Sistema de Reconhecimento de Escalas Musicais Utilizando Transformação Rápida de Fourier e Arduino

Guilherme de Nez Silvano[1], Giácomo Althoff Bolan[1]

[1]Ciência da Computação – Universidade do Extremo Sul Catarinense (UNESC)
Av. Universitária, 1105 - CEP: 88806-000 – Criciúma – SC – Brasil

guisilvano@hotmail.com, kinhobolan@live.com

Abstract. *Music is a common form of artistic manifestation, be it as a hobby or professional activity, music is present in the everyday life of a large number of people. Seeking to grow the number of available uses of technology in music, this research has the goal of using the fast Fourier transform algorithm in the development of a embeded system capable of identifying musical scales originating from an electrical instrument. This application can read and musically classify frequencies between 134 Hz and 1040 Hz.*

Resumo. *A música é uma manifestação artística comum, seja como hobby ou como atividade profissional, ela é uma constante no dia a dia de um grande número de pessoas. Visando ampliar a disponibilidade de opções do uso da tecnologia na música, esta pesquisa tem como objetivo aplicar o algoritmo da transformação rápida de Fourier no desenvolvimento de um sistema capaz de identificar escalas músicas oriundas de um instrumento elétrico. A aplicação pode ler e classificar musicalmente frequências entre 134 Hz e 1040 Hz.*

1. Introdução

Estamos cotidianamente envolvidos com as Tecnologias de Informação e Comunicação, em todas as áreas da vida e, particularmente no campo da arte, algumas propostas metodológicas são implementadas a fim de incentivar o uso das tecnologias como um valioso complemento à atividade artística. Porém, estas aplicações ainda são generalizadas e são observadas deficiências quanto à disponibilidade de opções para a sua união com a música (OLIVEIRA; COELHO, 2020).

Para o desenvolvimento de uma ferramenta digital capaz de interagir com o músico enquanto ele toca é necessário que os sinais analógicos emitidos por seu instrumento sejam convertidos para sinais digitais, permitindo a comunicação entre o músico e a máquina. A determinação da frequência fundamental de um sinal é uma das questões mais prevalentes no campo do processamento de sinais (SKJEI, 2011, tradução nossa). Segundo Jarne (2017, tradução nossa), algumas das mais notáveis utilizações algoritmos de detecção de frequência fundamental são aplicações de interpretação de sons emitidos por animais e no processamento e análise de voz para humanos, assim como em estudos de análise da estabilidade de linhas de eletrificação urbana e análise musical.

Diversas aplicações musicais necessitam a estimação da frequência fundamental para seu funcionamento, como sistemas de interatividade em tempo real ou transcrição de notas musicais. A frequência fundamental de uma onda periódica ideal é o inverso do seu período. Porém, em casos como na fala ou na música, as ondas sonoras emitidas não são ideais, portanto, o cálculo de suas frequências fundamentais é mais complexo

(DE CHEVEIGNE; KAWAHARA, 2002, tradução nossa). Existem dois algoritmos amplamente utilizados quando busca-se a frequência fundamental de uma onda: a Transformação Rápida de Fourier (*Fast Fourier Transform*, ou *FFT*) e o algoritmo de Autocorrelação.

Segundo Martins et al. (2016) O FFT permite extrair a frequência fundamental de uma onda sem a influência de ruídos externos através de cálculos complexos que permitem uma mudança de domínio: a onda deixa de ser representada a partir do seu tempo e passa a ser representada por sua frequência. Enquanto o algoritmo de autocorrelação é capaz de evidenciar a frequência fundamental de uma onda sonora em meio ao ruído através da comparação dessa onda com outros segmentos dela mesma, a fim de evidenciar a periodicidade da onda por meio de picos mais evidentes (TAN; KARNJANADECHA, 2003).

Durante o levantamento bibliográfico das etapas de proposta e projeto de pesquisa deste trabalho, foi dado foco à utilização do algoritmo de autocorrelação como ferramenta ideal para a detecção da frequência do sinal de um instrumento elétrico, porém durante o início da fase de desenvolvimento do sistema, ainda realizando pesquisas bibliográficas, notou-se que o FFT seria uma ferramenta mais adequada ao projeto. Segundo Apicella et al (2013, tradução nossa), em geral, a análise via transformação rápida de Fourier é mais eficaz quando existem picos mais densos e bem definidos, enquanto o método de autocorrelação é mais preciso na definição do período de ondas com menos picos e de resolução maior. Um instrumento musical gera uma onda senoidal periódica, favorecendo o uso da transformação rápida de Fourier por possuir picos evidentes.

Para permitir o cálculo e interpretação da frequência dos diferentes sinais gerados por um instrumento musical, foi necessária a criação de uma interface entre o instrumento e o Arduino. Esta interface foi desenvolvida utilizando um circuito amplificador não-inversor e um Arduino Due, que possui conversores analógico-digitais integrados.

