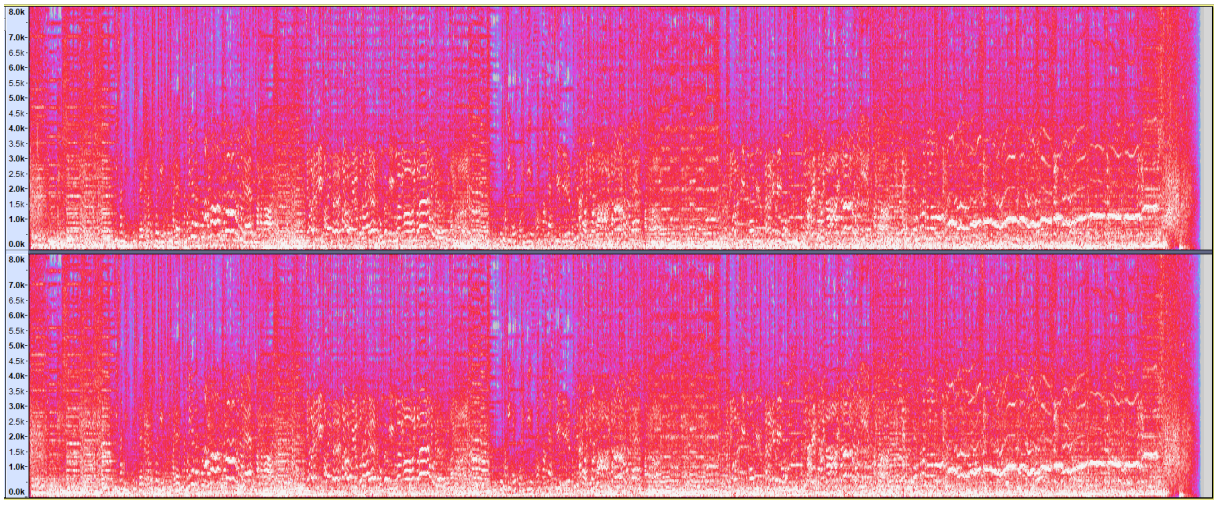

Figura 4.2: Sierra Boggess, Ramin Karimloo - Phantom of the Opera - Spectogram - 8kHz.

Resultados

Ao final os resultados delimitados por tópicos foram:

- Verificação de intensidade

  A intensidade dos motores utilizados atendeu as necessidades atuais. Mesmo funcional foi notado a falta de um controle maior sobre a intensidade de vibração dos motores individualmente no sistema, colocando a necessidade de uma implementação de *Pulse-Width Modulation* (PWM) em uma versão futura.

- Verificação de 2PD

  No teste de verificação de 2PD cujos resultados apresentados encontram-se na Tabela 4.2 pode-se verificar a necessidade de maior distância entre os motores posicionados nos braços. Devido à inexperiência, ao posicioná-los, alguns motores encontravam-se desviados de suas posições iniciais. Para uma versão futura uma nova reorganização do posicionamento nos braços será necessária.

