Software Para Conversão de Som em Partitura

Josepe Augusto Pereira Fornari, Leandro Rosniak Tibola

Instituto Federal de Educação, Ciência e Tecnologia Farroupilha – Campus Frederico Westphalen

Frederico Westphalen – RS – Brasil.

josepe.2021000267@aluno.iffar.edu.br,

leandro.tibola@iffarroupilha.edu.br

Abstract. *The act of writing sheet music is very common for maintaining a song with a similar sound, disregarding the artist's interpretation. However, to write it, one must either do it the old-fashioned way, using paper and pen, or use a program. For many, however, playing the instrument is simpler than writing, and although there are now software programs that convert the sound of an instrument into sheet music, they require the use of a MIDI cable connected to the instrument, usually a digital piano or keyboard, excluding non-electronic instruments. For those who do not have a MIDI cable, they must convert the recorded audio into a .midi file. The sheet music obtained through this conversion becomes something entirely different from the original music. Although direct audio reading also generates errors, the proposed software aims to eliminate as many of these inconsistencies as possible.*

Resumo. *O ato de escrever partituras é algo muito comum para manter uma música com uma sonoridade semelhante, desconsiderando a interpretação do artista, porém para escrever é necessário ou fazê-lo à moda antiga, utilizando papel e caneta, ou usando um programa. Porém para muitos é mais simples tocar o instrumento ao invés de escrever e, por mais que atualmente existem softwares que convertem o som de um instrumento em partitura, faz-se necessário o uso de um cabo MIDI conectado a tal instrumento, geralmente um piano digital ou teclado, excluindo instrumentos não eletrônicos. Porém para aqueles que não possuem um cabo MIDI, estes devem converter o áudio gravado para a extenção .midi. A partitura obtida por essa conversão torna-se algo totalmente diferente da música original. Embora fazer a leitura direta do áudio também vá gerar erros, o software proposto busca eliminar o máximo possível de tais inconsistências.*

1.Introdução

Com o avanço da tecnologia, as pessoas se acostumaram com a velocidade e eficiência fornecida pelo computador. Porém, na música, a escrita de partituras ainda é demorada e trabalhosa, mesmo com softwares para esse fim. Um método bem rápido de escrita, é a utilização de um cabo MIDI para transferir de um instrumento exatamente as notas que foram tocadas para a máquina, permitindo que o compositor escreva uma partitura gastando apenas o tempo para tocar a peça em questão. O maior problema é que na ausência de um cabo MIDI, o que pode ser feito é apenas a gravação de áudio e sua conversão para a extensão '.midi', essa conversão captura os sons ambientes e, no caso de instrumentos de múltiplas cordas, os sons pertencentes a série harmônica, isso altera a partitura de modo que esta fica totalmente "corrompida" e não representa mais a música tocada. O software em questão visa fazer a escrita da partitura sem realizar a conversão para um arquivo .midi, tentando remover ao máximo os sons ambientes e de série harmônica.

O objetivo deste trabalho é o estudar as formas de captura do som e desenvolvimento de um software que possa gerar uma partitura mais bem definida. Em questão de forma, seu objeto de teste foi reduzido para peças escritas de 1 a 3 vozes, desde que estas sejam alocáveis sem necessidade de linhas suplementares dentro de uma clave específica e, apesar de possível, também foi decidido suprimir as barras de compasso e notação de tempo, devido a quantia de imagens que necessitariam ser editadas para que isso fosse possível e, o pouco tempo disponível.

2.Referencial Teórico

Nesta seção serão descritos os conceitos fundamentais abordados neste trabalho.

2.1 - Partitura

A partitura é como o roteiro de uma peça de teatro, através dela é possível detalhar com precisão como uma peça deve ser tocada, porém, para isso é necessário que esta siga regras e possua algumas características especificas. Uma partitura consiste em pautas (cinco linhas horizontais), claves, armadura de clave, fórmula de compasso, tempo, notas musicais, dinâmicas etc. O conjunto desses elementos define como o compositor sugere que uma música seja tocada. Abaixo serão abordados mais detalhadamente componentes da partitura.

2.2 – Tempos musicais

As notas (figuras) musicais são definidas tanto numericamente como nominalmente, cada nota tem uma aparência

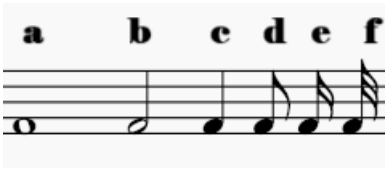

Figura 1. tempos musicais

Semibreve, visto na Figura 1a, é atualmente a nota mais longa usada, é representada pelo número 1 na fórmula de compasso.

Mínima, presente na Figura 1b, é o tempo da mínima é equivalente à metade do tempo da semibreve, é representada pelo número 2.

Semínima, demonstrada na Figura 1c, é seu tempo é metade do tempo da mínima, ou seja, um quarto do tempo da semibreve, portanto seu número é 4.

Colcheia, visto na Figura 1d, assim como as anteriores, seu tempo é metade da semínima e um oitavo da semibreve, tendo número representante 8, em vários momentos as colcheias, e as próximas notas a serem listadas, podem ser encontradas ligadas por uma barra.

Semicolcheia, visível na Figura 1e, pode aparecer ligada por 2 barras horizontais, seu tempo é metade da colcheia e seu número é 16.

Fusa, apresentada na Figura 1f, pode aparecer ligada por 3 barras horizontais, seu tempo é metade da semicolcheia e seu número é 32, a partir da fusa existem a semifusa e quartifusa. que não são muito utilizadas e seguem o mesmo padrão de desenho.

Além dos tempos positivos existem também os tempos negativos, presentes na Figura 2, eles têm o mesmo nome dos anteriores apenas ao invés de produzir som o tempo negativo mantem-se em silencio durante o período estimado.

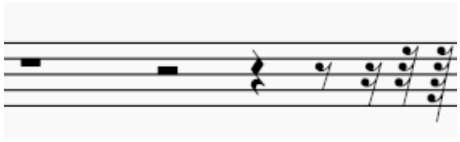

Figura 2. Tempos negativos

2.3 – Fórmula de compasso

A fórmula de compasso é encontrada no início da partitura junto da clave e da armadura de clave, nela são encontrados dois números, o número inferior representa o tempo musical (mencionado anteriormente), e o número superior representa quantas figuras daquele tipo cabem dentro do compasso, exemplos de fórmula de compasso bastante usadas são 4/4 (também representado como C), 3/4, 2/4, 6/4.