Segundo o levantamento bibliográfico não foram encontrados trabalhos que utilizam FFT ou autocorrelação com a finalidade específica de encontrar escalas musicais, porém há diferentes trabalhos que utilizam estas técnicas para a identificação de frequências fundamentais. O estudo publicado por Nessen e Needel (2013) descreve métodos para a utilização da transformação rápida de Fourier em um projeto para a identificação de acordes em músicas. O projeto descreve as funções matemáticas e as relações musicais entre diferentes notas para fundamentar um modelo de identificação de acordes. Jarne (2019), desenvolveu um projeto de código aberto utilizando a linguagem Python com a finalidade de identificar as frequências fundamentais no canto de canários. O código desenvolvido utilizou a função specgram da biblioteca de código aberto matplotlib. A autora conclui confirmando o êxito na implementação do código e ressalta que embora seu trabalho tenha sido desenvolvido para a identificação de canto de pássaros, o mesmo pode ser utilizado para a análise de sons de diferentes origens. Alain de Cheveigné e Hideki Kawahara (2002) desenvolveram o algoritmo de autocorrelação Yin com o objetivo de identificar frequências fundamentais tanto na música quanto na fala. Os autores declaram que um algoritmo de autocorrelação, quando utilizado por si só, gera erros demais para determinadas aplicações. Após a implementação de diferentes métodos para o tratamento de erros, os autores

compararam o novo algoritmo às ferramentas de abstração de frequência fundamental existentes e comprovam que o Yin é um método mais eficiente e possui menor latência do que as alternativas da época.

Durante as pesquisas para este trabalho, não foram encontrados produtos ou algoritmos com uma finalidade similar ao projeto proposto. Portanto, buscando expandir a utilização dos algoritmos de determinação de frequência fundamental, esta pesquisa tem como objetivo criar um protótipo de sistema embarcado utilizando Arduino para a implementação de um identificador de escalas musicais em tempo real baseando-se na utilização da transformação rápida de Fourier para a definição das notas musicais oriundas de um instrumento elétrico.

Os objetivos específicos deste trabalho consistem em: explorar a utilização de algoritmos de determinação de frequência fundamental de onda; desenvolver um algoritmo de reconhecimento de notas musicais utilizando a transformação rápida de Fourier; desenvolver um algoritmo de identificação de escalas musicais baseado na distância entre as notas tocadas em tempo real pelo usuário; desenvolver uma interface para instrumentos elétricos utilizando Arduino; combinar o software desenvolvido ao protótipo de hardware.

2. Materiais e Métodos

Esta é uma pesquisa aplicada, de base tecnológica e explicativa. No início da fase de desenvolvimento construiu-se um protótipo de interface entre um instrumento musical elétrico e um Arduino (Figura 1). O hardware conta com dois circuitos distintos além do Arduino: o circuito amplificador e o display que retorna ao usuário as notas e a escala musical que ele tocou em seu instrumento.

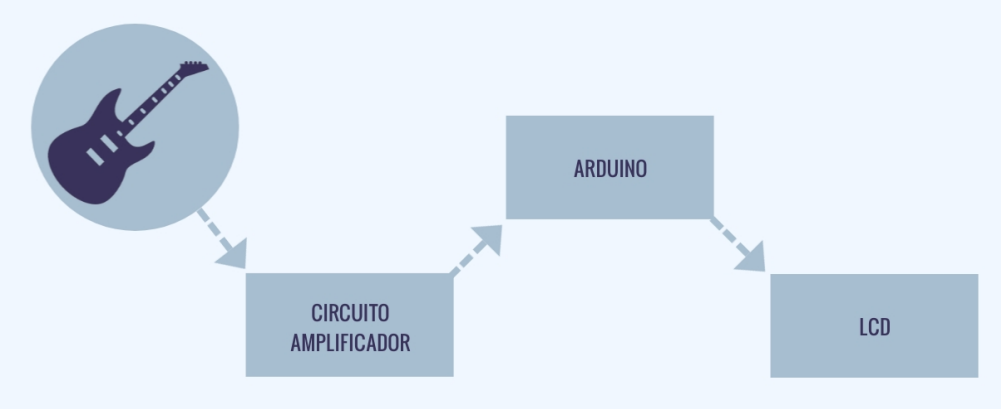

Figura 1. Modelo do hardware

2.1. Construção do Hardware

Para a primeira etapa da construção do circuito de interface foi desenvolvido o circuito de entrada e amplificação do sinal do instrumento elétrico. A entrada do circuito foi desenvolvida para instrumentos que utilizam cabos do padrão P10, como violões, baixos e guitarras. A saída dos dados para a visualização pelo usuário é feita por um LCD de 16x2 caracteres.

2.1.1. Amplificador

O amplificador tem o papel de garantir o ganho e amplitude necessárias para que o sinal de entrada seja coerente com o esperado pelo conversor (BRYANT; HENDRIKS; KESTER, 2005, tradução nossa).