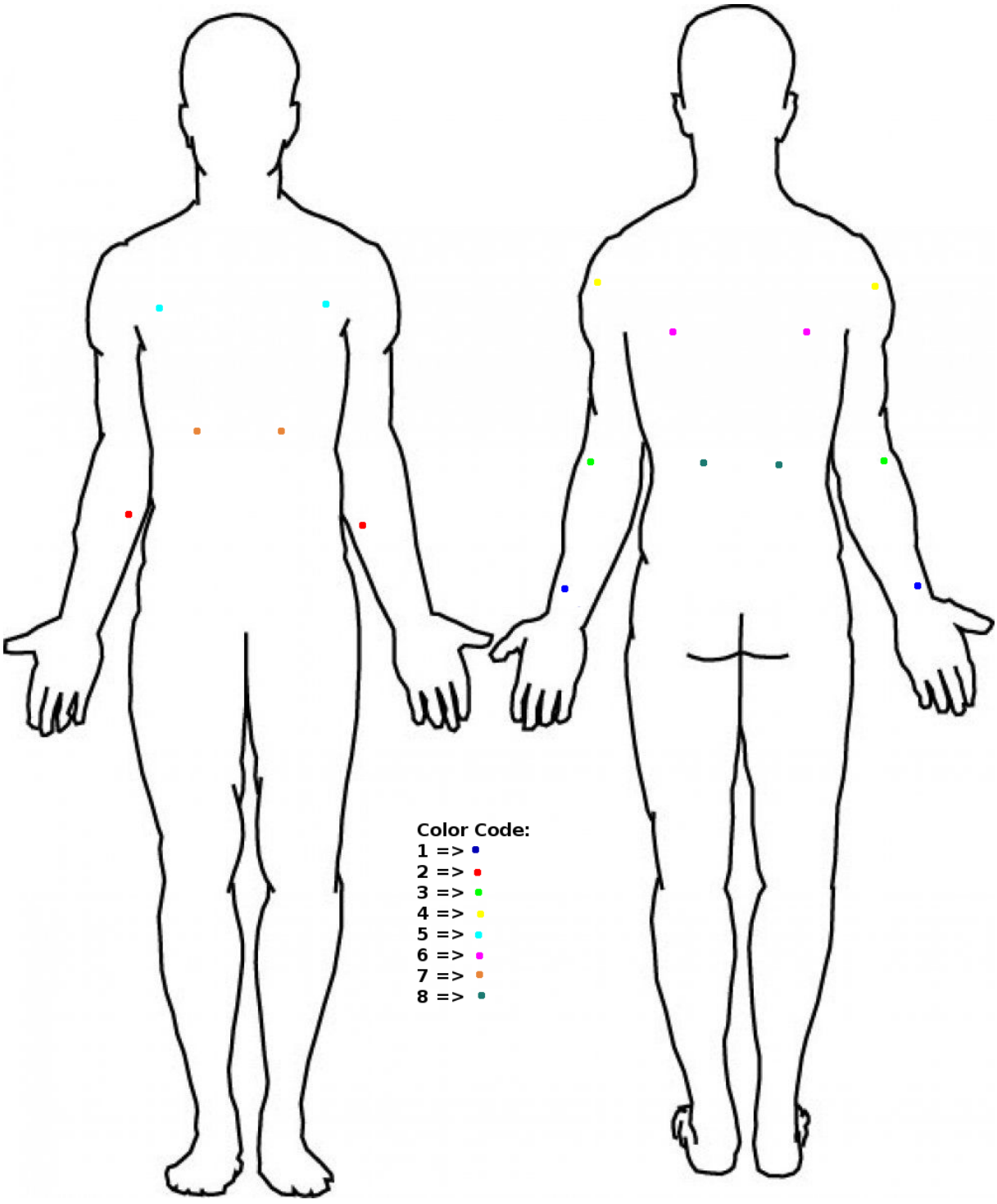

Figura 4.3: Posição das saídas no segundo teste de hardware.

Figura 4.4: Foto da posição dos outputs no segundo teste de hardware.

- Verificação de Saltatory Phenomenon

  Através do teste de ativação e desativação do motor no posicionamento de número 1 a uma frequência de aproximadamente 100Hz (frequência máxima apresentada em caso de uso normal do protótipo) não foi apresentado desvio referente ao Salatory Phenomenon.

- Verificação do funcionamento com uma música pré-definida

  O teste do protótipo reproduzindo a música apresentada na Figura 4.2 demonstrou a capacidade deste de transladar sensações sentidas pelo sistema auditivo ao táctil de modo que mesmo variações da intensidade e timbre na voz aguda da cantora principal fossem percebidas.

- verificação do funcionamento com música livre

  Como pode-se verificar nos vídeos[1] o protótipo demonstrou-se funcional para músicas escolhidas no momento e até capturando voz a *cappella*.

Os vídeos do funcionamento do protótipo podem ser verificados na url: (`https://www.youtube.com/playlist?list=PLpXRCTEGQpQikVAO7_kX7zJKcP1VvCt1M`)

[1] `https://www.youtube.com/playlist?list=PLpXRCTEGQpQikVAO7_kX7zJKcP1VvCt1M`.

Tabela 4.2: Resultados da segunda Verificação de 2PD.

Test ID	**Outputs Testados**	**Outputs Sentidos**
1	1/2	1
2	2/3	3
3	4/6	4
4	5/7	5/7

Tabela 4.3: Custos de componentes do hardware.

Componente	**Custo Por Unidade**	**Quantidade**	**Custo Total**
Motores de vibração	R$ 1,4	16	R$ 22,4
Cinta Elástica Com Velcro	R$ 6,5	8	R$ 52
Jaqueta	R$ 60,7	1	R$ 60,7
Raspberry PI 3B	R$ 168	1	R$ 168
2N2222	R$ 0,4	16	R$ 6,4
sn74ls164	R$ 3,5	2	R$ 7
fonte de alimentação	R$ 6,5	1	R$ 6,5

4.2 Estimativa de custos para reprodução

O protótipo apresenta custo final de R$323. Os custos estimados para uma reprodução do hardware do protótipo nesta versão por componente estão apresentados na Tabela 4.3. Estes custos representam a construção de uma unidade somente e podem ser reduzidos com produção em larga escala e modificações futuras apresentadas no capitulo 5.