2.4 – Clave

A clave indica qual a altura que será tocada a nota, existe 3 tipos de clave, sol, fá e dó, porém uma mesma clave pode representar uma altura diferente dependendo de qual linha

do pentagrama ela for posicionada. Na figura 3 pode-se verificar uma imagem contendo as 3 claves respectivamente nas suas posições mais usadas com uma escala de dó até dó.

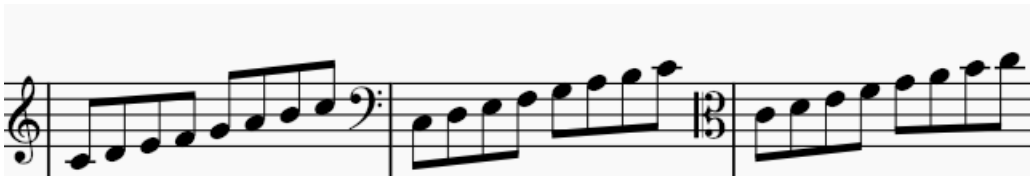

Figura 3. Escala de dó em diferentes claves

Através da imagem percebe-se que dependendo da clave usada, é alterada a posição da nota musical, portanto é importante levar em consideração a clave usada no momento.

2.5 – Notas

O nome das notas musicais são Dó Ré Mi Fá Sol Lá Si, porém é importante considerar que essas mesmas notas podem ser referidas como C D E F G A B em outros países.

2.6 – Tempo

Em uma partitura o tempo é indicado normalmente por um termo (Lento, Adagio, Moderato, Presto etc.), indicado em italiano, em trabalhos antigos ou por um número nas partituras modernas, esse número é medido em Bpm ou batidas por minuto, para utilizar esse valor o músico deve configura-lo em um mecanismo criado em 1812 chamado metrônomo, que tem como propósito medir o tempo de um compasso. Seu funcionamento está relacionado a física, para a indicação do tempo é deslizado um peso em uma haste metálica, na base desta haste há um eixo e um peso de maior massa formando uma alavanca, quanto mais próximos os pesos, mais rápido a haste vai se movimentar. Para um movimento contínuo é armazenada energia elástica no tambor da mola e a cada movimento da haste é liberado um pouco desta energia. Atualmente o metrônomo tem versões digitais, porém sua funcionalidade permanece a mesma. Na figura 4 encontra-se a imagem de um metrônomo mecânico.

Figura 4. Metrônomo

2.7 – Musescore

O Musescore é um programa de composição, nele é possível selecionar um ou mais instrumentos e compor uma música para estes usando as regras de escrita de partitura, o software também é capaz de reproduzir o som da composição seguindo fielmente os

efeitos e mudanças de tempo etc (Werner Schweer, 2002). A Figura 5 apresenta uma partitura no Musescore.

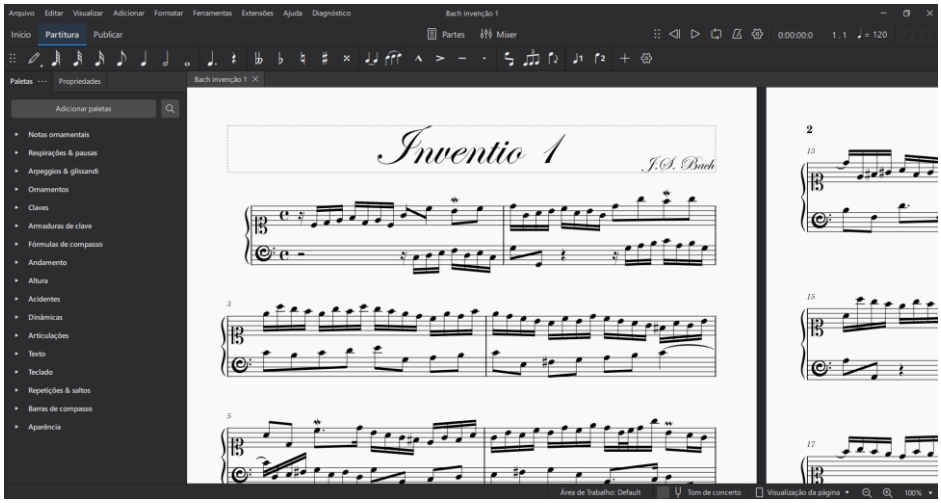

Figura 5. Interface do Musescore

2.8 – Klang.Io

Klang.io é uma plataforma online que permite, através do uso de inteligência artificial, a transcrição de áudio (em diversos formatos, inclusive através de links para youtube) em partitura. Sua versão gratuita permite conversões de apenas 20 segundos (Sebastian Murgul, 2021). A Figura 6 mostra a interface do Klang.io

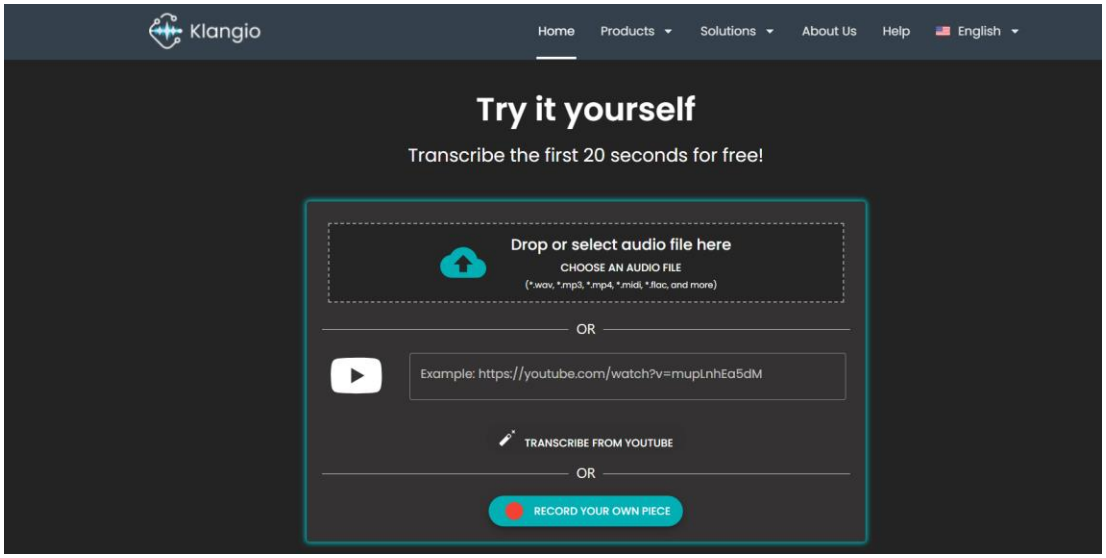

Figura 6. Site Klang.io

2.9 – MidiSheetMusic

MidiSheetMusic é um aplicativo para celulares que permite a conversão de arquivos .midi em partituras, ele vem com algumas partituras junto do sistema mas, várias incompletas. Na Figura 7 é vista a tela inicial do MidiSheetMusic.