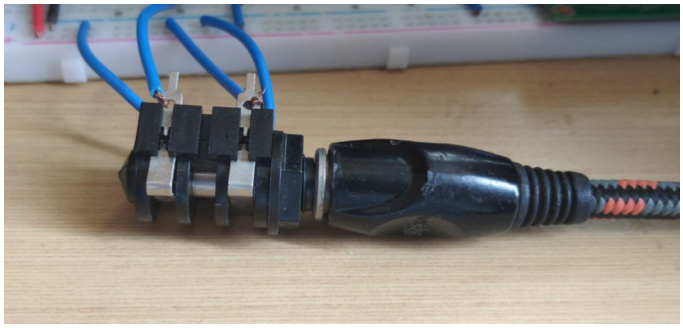

Figura 2. Conexão Entre o Instrumento e o Hardware

O instrumento é ligado a um *jack* de áudio por um cabo comum (Figura 2), e tem sua saída direcionada a um amplificador não-inversor descrito na figura 3. O amplificador operacional LM-082 é alimentado pelo trilho de 3.3 Volts e aterrado ao pino GND, ambos do próprio Arduino.

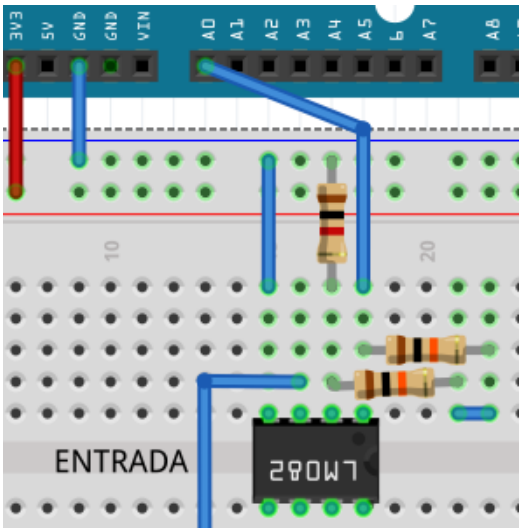

Figura 3. Circuito Amplificador

Para calcular o ganho do sinal de saída em relação ao de entrada, segundo Pertence Jr. (2003), emprega-se a fórmula $Ganho = 1 + \frac{R\,saída}{R\,entrada}$. Sendo as resistências de entrada e saída 10k Ohms e 20k Ohms, respectivamente, o amplificador desenvolvido possui um ganho de valor três, ou seja, todo o sinal que ele recebe como entrada terá o valor de sua amplitude multiplicada por três. A saída do amplificador é ligada ao pino A0, que é capaz de ler sinais analógicos e convertê-los em valores de até 12-bits.

2.1.2. Display

O circuito do display de saída do protótipo conta com um LCD de 16x2 caracteres conectado ao Arduino por uma *backpack* (mochila), uma placa que permite a comunicação do Arduino com o display pelo protocolo I2C. Todos os pinos do LCD são ligados a esta placa, que por sua vez é ligada aos pinos 20 (SDA, ou *serial data line*) e 21 (SCL, ou *serial clock line*) do Arduino.