Capítulo 5

Conclusão

5.1 Considerações finais

Este trabalho desenvolveu um protótipo para a conversão de um sinal sonoro em sinais táteis. Para isto, um software realiza a conversão do sinal sonoro e faz uma interface com um módulo de hardware que aciona motores de vibração, os quais geram sinais táteis na pele do usuário.

A conversão de sinais sonoros para táteis pode ter como entrada um arquivo sonoro já gravado (apresentado na segunda versão do software deste projeto) gerando um arquivo chamado HFM, com um mapa de ativação dos motores de vibração. Além disso, essa conversão pode ser realizada em tempo real, ou seja, um aplicativo de celular obtém os sinais sonoros através do microfone convertendo-os em sinais táteis em tempo real. Assim, esse projeto possui versões de software embutido diretamente no hardware como processamento delegado a aplicativo para dispositivos móveis. Esse aplicativo comunica-se com o módulo de hardware através da tecnologia bluetooth.

Foram realizados testes com a participação de um deficiente auditivo para verificar a viabilidade deste trabalho, assim como o posicionamento dos motores de vibração no corpo do usuário. Através destes testes foi constatado que o protótipo cumpriu seus objetivos de forma com que o deficiente auditivo pode ter uma noção de ritmo e timbre associado ao sinal sonoro de entrada.

O protótipo não apresenta fonte de alimentação móvel nesta versão por imprevistos, estes apresentados na Seção 3.2.2.

Com um custo final de R$323 e sendo possível alterar o número de motores de vibração e seus limiares de ativação o projeto produziu um protótipo funcional dentro dos requisitos propostos.

5.2 Trabalhos futuros

5.2.1 Software

- Criação, leitura e reprodução de arquivos HFM na plataforma mobile.

  Implementada na segunda versão do software, a leitura e criação de arquivos HFM não for portada ainda para a quarta e atual versão, com o algoritmo do cálculo do arquivo HFM e o controle dos motores já implementados no módulo HfmLogic deste. Faltando apenas implementar um módulo para leitura e gravação de arquivos na memória do dispositivo.

- Melhorar a interface gráfica do aplicativo.

  A interface atual infelizmente é pífia e não atende às necessidades de um usuário final. Para uma versão final deve-se implementar uma interface de melhor apresentação visual e de maior facilidade de uso.

- Implementação de algoritmo para o cálculo de faixas de frequências e amplitudes de ativação em tempo real.

  Atualmente a faixa de frequência e o ponto de ativação da amplitude dos motores no processamento em tempo real são controlados diretamente pelo usuário. Para uma experiência melhorada poderia-se implementar um algoritmo para definição destes em tempo real baseado na própria entrada.

5.2.2 Hardware

- Modificação do sistema de alimentação do protótipo.

  Como apresentado no Capítulo 3 o sistema de alimentação do protótipo ainda encontra-se deficiente. Será necessário uma pesquisa de outros métodos.

- Implementação de PWM para melhor controle dos motores de vibração.

  Um dos modos de melhorar a fidelidade da saída para o sinal de entrada é a implementação de controle por PWM dos motores, assim colocando a ativação não em um ponto especifico de amplitude, mas sim em uma faixa gradiente desta.

- Novos testes para melhor posicionamento sugerido dos motores na parte do braço e antebraço.

  Como apresentado no Capítulo 4 o último posicionamento dos motores no braço e antebraço ainda apresentam 2PD. Assim novos testes devem ser efetuados para melhor entendimento do que ocorre e como pode-se limitar este problema.

- Redução de custos substituindo Raspbery PI 3B por arduino

  Como a partir da terceira versão do software a placa é utilizada somente para o recebimento dos dados via bluetooth e acionamento dos GPIO ,esta pode ser substituída por um Arduíno ATtiny85 e um módulo bluetooth, reduzindo o preço da placa para controle dos motores de 165 reais para aproximadamente 40 reais.