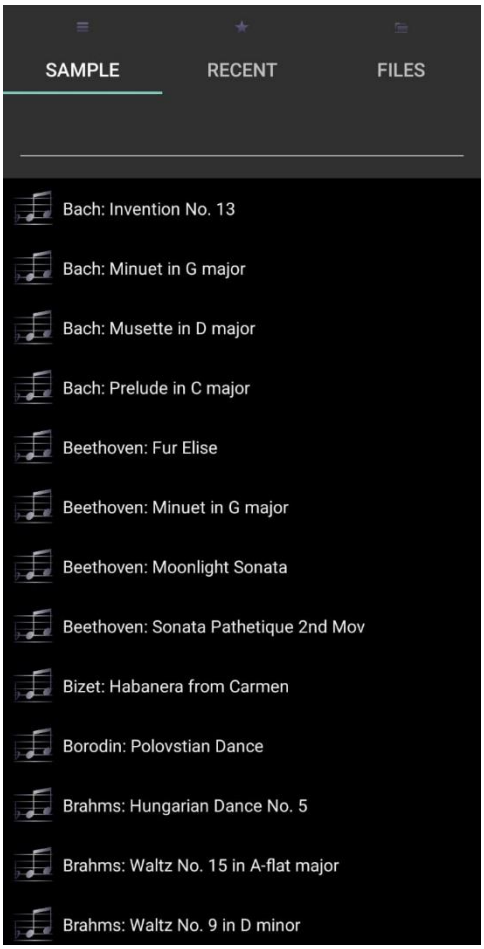

Figura 7. Tela inicial do MidiSheetMusic

2.10 – Python

Python é uma linguagem de programação de script, pode ser usada como orientada a objetos, mas para a tarefa proposta por esse TCC será utilizada com o paradigma procedural. Possui ampla variedade de bibliotecas das quais serão usadas wave, pyaudio, matplotlib, numpy, cv2 (Van Rossum e Drake, 2011; Hubert, 2025; Hunter, 2007; Harris et al., 2020; Bradski e Kaehler, 2008).

2.11 – wave

Essa biblioteca permite gravar um áudio capturado em extensão .wav, caso não seja possível fazer a análise e conversão do som para partitura em tempo real o áudio será gravado utilizando essa biblioteca. (Dependendo do desempenho e outras circunstâncias pode ser trocada por outra biblioteca.) (Python Software Foundation, 1994)

2.12 – pyaudio

Através do pyaudio é possível realizar a captura de som ambiente utilizando o microfone do computador, seu proposito pode variar conforme o desenvolvimento do software avance. (Python Software Foundation, 2006)

2.13 – matplotlib

Utilizando a matplotlib é possível fazer o desenho de gráficos, sua aplicação principal está na análise de gráficos de frequência para ajudar a compreensão do que o software está capturando. (Hunter, 2007)

2.14 – numpy

É uma biblioteca focada em operações matemáticas (Van Der Walt, Colbert, & Varoquaux, 2011)

2.15 – Cv2

A utilização do cv2 (OpenCV) será com intuito de manipular imagens a fim de desenhar a partitura correspondente ao áudio captado. Originalmente desenvolvido pela Intel, o OpenCV é uma biblioteca open-source amplamente usada para computação gráfica em tempo real (Open Source Computer Vision Library, 2000).

3.Trabalhos Relacionados

Como trabalhos relacionados pode-se citar “Protótipo de um conversor MusicXML para MIDI usando Android” (Kerber, 2012), onde a pesquisadora propõe a conversão de MusicXML, um formato de escrita de partitura, para áudio MIDI. No decorrer do trabalho são entregues dados detalhados sobre como o MIDI funciona e, apesar de não ser o objetivo do software que proponho, o conhecimento de como um arquivo MIDI funciona acaba ajudando na construção de um interpretador de som.

Outro trabalho bastante inspirador é “Ferramenta de Apoio ao Ensino de Instrumento Musical”, onde Melo (2009) visa, através da importação de uma partitura MusicXML, ouvir a execução de um aluno e determinar se este está ou não tocando a peça corretamente, o software então marca na partitura os locais onde existem erros de ritmo ou locais onde a nota musical não foi tocada corretamente. Esse trabalho também apresenta descrições bastante interessantes sobre FFT e DFT (Fast Fourier Transform e Discrete Fourier Transform), operações que permitem a leitura do sinal de áudio do microfone através de um conjunto de especificações.

Em “Automatic Music Transcription: Breaking the Glass Ceiling”, Benetos et al (2013) descrevem os desafios de uma transcrição automática de música, abreviada na obra como AMT. Ao decorrer de sua escrita este fala sobre métodos usados para se aproximar do problema em questão, fala também sobre ideias para contornar o problema temporariamente, como uma transcrição semiautomática, ou seja, com auxílio do usuário. Benetos conclui o artigo com uma visão positiva sobre o assunto, e diz que um dos possíveis métodos para levar a AMT é o aumento de informações dentro do problema.

4.Metodologia

Para o desenvolvimento do software foi utilizada a linguagem de programação Python, sendo que esta é uma linguagem que possui pacotes como pyaudio, matplotlib, wave, que respectivamente podem capturar áudio, exibir graficamente valores obtidos, ler e escrever áudio em extensão .wav.

4.1 Captura de frequência

Nesta proposta, o software captura o áudio da música que está sendo reproduzida e o arquiva em uma extensão própria para áudio (.wav). Depois da captura o programa conduz uma análise das frequências encontradas no áudio e, cruzando os valores de frequência e magnitude com 2 vetores contendo a frequência e o nome da nota, cria um terceiro contendo as notas encontradas no áudio cuja magnitude esteja acima de um ponto de corte com proposito de eliminar o som ambiente. Após esse processo inicial, o programa conduz uma segunda análise, onde baseado na variação da magnitude das notas arquivadas, este decide o tempo de cada nota, reiniciando a contagem quando a uma magnitude maior é apresentada para uma nota repetida sucessiva.

Após a análise o software percorre os dados armazenados e classifica as notas encontradas por segmento de áudio em dois valores, representação de posição na partitura (determinado pela clave utilizada no momento) e duração, sendo o último um arredondamento do tempo capturado por segmento.