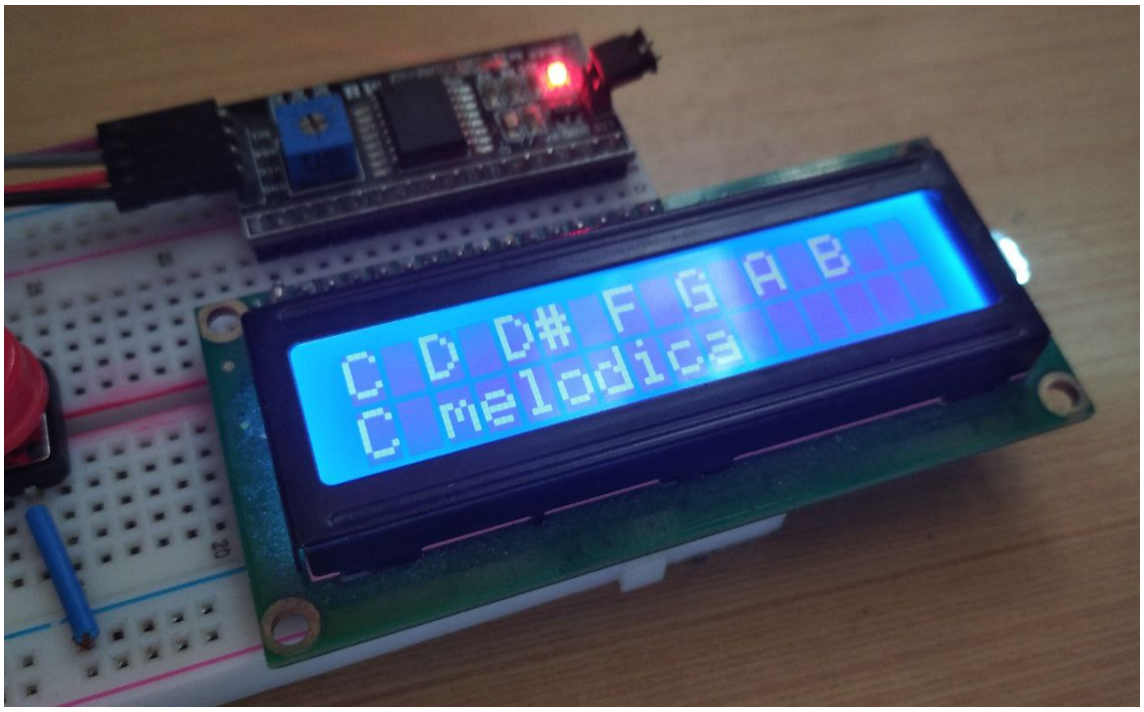

Figura 4. Display Após o Usuário Tocar a Escala Dó Melódica

Este tipo de ligação permite a serialização das instruções enviadas do Arduino até o LCD. Para isto foi utilizada a biblioteca de código aberto *LiquidCrystal_I2C.h*, desenvolvida por John Rickmandis e disponível gratuitamente no GitHub e GitLab, que utiliza a biblioteca padrão *Wire.h* à fim de permitir uma implementação mais simples do código responsável pelo gerenciamento do display.

2.2. Desenvolvimento do Software

O software foi desenvolvido utilizando a linguagem de programação *C* e projeto pode ser dividido em duas partes distintas: o processamento do sinal de entrada e a interpretação musical das frequências fundamentais obtidas.

2.2.1 Processamento do Sinal

O primeiro passo para o desenvolvimento do software foi implementar a transformação rápida de Fourier para obter a frequência fundamental do sinal de entrada gerado pelo hardware. O desenvolvimento deste algoritmo foi realizado utilizando a biblioteca de código livre *arduinoFFT.h*, criada e mantida por Enrique Condes, disponível no GitHub. Esta biblioteca foi desenvolvida utilizando a linguagem *C++*.

O sinal analógico de entrada é lido no pino A0 do Arduino e convertido para um valor digital de 10-bits pela função padrão *analogRead*, este sinal é por sua vez inserido em um vetor *vReal* de 256 elementos que são lidos à uma taxa de 2 kHz.

Após sua leitura, os elementos deste vetor são adicionados a uma classe nomeada *FFT*, do tipo *arduinoFFT* (Figura 5), implementado pela biblioteca. Este objeto recebe um ponteiro para o vetor *vReal*, um ponteiro para o vetor *vImg* (que durante a leitura do sinal foi preenchido com zeros), o número de elementos dos vetores (*samples*) e sua taxa de leitura (*samplingFrequency*).

```
arduinoFFT::arduinoFFT(double *vReal, double *vImag, uint16_t samples, double samplingFrequency)
{// Constructor
    this->_vReal = vReal;
    this->_vImag = vImag;
    this->_samples = samples;
    this->_samplingFrequency = samplingFrequency;
    this->_power = Exponent(samples);
}
```

Figura 5. Classe *arduinoFFT*

A construção desta classe é realizada pela função *FFT.Compute*, responsável também pela realização da transformação rápida de Fourier. Esta função recebe a onda armazenada no buffer de entrada, composta pelo sinal elétrico gerado pelo instrumento, e retorna uma outra onda que representa a frequência fundamental da onda do buffer.

A função computa os valores do buffer de entrada seguindo a equação $x_k = \frac{1}{N}\sum_{k=0}^{N-1} x_n W_N^{uk}$, sendo que $W_N^{uk} = \varrho^{j2\pi \cdot x/N}$, onde N é o número de amostras que serão calculadas, x é o elemento atual que está sendo processado, x_n é o valor do elemento x no tempo atual, k é a taxa de leitura. Para que a função funcione como esperado, presume-se que N seja um fator de *2*.

Esta equação retorna um vetor de números complexos, porém, como não há um tipo de variável padrão disponibilizada pelas linguagens *C/C++* para o armazenamento e manipulação de números complexos, o resultado desta equação é abstraído e dividido em dois vetores: um armazena os valores reais *vReal* e o outro os componentes imaginários *vImg*. Este método de armazenamento de números complexos também pode ser encontrado na biblioteca padrão *complex.h*.

Após a função *FFT.Compute*, é chamada a função *FFT.ComplexToMagnitude* responsável por transformar os números complexos obtidos na função anterior em números reais que representam a amplitude do sinal gerado em um ponto *i*. Esta função age conforme a equação $\sum_{i=0}^{amostras} \sqrt{\sqrt{vReal} + \sqrt{vImg}}$ e insere cada resultado no vetor *vReal* da classe *arduinoFFT*.