Lista de Abreviaturas e Siglas

2PD Two-point discrimination. 15, 42, 47

ASA Acoustical Society of America. 4

FFT Fast Fourier Trasform. 21, 24, 31

GPIO General-purpose input/output. 29, 34, 48

HFM Haptic Feedback Music. 21–23, 26–29, 47

JVM Java Virtual Machine. 15

PCM Pulse Code Modulation. 6, 7

PWM Pulse-Width Modulation. 42, 47

RIFF Resource Interchange File Format. 5, 6, 24

TPDT Two-Point discrimination threshold. 15

UnB Universidade de Brasília. 38

WAV Waveform Audio File Format. 6, 9, 21, 22, 24

WAVE Waveform Audio File Format. 3, 5, 6

Referências

[1] Frank A. Geldard e Carl E. Sherrick. Space, time and touch. *Scientific American*, 255:90–5, 08 1986. 16, 41

[2] Cute Circuit Corporation. Cute circuit website. 17

[3] Cute Circuit Corporation. Shop page of the sound shirt dev-kit. 17

[4] Drifty Corporation. Ionic framework docs page. 15, 20, 30

[5] IBM Corporation e Microsoft Corporation. Multimedia programming interface and data specifications 1.0, 1991. 5

[6] Ionic Corporation. Benefits of typescript. 14

[7] Microsoft Corporation. New multimedia data types and data techniques, 1994. 6

[8] Microsoft Corporation. Multiple channel audio data and wave files, 2002. 6

[9] Stack Overflow Corporation. Developer survey results - 2018. 14

[10] Hong Z. Tan Emanuele Ruffaldi Antonio Frisoli Domenico Prattichizzo, Hiroyuki Shinoda. *Haptics: Science, Technology, and Applications: 11th International Conference, EuroHaptics 2018, Pisa, Italy, June 13-16, 2018, Proceedings, Part II*. Lecture Notes in Computer Science 10893. Springer International Publishing, 1st ed. edition, 2018. 17

[11] EDWING.BORING. Sensation and perception in the history of experimental psychology. 16

[12] Jinlong Machinery |& Electronics. C0834b011f datasheet. 36

[13] RaspberryPi Foundation. raspberrypi.org about us page, 2019. 12

[14] RaspberryPi Foundation. raspberrypi.org documentation page, 2019. 12

[15] Jean-Baptiste-Joseph Fourier. *Théorie analytique de la chaleur*. Paris : F. Didot, 1822. 11

[16] Google. Webp container specification, 2019. 6

[17] Junge Symphoniker Hamburg. Sound shirt project website. 17

[18] Harry Helson. The tau effect—an example of psychological relativity. *Science*, 71(1847):536–537, 1930. 16

[19] Google Inc. Develop android apps with kotlin. 15

[20] Merriam-Webster Inc. sound wave definiton. 4

[21] Texas Instruments. 8-bit parallel-out serial shift registers datasheet - sn74ls164.pdf, 1988. 13, 34

[22] Yang Jiao, Frederico M. Severgnini, Juan Sebastian Martinez, Jaehong Jung, Hong Z. Tan, Charlotte M. Reed, E. Courtenay Wilson, Frances Lau, Ali Israr, Robert Turcott, Keith Klumb, e Freddy Abnousi. A comparative study of phoneme- and word-based learning of english words presented to the skin. In Domenico Prattichizzo, Hiroyuki Shinoda, Hong Z. Tan, Emanuele Ruffaldi, e Antonio Frisoli, editors, *Haptics: Science, Technology, and Applications*, pages 623–635, Cham, 2018. Springer International Publishing. 18

[23] Prof. Peter Kabal. Audio file format specifications, 2017. 6

[24] Microsoft. Typescript roadmap. 14

[25] Monty (monty@xiph.org). 24/192 music downloads ... and why they make no sense. 8

[26] Multicomp. 2n2222 low power bipolar transistors, 2006. 13, 14, 34

[27] Acoustical Society of America. Standards Secretariat.; American National Standards Institute. *American national standard : acoustical terminology*, volume 1994. New York : Standards Secretariat, 1994. 4

[28] University of Notre Dame. Notes on physics in medicine, 2004. 10, 11

[29] Suranga Nanayakkara; Elizabeth Taylor; Lonce Wyse; S. H. Ong. An enhanced musical experience for the deaf: design and evaluation of a music display and a haptic chair, 2009. 18, 19

[30] Oracle. The history of java technology. 15

[31] Stack Overflow. Developer survey results - 2019. 14

[32] Michel Paré, Robert Elde, Joseph E. Mazurkiewicz, Allan M. Smith, e Frank L. Rice. The meissner corpuscle revised: A multiafferented mechanoreceptor with nociceptor immunochemical properties. *Journal of Neuroscience*, 21(18):7236–7246, 2001. 15

[33] Craig Stuart Sapp. Wave pcm soundfile format. 7

[34] Guido van Rossum. A brief timeline of python. 14

[35] Max Wertheimer. Experimentelle studien über das sehen von bewegung. zeitschrift für psychologie, volume 61, 1912, p. 161–265. 16