Utilizando a biblioteca numpy é possível aplicar um processo chamado transformada de Fourier, com isso é possível descobrir qual a frequência capturada pelo microfone e armazená-la em um vetor para ser analisada depois.

O processo de descobrir qual a nota musical está sendo tocada no momento foi feita através da leitura de sua frequência, pois cada nota musical possui uma frequência especifica que define seu nome e sua altura, como exemplos: Lá 4 = 440Hz, Lá 5 = 880Hz, Lá 6 = 1760Hz. No exemplo, podemos observar um padrão muito interessante a respeito das frequências musicais. Podemos criar uma fórmula para a definição da frequência de uma nota:

$F(x_p) = F(x_0).2^P$ onde: X – nome da nota, p – Altura da nota, F(Xp) – Frequência da nota X de altura p, F(X0) – Menor frequência da nota X.

Reproduzindo na prática e novamente usando o Lá como modelo:

$$F(A5) = F(A0).2^5 => F(A5) = 27{,}5\,.\,32 => F(A5) = 880Hz$$

Apesar de não usada no programa final, através da fórmula foi possível economizar espaço de memória do programa, já que não foi necessário gravar a frequência de todas as notas possíveis para um instrumento, apenas a frequência inicial das notas base. Para uma economia maior de espaço no programa ao invés de escrever a fórmula inteira, foi utilizada a técnica de movimentação de carro binário para realizar a multiplicação da frequência. Através desse método foi possível economizar 2 bytes por comando. Inicialmente a hipótese era de uma grande economia de espaço no programa, porém, a economia de dados foi baixa, aproximadamente 2000 bytes, e ofereceu um custo

maior no quesito de complexidade de algoritmo. Embora não tendo testado todos os métodos possíveis para redução de tamanho, ainda é possível especular que durante a compilação para linguagem de máquina o tamanho de 2 Kb pudesse aumentar pois apenas um valor seria necessário na memória do computador ao invés de diversos ponteiros usados comummente na criação de “vetores”.

Ainda no tópico de testes, uma das soluções para encontrar a frequência que se posiciona entre duas notas, com objetivo de usá-la como ponto de corte para realizar a definição do nome da nota, foi uma fórmula matemática que desenvolvida pelo autor, através da diferença sucessiva das frequências vizinhas, a fórmula é vista na Figura 8:

$$P_{arr}(n) = \frac{\sum_{i=0}^{n-1} \{Pa\} + 1{,}95}{2}$$

Figura 8. Fórmula para achar meio termo entre frequências: autoria própria

Essa fórmula usa de uma progressão aritmética para encontrar o valor a ser adicionado à frequência da nota analisada para descobrir onde deve ser seu ponto de corte maior e menor, ela foi abandonada após a comprovação de forte oscilação nas frequências capturadas pelo microfone, tornando a captura de notas muito mais frequente. Como solução foi implementado um ponto de corte de 10% para o valor superior e inferior, também foi inserido no código um vetor de memória cujo papel é impedir a entrada de notas repetidas. Isso significa que, agora o programa atribui uma nota como A 440 se esta estiver em um valor entre 437.75 e 442.37, com a primeira fórmula, a atribuição era feita entre 427.65 e 453.08, que é o equivalente à metade entre 440 e suas notas inferiores e superiores. O código correspondente ao corte de 10% é apresentado na Figura 9, já a Figura 10 mostra a respectiva fórmula.

```
for a in range (0,len(freqtotal),1):
    if len(freqtotal[a]) > 0:
        for b in range (1,len(freqtotal[a]),1):
            if freqtotal[a][b][0] >= freqhertz[0] and freqtotal[a][b][0] < freqhertz[1]-((freqhertz[1]-freqhertz[0])/1.1):
                if freqnome[0] not in marca:
                    frorle.append([freqnome[0],freqtotal[a][b][1]])
                    marca.append(freqnome[0])
            for c in range (1,len(freqnome)-1,1):
                if freqtotal[a][b][0] > (freqhertz[c-1]+((freqhertz[c]-freqhertz[c-1])/1.1)) and freqtotal[a][b][0] <= (freqhertz[c+1]-((freqhertz[c+1]-freqhertz[c])/1.1)):
                    if freqnome[c] not in marca:
                        frorle.append([freqnome[c],freqtotal[a][b][1]])
                        marca.append(freqnome[c])
        frorse.append(frorle)
        frorle = []
        marca = []
```

Figura 9. Código para tolerância de 10%

Fv = frequência declarada em Hz
Fa = frequência capturada em Hz
Fn = nome de frequências baseado em Fv

IF Fa[a][b][0] > (Fv[c-1]+((Fv[c]-Fv[c-1])/1.1))
AND
Fa[a][b][0] <= (Fv[c+1]-((Fv[c+1]-Fv[c])/1.1)):
GRAVA: Fn[c]