Em seguida, é chamada a função *FFT.MajorPeak*, que determina a frequência da onda obtida na rotina anterior através da detecção dos picos da onda gerada. A função retorna um valor do tipo *double* calculado pela equação $freq = \frac{Ypico \times taxa\ de\ amostras}{número\ de\ amostras}$, este é o valor em Hz da frequência fundamental do sinal de entrada.

As rotinas acima são executadas novamente toda vez que o programa inicia um novo ciclo em seu *loop* principal.

2.2.2 Interpretação Musical das Frequências

A cada ciclo do programa, é chamada uma função *get_nota* responsável pela conversão dos resultados da transformação rápida de Fourier em notas musicais utilizando a notação por letras. A função recebe a frequência fundamental do sinal de entrada e retorna um valor de 16-bits, ou dois *bytes*.

```
// GET_OITAVA 4
else if (val < 505.6) {
    if (val < 269.4) {
        return (('C' << 8) | (4 << 4) | 0);
    }
    else if (val < 285.42) {
        return ((('C' << 8) | (4 << 4)) | 1);
    }
    else if (val < 301.39) {
```

Figura 6. Retorno da função *get_nota* para as notas *dó* (C) e *dó sustenido* (C#) na 4ª oitava musical

O primeiro byte contém um *char* que representa a notação musical daquela frequência (C, D, E…), e o segundo byte é dividido em duas palavras: a primeira possui 7-bits e denomina a oitava em que esta nota musical está inserida, já a segunda palavra tem apenas um bit e determina se a nota é sustenida ou não. O retorno desta função é armazenado em um *buffer* de sete elementos denominado *buffer_notas* conforme a figura 7:

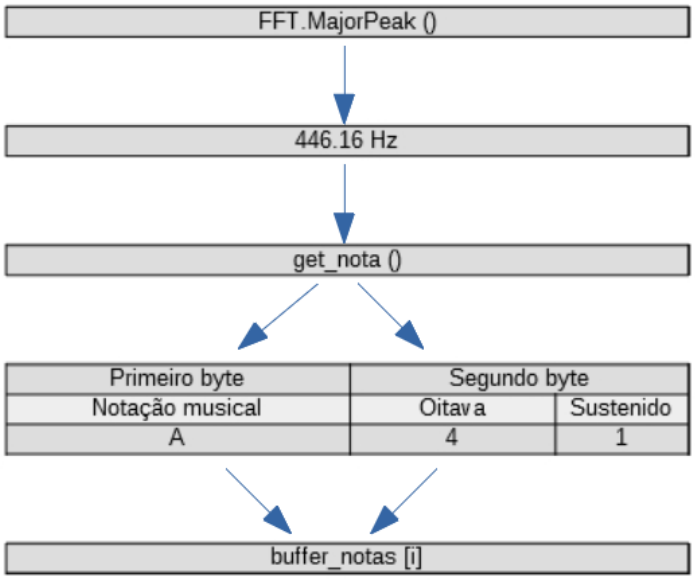

Figura 7. Fluxo da Etapa de Reconhecimento de Notas

Para que este buffer não possua elementos iguais, chama-se a rotina *get_inseridas*, que compara o primeiro byte e o último bit à esquerda de cada elemento com o elemento a ser inserido, ignorando a oitava em que a nota se encontra. Logo após a função acima, é chamada a rotina *ordena_buffer*, para que os elementos de

buffer_notas sejam colocados em ordem crescente de frequência utilizando o algoritmo *bubble sort*.

2.2.3 Relacionando as Notas Musicais

Existem doze notas distintas na música ocidental e a distância entre elas é chamada de *intervalo*, o sistema classifica as notas musicais da entrada utilizando a *notação por letras* e também por sua distância em relação a nota *dó*.

Nota	Símbolo	Intervalo
Dó	C	0
Dó sustenido	C#	1
Ré	D	2
Ré sus.	D#	3
Mi	E	4
Fá	F	5
Fá sus.	F#	6
Sol	G	7
Sol sus.	G#	8
Lá	A	9
Lá sus.	A#	10
Si	B	11

Figura 8. Representação das Notas Musicais

Escala musical é o nome dado a uma sequência de notas distintas. Estas escalas são categorizadas de acordo com sua tônica (ou nota raíz) e os intervalos entre as notas seguintes. Uma escala contendo todas as notas musicais é denominada *escala cromática*.