Figura 10. Fórmula para tolerância 10% resumida

Na figura 9 pode-se perceber o funcionamento da formula nova: dentro de três laços for, é realizada a leitura dos segmentos de áudio gravados, caso possuam conteúdo, depois é analisada individualmente cada frequência contida no segmento, por último é chamado um terceiro loop com o objetivo de percorrer um vetor que contém as frequências em hertz, caso a frequência analisada no momento ficar dentro da tolerância do programa este buscara o nome da nota encontrada em um vetor específico, utilizando o ponto onde foi lida a última frequência como referência, na Figura 11 pode-se observar os 2 vetores fixos do código.

```
freqhertz = [
    55, 58.270466, 61.735416, 65.40638, 69.295647, 73.416199,
    77.781746, 82.406876, 87.307053, 92.498604, 97.998848, 103.82618, 110, 116.540947, 123.470818,
    130.812775, 138.591324, 146.832367, 155.563492, 164.813782, 174.614105, 184.997208, 195.997711,
    207.652344, 220, 233.081848, 246.941635, 261.625519, 277.182648, 293.664734, 311.126984,
    329.627533, 349.228241, 369.994385, 391.995392, 415.304688, 440, 466.163788, 493.883301,
    523.251099, 554.365234, 587.329529, 622.253906, 659.255127, 698.456482, 739.988831, 783.990845,
    830.609375, 880, 932.327576, 987.766602, 1046.502075, 1108.730591, 1174.659058, 1244.507935,
    1318.510254, 1396.912964, 1479.977539, 1567.981812, 1661.21875, 1760, 1864.654785, 1975.533325,
    2093.004395, 2217.460938, 2349.318115, 2489.015625, 2637.020264, 2793.825928, 2959.955078,
    3135.963135, 3322.4375, 3520, 3729.30957, 3951.066895, 4186.008301, 4434.921875, 4698.636719,
    4978.03125, 5274.040039, 5587.651367, 5919.910645, 6271.92627, 6644.875, 7040, 7458.621094,
    7902.131836, 8372.016602, 8869.844727, 9397.270508, 9956.063477, 10548.083008, 11175.301758,
    11839.820312, 12543.855469, 13289.748047, 14080, 14917.242188, 15804.263672, 16744.033203,
    17739.6875, 18794.542969, 19912.125, 21096.166016, 22350.605469, 23679.640625, 25087.710938,
    26579.496094, 28160, 29834.486328, 31608.527344
]

freqnome = [
    'A0', 'A#0', 'B0', 'C0', 'C#0', 'D0', 'D#0', 'E0', 'F0', 'F#0', 'G0', 'G#0',
    'A1', 'A#1', 'B1', 'C1', 'C#1', 'D1', 'D#1', 'E1', 'F1', 'F#1', 'G1', 'G#1',
    'A2', 'A#2', 'B2', 'C2', 'C#2', 'D2', 'D#2', 'E2', 'F2', 'F#2', 'G2', 'G#2',
    'A3', 'A#3', 'B3', 'C3', 'C#3', 'D3', 'D#3', 'E3', 'F3', 'F#3', 'G3', 'G#3',
    'A4', 'A#4', 'B4', 'C4', 'C#4', 'D4', 'D#4', 'E4', 'F4', 'F#4', 'G4', 'G#4',
    'A5', 'A#5', 'B5', 'C5', 'C#5', 'D5', 'D#5', 'E5', 'F5', 'F#5', 'G5', 'G#5',
    'A6', 'A#6', 'B6', 'C6', 'C#6', 'D6', 'D#6', 'E6', 'F6', 'F#6', 'G6', 'G#6',
    'A7', 'A#7', 'B7', 'C7', 'C#7', 'D7', 'D#7', 'E7', 'F7', 'F#7', 'G7', 'G#7',
    'A8', 'A#8', 'B8', 'C8', 'C#8', 'D8', 'D#8', 'E8', 'F8', 'F#8', 'G8', 'G#8']
```

Figura 11. o vetor superior contém as frequências em hertz e inferior seus nomes.

Também é valido mencionar a série harmônica, ela é formada através dos múltiplos do valor da frequência fundamental (X1), em um exemplo prático podemos escrever a ordem harmônica como Lá1 Lá2 Mi3 Lá4 Dó#4 Mi4 Sol4 Lá5, convertendo isso em frequência para ter um entendimento simplificado: 110 220 330 440 550 660 770 880.

Durante testes com o programa foi observado a presença desta série com áudios que deveriam conter apenas uma nota, um fenômeno comum em instrumentos acústicos,

simulado em instrumentos digitais, por conta do ponto de corte de magnitude na identificação de frequência foi possível cortar grande parte dessa interferência da captura de áudio. Não seria possível pedir para o programa remover qualquer nota pertencente a série harmônica pois poderia entrar em conflitos com notas que estivessem na música tendo em vista que é bastante comum o uso de oitavas em várias composições.

Apesar de não previsto inicialmente, outro problema que surgiu na construção e testes do programa é o som emitido pelas teclas do piano (um dos instrumentos usados para o teste), que interferem com o áudio de maneira significativa. Para contornar essa interferência, foi iniciado o estudo de timbres, para criar um filtro capaz de barrar sons indesejados, porém, o estudo de timbres é complexo, demanda mais tempo do que o disponível para este projeto, como está fora de seu escopo, foi desconsiderado.

4.2 Escrita da partitura

Detalhando o processo de posicionamento das notas no programa, essas são atribuídas baseadas em onze valores fixos representando os espaços e linhas do pentagrama. A impressão da imagem da nota também segue regras próprias, onde o arquivo PNG que contêm a nota tem coloração azul 255 de fundo, permitindo que o programa descubra quais partes da imagem pertence a nota e quais devem ser ignoradas. Para que o programa saiba onde a nota deve ser colocada, este recorre a 4 vetores criados com base nas claves de sol (1ª e 2ª linha) e fá (1ª e 4ª linha), esses vetores contém a posição das notas para dada clave, e são usados como comparativo para decidir entre colocar uma nota em uma linha ou espaço.

A definição da clave ocorre 3 vezes ao longo do programa, a primeira vez, o software busca o primeiro conjunto de sons capturados pelo microfone e compara as posições das notas, em questão de altura, com as 4 claves utilizadas, cada vez que uma correspondência é encontrada dentro do segmento, é adicionado 1 ponto para a clave que possui essa correspondência. Ao final do processo define-se a clave com maior pontuação, dando, em caso de empate, prioridade para as claves sol 2ª linha e fá 4ª linha. Depois de atribuir uma clave o programa adiciona uma vantagem de 0.5 pontos para a clave atual, para impedir sua troca caso a diferença seja pouca nas próximas checagens.

A segunda e terceira checagem de clave ocorrem durante a escrita das notas, o software constantemente verifica a altura das notas do segmento onde está trabalhando e caso a altura supere a da clave atual, esta é trocada pela clave que possuir maior pontuação. A última escrita de clave ocorre apenas quando se chega ao fim da folha, a clave é posicionada no início da linha seguinte. O posicionamento gráfico das notas é indicado por um conjunto de vetores que pode ser visto na Figura 12, estes são lidos pelo programa mostrado na Figura 13.

```
###### COBERTURA F1 até B5

s2 = ["G4","F4","E4","D4","C4","B4","A4","G3","F3","E3","D3"]
s1 = ["B5","A5","G4","F4","E4","D4","C4","B4","A3","G3","F3"]
f4 = ["B3","A3","G2","F2","E2","D2","C2","B2","A2","G1","F1"]
f1 = ["A4","G3","F3","E3","D3","C3","B3","A3","G2","F2","E2"]


vetclave = [[47,34],[52,36],[32,16],[41,25]] ##### VETOR PARA CALCULO MUDANÇA
##############4G 3D

# A=1 B=2 C=3 D=4 E=5 F=6 G=7
```

Figura 12. Vetores para uso da clave

Na figura 12 encontram-se na parte superior os 4 vetores (s1, s2, f1 e f4), usados para posicionar as notas em linhas ou espaços de acordo com a clave e o vetor inferior indicando a cobertura de cada clave. Na figura 13 expõe-se o código que realiza a definição da clave, nota-se que este transforma o nome da nota musical em números, através do comando *replace* para compará-la com o vetor 'vetclave', visto na parte inferior da Figura 12.

```
################### INICIO DEFINIR A CLAVE

for j in range (0,len(frorse[i]),1):
    notatemporaria = frorse[i][j][0]
    nt = notatemporaria.replace("#","")
    V1,V2 = nt[0],nt[1]
    V1 = V1.replace("A","1")
    V1 = V1.replace("B","2")
    V1 = V1.replace("C","3")
    V1 = V1.replace("D","4")
    V1 = V1.replace("E","5")
    V1 = V1.replace("F","6")
    V1 = V1.replace("G","7")
    numero = int(str(V2)+str(V1))
    if numero >= vetclave[0][1] and numero <= vetclave[0][0]:
        ps2 = ps2 + 1
    if numero >= vetclave[1][1] and numero <= vetclave[1][0]:
        ps1 = ps1 + 1
    if numero >= vetclave[2][1] and numero <= vetclave[2][0]:
        pf4 = pf4 + 1
    if numero >= vetclave[3][1] and numero <= vetclave[3][0]:
        pf1 = pf1 + 1

if ps1 > ps2 and ps1 > pf4 and ps1 > pf1:
    clv = "s1"
    pori = 8
    imclave = sol.copy()
    ananota = s1.copy()
    Pclav = 3
if pf1 > ps2 and pf1 > pf4 and pf1 > ps1:
    clv = "f1"
    pori = 25
    imclave = fa.copy()
    ananota = f1.copy()
    Pclav = 2
```

Figura 13. Trecho do código que define a clave antes da escrita

5.Análise de resultados

A validação do programa foi feita em um comparativo de legibilidade entre um arquivo convertido de WAV para MIDI e os resultados do programa, sem novidade ambos erraram na escrita da partitura, a diferença é que o programa foi capaz de remover a maior parte dos ruídos ambientes e entregar uma partitura mais limpa e semelhante a peça original, em alguns casos ambas as transcrições foram exibidas de maneira ilegível.

Para realizar os comparativos, serão colocadas imagens dos resultados do programa e de uma conversão wav para midi. Será explicado o primeiro exemplo de um trecho gravado pelo computador, seguirão em ordem imagens de como a partitura deveria se parecer (Figura 14), da partitura gerada pelo software proposto (Figura 15) e, por último, a partitura gerada através de uma conversão para .midi (Figura 16).

Na figura 14 pode ser encontrado um exemplo de parte da 9ª sinfonia de Beethoven, conhecida como Ode a alegria (versão simplificada).

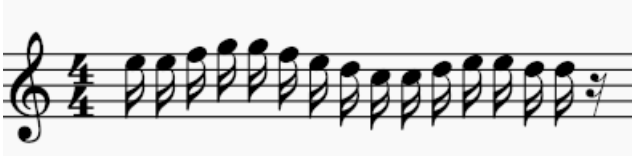

Figura 14. Resultado esperado

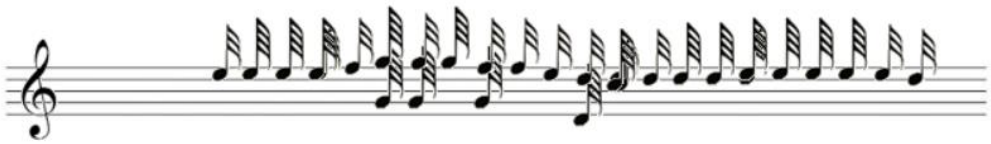

Figura 15. software proposto

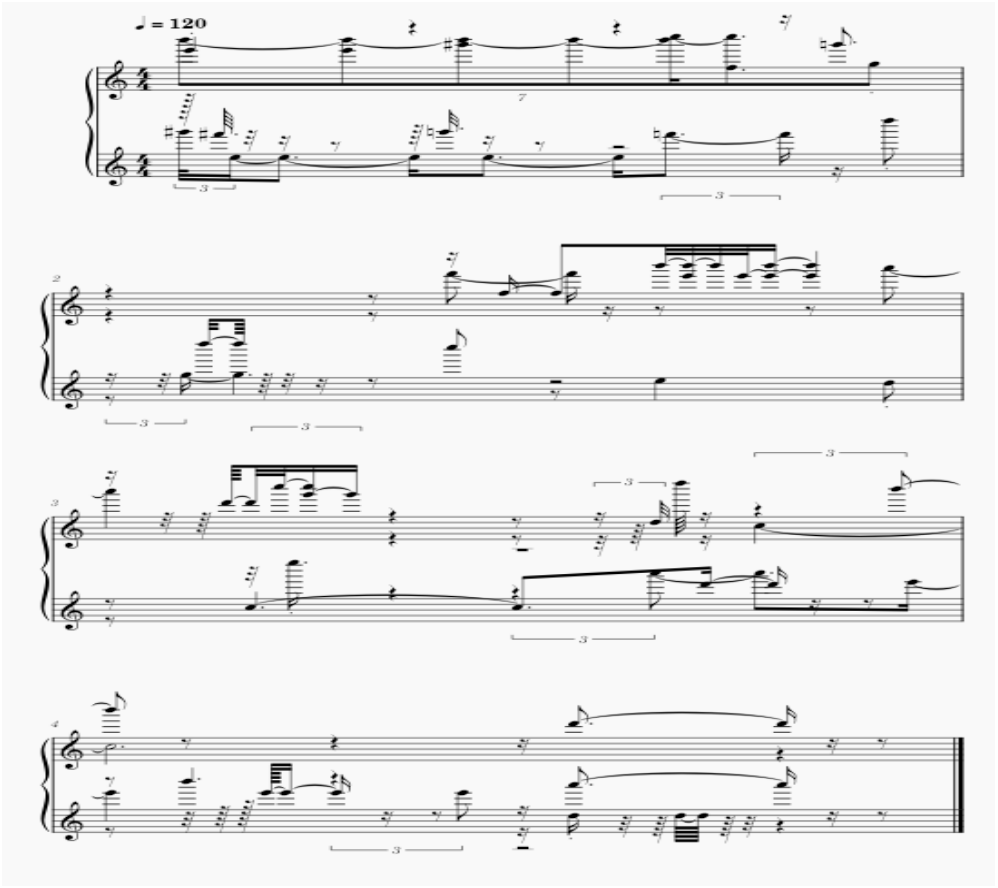

Figura 16. Conversão WAV para MIDI

Podemos perceber em um comparativo da Figura 14 com a 15 que estas tem uma semelhança muito grande, com alguns erros presentes na figura 15 cuja ocorrência se dá por fatores já discutidos e, uma oscilação inesperada por parte da magnitude, que resultou em notas repetidas. Em alguns pontos da partitura produzida pelo software também é possível notar a invasão de notas da série harmônica, isso ocorre na nota sol e permanece ainda junto de uma nota fá, a probabilidade maior é pela oscilação da frequência e magnitude, já comentado antes.