Para a identificação da escala musical contida nos elementos do buffer de entrada, é criado um segundo buffer denominado *buffer_intervalos*. Esta estrutura de até sete elementos armazena valores numéricos de 0 até 12, representando cada intervalo a partir da nota de frequência mais baixa armazenada no primeiro buffer.

```
uint8_t maior[7] =          {0, 2, 4, 5, 7, 9, 11};
uint8_t menor[7] =          {0, 2, 3, 5, 7, 8, 10};
uint8_t harmonica[7] =      {0, 2, 3, 5, 7, 8, 11};
uint8_t melodica[7] =       {0, 2, 3, 5, 7, 9, 11};
```

Figura 9. Escalas musicais representadas em intervalos a partir de uma nota raíz

Os valores contidos em *buffer_intervalos* são então comparados aos dos vetores ilustrados na figura 9 pela função *get_escala*, que faz uso do comando padrão da linguagem C *memcmp*. Quando estas duas estruturas forem iguais, a função retorna uma *string* contendo o nome da escala musical tocada pelo usuário, caso contrário a função retorna a string *"cromática"* a fim de informar ao usuário que as notas tocadas não se encaixam em nenhuma das escalas musicais definidas.

2.2.4. Visualização Pelo Usuário

O sistema atualiza o display LCD a cada um de seus ciclos, informando ao usuário quais notas musicais estão armazenadas no buffer de entrada, assim como qual a escala musical em que elas se encontram.

Após a função *get_escala* retornar qualquer valor diferente de "cromática", o sistema pausa sua execução por 2500 milisegundos enquanto exibe o resultado da função *get_escala* e os valores contidos em *buffer_notas* no display de saída, ou seja, as últimas notas e a escala musical que o usuário tocou em seu instrumento.

3. Resultados e Discussões

Durante as etapas iniciais de pesquisa e desenvolvimento, houve um foco na utilização do algoritmo de autocorrelação para a detecção de frequências fundamentais. Os trabalhos de Jarne (2017), Skjei (2011) e Alain de Cheveigné em conjunto com Hideki Kawahara (2002) levaram a crer que este algoritmo seria ideal para o sistema de identificação de notas musicais. Porém ao longo das primeiras fases de desenvolvimento do software foi observado que a implementação do mesmo era demasiadamente custosa: o Arduino utilizado não tinha capacidade computacional suficiente para realizar a tarefa em tempo real utilizando o método da autocorrelação.

Buscando aplicações específicas para o uso na música e com instrumentos musicais, o trabalho de Nessen e Needel (2013) destacou-se por descrever a fundação para um algoritmo de detecção de acordes utilizando a transformação rápida de Fourier. Utilizando os conhecimentos obtidos por meio deste trabalho correlato, o algoritmo de escolha para o desenvolvimento das funções de reconhecimento de frequência fundamental foi alterado. Dando continuidade àquele trabalho, porém alterando seu objetivo final para a detecção de escalas musicais, este trabalho utilizou a FFT a fim de permitir que o Arduino pudesse computar a frequência fundamental da onda elétrica emitida por um instrumento em tempo real, de forma virtualmente instantânea.

A aplicação demonstrou-se eficaz na identificação de notas musicais quando tocadas individualmente, assim como na classificação destas notas dentro de uma escala musical. A biblioteca responsável pela transformação rápida de Fourier permitiu que a implementação deste algoritmo de detecção de frequências fundamentais fosse facilmente implementado, mas ainda assim exigiu domínio do conhecimento sobre o funcionamento do processo: experimentações com diferentes quantidades de amostras e taxas de leitura, assim como a escolha entre diversos tipos de *windowing*, função responsável por seccionar os elementos de entrada de maneiras diferentes.

Ao fim da etapa de desenvolvimento, enquanto eram feitos testes de funcionamento, notou-se uma diferença na precisão da leitura de notas, onde ruídos como o deslizar das mãos do usuário nas cordas do instrumento não geraram tantos falso-positivos quanto anteriormente. Ao observar o código que havia sido compilado notou-se que uma chamada para a função *FFT.windowing*, da biblioteca *arduinoFFT.h*, havia sido removida. O trabalho final não conta com o uso desta função.

Para comprovar se esta melhora realmente existiu foi realizado um teste onde tocou-se a escala de *dó maior* dez vezes seguidas a uma velocidade de 150 bpm (batidas por minuto) com duração *semínima*. Os testes comprovaram um aumento de pouco mais de 60% na precisão da leitura das notas musicais.

Tabela 1. Resultados dos testes de *windowing*.

	Escalas computadas corretamente				
	1º teste	2º teste	3º teste	Total	Média
Com *windowing*	5	6	5	16	5,33
Sem *windowing*	9	9	8	26	8.66

Fonte: dados da pesquisa, 2021.

Após a remoção da chamada para esta função, o sistema mostrou-se eficiente na detecção de notas musicais entre 134 Hz e 1040 Hz, cobrindo parte das notas presentes em uma guitarra elétrica. O sistema se torna instável quando tentamos ler valores menores do que 134 Hz devido à ruídos gerados tanto pelo circuito quanto pelo usuário e seu instrumento. A fim de ilustrar a leitura desses ruídos, foi realizado um teste onde tocou-se as notas de *dó* até *sí* de forma ascendente em duas oitavas diferentes (figura 8).