Agora, discutindo os erros da Figura 16, devido a conversão do arquivo WAV em MIDI, o conversor recebeu todas as frequências que o software deste projeto remove através do corte da magnitude, tornando a partitura muito mais poluída e difícil de tocar do que realmente é, claro que a partitura é muito mais "bonita" pois está seguindo as regras de escrita de partitura fielmente, porém sua representação do que deve ser tocado é muito inferior ao que o software foi capaz de escrever. Abaixo encontram-se mais algumas imagens de conversões que o software proposto fez para partitura, todas as imagens foram geradas com um limite de 10 segundos para as gravações, permitindo gerar as partituras mais rapidamente.

Ainda, forma feitos teste comparando a transcrição sonora de um instrumento de sopro (ocarina), presente na Figura 17, onde resultado gerado pelo software está à esquerda.

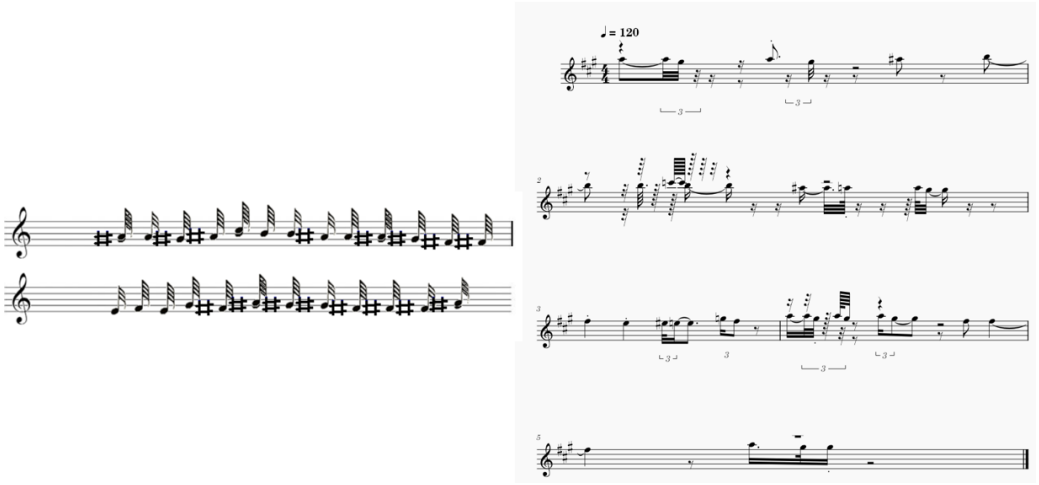

Figura 17. Comparando com um instrumento de sopro (ocarina).

Na Figura 18 mostra p resultado gerado pelo software proposto, onde é possível visualizar um sopro de 3 notas na ocarina e perceber a troca de clave para suprimir linhas suplementares na nota dó#.

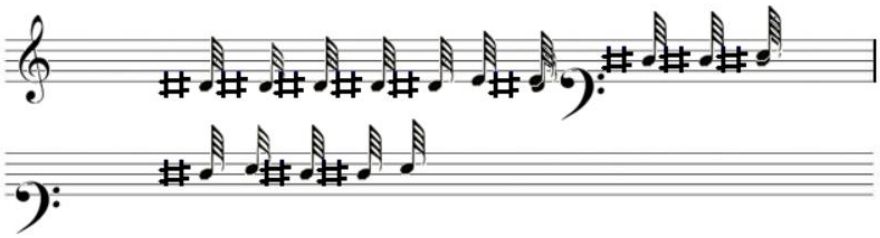

Figura 18. Um sopro de 3 notas na ocarina.

Por sua vez, a Figura 19 demonstra um sopro de 3 notas na ocarina, com a conversão wav para midi no Musescore, o qual utiliza na armadura de clave bemóis (b), portanto onde se lê Mib, no software se lê ré#.

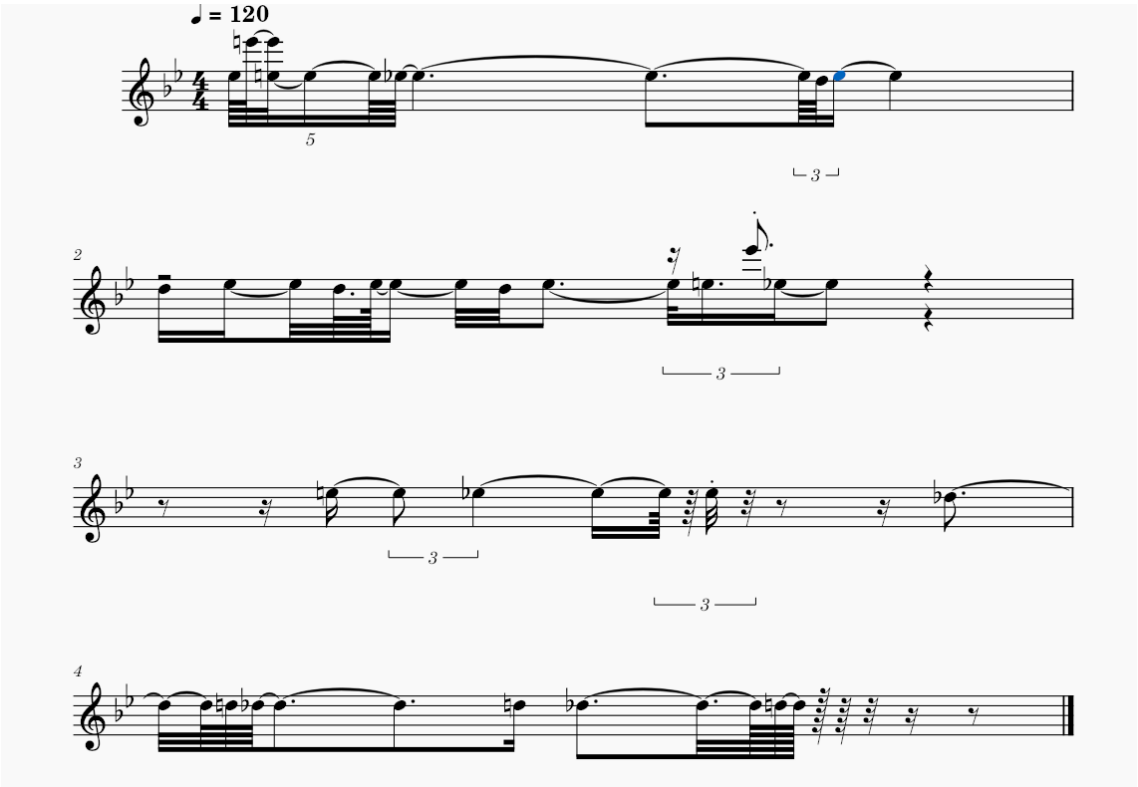

Figura 19. Apenas um sopro de 3 notas na ocarina (conversão wav para midi).

Já a Figura 20 apresenta a conversão um sopro de 3 notas na ocarina realizada pelo software proposto, em comparação com a Figura 21, conversão realizada pelo Musescore. Visualmente, é possível perceber que em as figuras se assemelham.

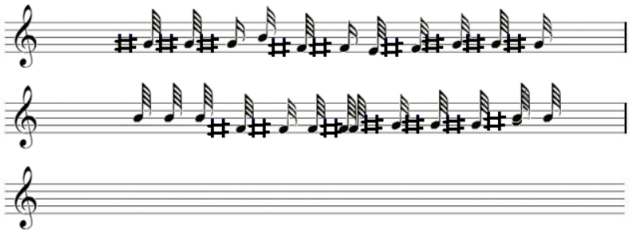

Figura 20. Programa proposto

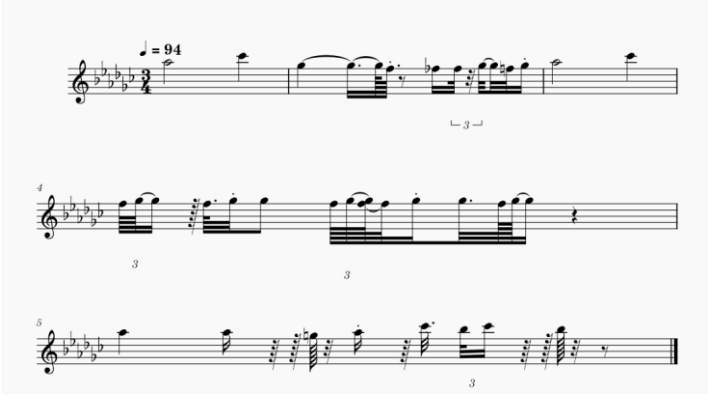

Figura 21. Conversão Musescore.

6.Conclusão

Pode-se perceber com esse trabalho que, a conversão de áudio em partitura é uma tarefa complexa, onde vários fatores podem influenciar o resultado final, podendo ser citados como principais: som ambiente, série harmônica, som das teclas do piano, som do ar soprado em uma ocarina ou flauta. A provável solução para o som ambiente, das teclas e sopro, seria uma análise aprofundada de timbres, porém ao chegar nesse ponto deve-se levar em consideração que cada instrumento tem um timbre diferente e, uma configuração para um instrumento fara com que outro possa ser excluído.

O programa pode ser usado, apesar de suas limitações (sem armadura de compasso ou tempo), para uma noção básica do formato da música, em conjunto com o áudio pode ser suficiente para descobrir as notas de uma determinada peça com mais velocidade do que apenas ouvindo.

6.Referências

PYTHON SOFTWARE FOUNDATION. *PyAudio*. Disponível em: https://people.csail.mit.edu/hubert/pyaudio/. Acesso em: 30 ago. 2024.

VAN DER WALT, S.; COLBERT, S. C.; VAROQUAUX, G. The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, v. 13, n. 2, p. 22-30, 2011. Disponível em: https://doi.org/10.1109/MCSE.2011.37. Acesso em: 30 ago. 2024.

PYTHON SOFTWARE FOUNDATION. wave — Interface para arquivos de áudio WAV. Disponível em: https://docs.python.org/3/library/wave.html. Acesso em: 30 ago. 2024.

PYPI. opencv-python. Disponível em: https://pypi.org/project/opencv-python/. Acesso em: 30 ago. 2024.

OPEN SOURCE COMPUTER VISION LIBRARY. *OpenCV*. Disponível em: https://opencv.org/. Acesso em: 30 ago. 2024.

WIKIPÉDIA. Metrônomo. Disponível em: https://pt.wikipedia.org/wiki/Metr%C3%B4nomo#:~:text=O%20metr%C3%B4nomo%20ou%20metr%C3%B3nomo%20%C3%A9,de%20estudo%20ou%20interpreta%C3%A7%C3%A3o%20musical. Acesso em: 30 ago. 2024.

WIKIPEDIA. Metronome. Disponível em: https://en.wikipedia.org/wiki/Metronome. Acesso em: 30 ago. 2024.

SCHWEER, Werner. *MuseScore*. 2002. Disponível em: https://musescore.org. Acesso em: 30 ago. 2024.

HUNTER, John D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, v. 9, n. 3, p. 90-95, 2007. Disponível em: https://doi.org/10.1109/MCSE.2007.55. Acesso em: 30 ago. 2024.

KERBER, Ana Lívia. Protótipo de um conversor MusicXML para MIDI usando Android. 2012. Trabalho de Graduação – Universidade Federal do Rio Grande do Sul, Porto Alegre, 2012. Orientador: Prof. Dr. Marcelo Soares Pimenta.

MELO, Fábio Delicato Feijó de. *Ferramenta de apoio ao ensino de instrumento musical*. 2009. Trabalho de Conclusão de Curso (Bacharel em Engenharia da Computação) – Escola Politécnica de Pernambuco, Universidade de Pernambuco, Recife, 2009. Orientador: Prof. Carmelo Jose Albanez Bastos Filho.

Benetos, E., Dixon, S., Giannoulis, D., Kirchhoff, H., & Klapuri, A. (2013). Automatic Music Transcription: Breaking the Glass Ceiling. PLoS ONE, 8(5), e61867.

MURGUL, Sebastian; LÜNGEN, Alexander. Klang.io. Disponível em: https://klang.io/about-us/?utm_source=chatgpt.com. Acesso em: 17 fev. 2025.

Klapuri, A., & Davy, M. (Eds.). (2006). Signal Processing Methods for Music Transcription. Springer.

Klapuri, A. (2004). Automatic music transcription as we know it today. Journal of New Music Research, 33(3), 269–282.

WIKIPEDIA. Partitura. Disponível em: https://pt.wikipedia.org/wiki/Partitura. Acesso em: 18 fev. 2025.