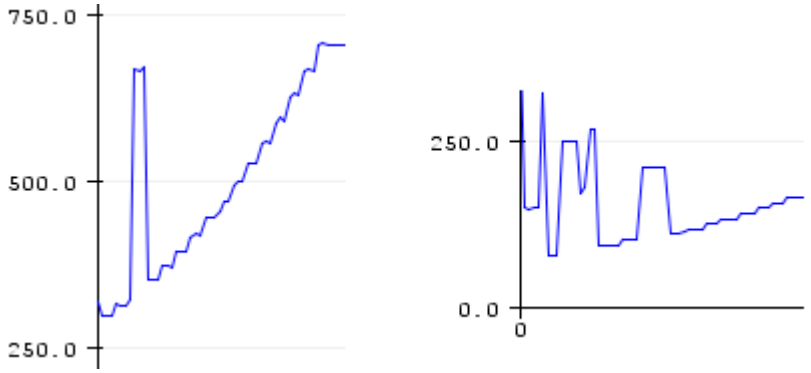

Figura 10. Comportamento da Detecção de Frequência em Frequências Médias (esq.) e Baixas (dir.)

Nota-se que ainda há erros de leitura na faixa de frequência em que o sistema atua, porém estes são raros o suficiente a ponto de não interferir no funcionamento do detector de escalas de forma negativa.

4. Conclusão

Este trabalho aplicou os conceitos de detecção de frequência fundamental em um protótipo de sistema embarcado que permite que o usuário conecte um instrumento musical no circuito, toque o instrumento e obtenha uma escala musical que representa as notas que ele tocou. Os resultados obtidos comprovaram a eficiência do uso da transformação rápida de Fourier na identificação das notas musicais geradas por impulsos elétricos oriundos de um instrumento musical.

O algoritmo desenvolvido para a identificação de notas e escalas a partir de uma frequência já definida mostrou-se eficiente em seus resultados: o algoritmo é rápido o suficiente para rodar em um processador de 84MHz como o do Arduino e seus resultados para a classificação de escalas musicais são ideais.

O algoritmo de classificação de notas musicais é eficiente, permitindo seu pleno funcionamento em uma plataforma com apenas 96KB de memória RAM. Sua eficiêcia se dá, em partes, devido ao uso de apenas uma variável de dois bytes para a classificação de três características distintas de cada nota musical: seu nome, oitava e

sua entonação. A função de classificação de escalas musicais foi desenvolvida a fim de permitir que novas escalas sejam implementadas no algoritmo com facilidade, proporcionando uma fácil expansão do número de diferentes escalas que o sistema pode reconhecer.

O hardware construído ainda pode ser otimizado, mas demonstrou que um circuito simples contando apenas com um amplificador não-inversor e um display LCD permitem o funcionamento da ferramenta. Devido à natureza do funcionamento do software, especialmente tratando-se da abrangência do uso da transformação rápida de Fourier na detecção de frequências fundamentais, pode-se concluir que o sistema é capaz de interagir com outros tipos de instrumentos musicais além de violões, baixos e guitarras necessitando apenas poucas mudanças no circuito de entrada do hardware.

A ferramenta construída, segundo as pesquisas, ainda é inexistente no mercado e seu pontencial é observavel através do funcionamento do protótipo: o sistema permite que o músico explore e observe em tempo real as relações entre diferentes notas e escalas musicais. Este novo método de estudo musical proporcionado pela ferramenta permite que o músico mantenha o foco na parte prática de seus estudos enquanto a ferramenta analisa a teoria referente ao que o mesmo está tocando.

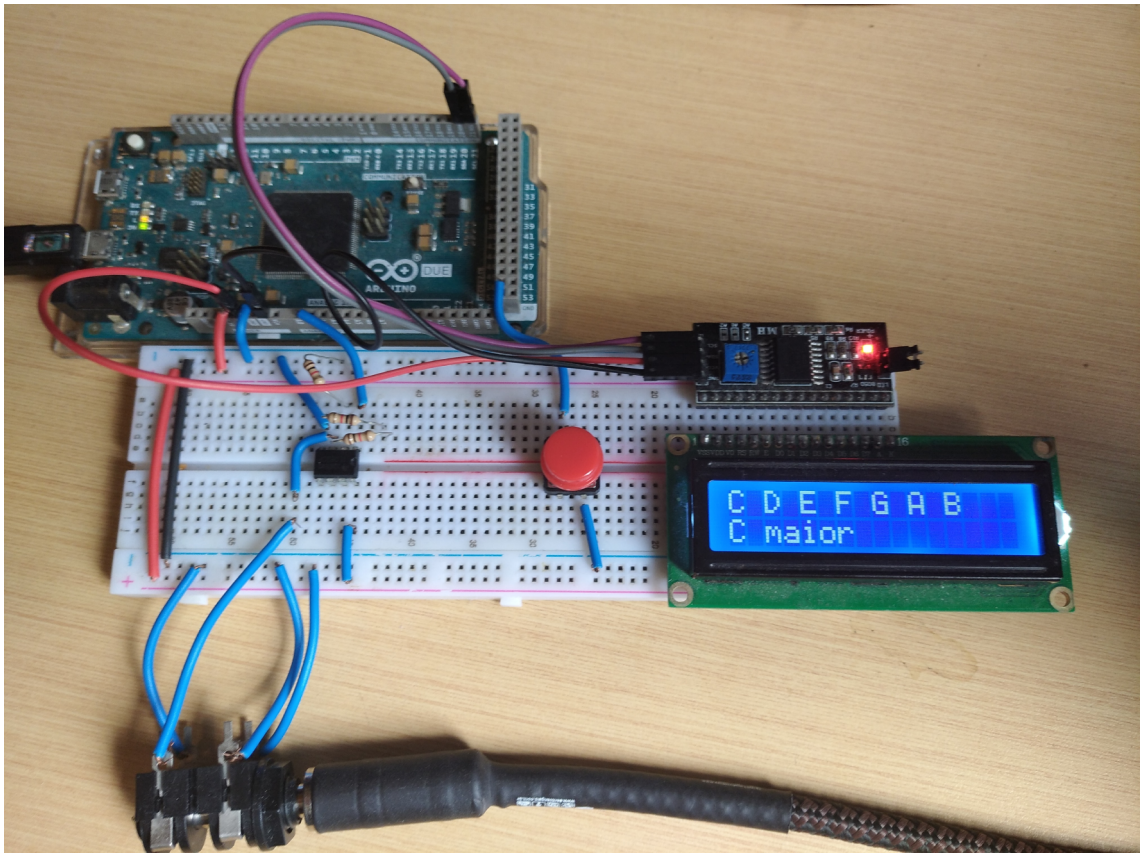

Figura 11. Hardware do sistema identificador de escalas musicais.

Buscando a criação de um produto comercial com o mesmo objetivo do protótipo, propõe-se para futuros trabalhos: implementar outras possíveis escalas a serem reconhecidas; desenvolver, com o uso das bibliotecas criadas para este protótipo, um plugin de audio no formato *VST* (*Virtual Studio Technology*) para o uso do

algoritmo em *DAWs* (*Digital Audio Workstation*); a criação de um hardware dedicado ao produto, sem o uso do Arduino; buscar soluções de hardware e software para o problema na leitura de frequências menores que 134 Hz; desenvolver um modelo portátil do produto, robusto o suficiente para ser utilizado em situações em que o músico está tocando em grupo; desenvolver um circuito de saída do sinal a fim de permitir que o usuário ligue seu instrumento no seu equipamento restante (pedais, amplificadores, caixas de som, etc…); realizar experimentos em situações de ensaios, gravações e *shows* a fim de avaliar o sistema em ambientes em que um músico normalmente se encontraria.

Referências

APICELLA, B. et al. Fast Fourier Transform and autocorrelation function for the analysis of complex mass spectra. **International Journal of Mass Spectrometry**, v. 338, p. 30–38, mar. 2013.

BRYANT, James; HENDRIKS, Paul; KESTER, Walt. Using Op Amps with Data Converters. In: JUNG, Walt. **Op Amp Applications Handbook**. [S.l]: Newnes/Elsevier, 2005. cap. 3, 56 p.

DE CHEVEIGNÉ, Alain; KAWAHARA, Hideki. YIN, a fundamental frequency estimator for speech and music. **The Journal of the Acoustical Society of America**, v. 111, n. 4, p. 1917–1930, abr. 2002.

FARAHANI, Gholamreza. Autocorrelation-based noise subtraction method with smoothing, overestimation, energy, and cepstral mean and variance normalization for noisy speech recognition. **EURASIP Journal on Audio, Speech, and Music Processing**, v. 2017, n. 1, p. 13, 21 dez. 2017.

JARNE, Cecilia. A method for estimation of fundamental frequency for tonal sounds inspired on bird song studies. **MethodsX**, v. 6, p. 124–131, 2019.

LENSSEN, N.; NEEDELL, D. **On the Mathematics of Music: From Chords to Fourier Analysis**. 11 jun. 2013.

MARTINS, et al. Estudo da Aplicação da FFT (Fast Fourier Transform) em Análise da Condição de Máquinas Rotativas. **XXXVII Iberian Latin American Congress on Computational Methods in Engineering**. 2016.

OLIVEIRA, Solamy; COELHO, José Pedro; Arduino e Educação: instrumentalização e tecnologias no ensino da música. **Revista Científica Multidisciplinar Núcleo do Conhecimento**, p. 05–15, 18 set. 2020.

SKJEI, Thomas. **Real-Time Fundamental Frequency Estimation Algorithm for Disconnected Speech.** Virginia Commonwealth University, 2011.

TAN, Li; KARNJANADECHA, Montri. **Pitch Detection Algorithm: Autocorrelation Method and AMDF**. Hat Yai: Prince of Songkhla University, jan. 2003.