A Implementação da Transformada de Fourier na Criação de um Afinador Corda a Corda de Violão com Python e Arduino

Natália da Silva Guimarães e Rafael da Silva Gonçalves

Engenharia de Computação – Universidade Federal do Ceará (UFC)
Av. José de Freitas Queiroz, 5003 - Quixadá, CE, 63902-580

{nataliaguimaraes@alu.ufc.br e rafagomes53@alu.ufc.br}

Abstract. *In this article it will be presented how the development of the system of a string-to-string guitar tuner using the Fourier Transform in Python in conjunction with the Arduino will be presented. A theoretical and practical approach was used to simplify the understanding of the controls used in the constitution of the idea, so that the final results found are easily understood. This will show the importance of signals for acoustics.*

Resumo. *Nesse artigo será apresentado como ocorreu o desenvolvimento do sistema de um afinador corda a corda para violão usando a Transformada de Fourier no Python em conjunto com o Arduino. Uma abordagem teórica e prática foi usada para simplificar o entendimento dos mecanismos utilizados na constituição da ideia, a fim de que os resultados finais encontrados sejam facilmente compreendidos. Isso mostrará a importância dos sinais para a acústica.*

1. Introdução

O objetivo desse artigo consiste em descrever como foi implementado um afinador corda a corda para violão utilizando Python e Arduino. O Python, como será explicado mais adiante, se trata de uma linguagem de programação bastante utilizada atualmente para diversos fins. Enquanto isso, o Arduino é uma placa microcontroladora de baixo custo usada na prototipação de diversos projetos, principalmente na área de embarcados. Como será exposto mais adiante, o retorno dessas duas tecnologias trabalhando em conjunto será aplicada a um circuito sinalizador de notas corda a corda utilizando resistores e leds.

Para entender melhor como o projeto foi construído, será apresentada uma breve fundamentação teórica dos principais mecanismos aplicados à essência do projeto em si. Dentre elas, será explicado o funcionamento da Transformada de Fourier, que foi implementada através da biblioteca *Numpy* do Python, em conjunto com outros módulos existentes na linguagem. Além disso, será mostrado também a essência por trás da Transformada Discreta de Fourier (TDF) e o algoritmo da Transformada Rápida de Fourier (TRF) que foi de extrema importância para filtrar as frequências por meio de um microfone conectado ao computador. Seguindo essa ordem, foi apresentado uma leve introdução sobre as origens do Python e do Arduino justificando o porquê de sua escolha para compor o sistema do afinador corda a corda.

Mais adiante, será explorado como ocorreu a montagem prática da ideia, desde a criação do circuito até as principais características operacionais que compõem a programação Por fim, serão apresentadas as metodologias necessárias para alcançar os

resultados finais do projeto, mas também uma conclusão sucinta da importância do projeto no que se refere ao ensino aprendizado sobre sinais e sistemas.

2. Fundamentação Teórica

Nesse tópico serão apresentados alguns conceitos importantes a fim de facilitar o entendimento sobre as ferramentas matemáticas e computacionais utilizadas na construção do projeto. A partir disso, será possível obter uma melhor familiarização com a implementação do afinador.

2.1 A Série de Fourier

A Série de Fourier se trata de uma série trigonométrica bastante utilizada na representação de funções infinitas e periódicas complexas de mecanismos físicos no formato de funções que utilizam senos e cossenos. Uma função periódica é definida como um sinal que apresenta uma série de repetições de seus valores durante um certo intervalo de tempo, conhecido como período. A fórmula geral da série é dada por:

$$T(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n . cos(\frac{n\pi t}{L}) + b_n . sen(\frac{n\pi t}{L})]$$

onde a_0, a_n e b_n são valores que mudam de acordo com a função que for apresentada com período fundamental $2L$ e são calculados da seguinte forma:

$$a_0 = \frac{1}{L} \int_c^{c+2L} f(t)\, dt$$

$$a_n = \frac{1}{L} \int_c^{c+2L} f(t)\, cos(\frac{n\pi t}{L})\, dt$$

$$b_n = \frac{1}{L} \int_c^{c+2L} f(t)\, sen(\frac{n\pi t}{L})\, dt$$

A Série de Fourier foi descoberta pelo matemático e físico francês Jean Baptiste Joseph Fourier (1768-1830) enquanto buscava solução para um problema físico durante os seus estudos sobre a propagação do calor em corpos sólidos. Atualmente, esse estudo é amplamente aplicado nas engenharias, mas também na acústica que é o foco do projeto do visualizador de sinais que será explicado mais adiante.

Figura 1 - Jean Baptiste Joseph Fourier (1768-1830).

2.2 A Transformada de Fourier

A Transformada de Fourier é caracterizada por representar um sinal no domínio da frequência. O atributo "transformada" se refere às possíveis representações do domínio da frequência através de um eixo em função do tempo. Diferente da Série de Fourier, a sua transformada contém limites de integração infinitos e o parâmetro de frequência (ω) manipula quaisquer valores reais. Desse modo, é possível obter:

$$F(\omega) = F_{[f]}(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(x)e^{-i\omega x}dx$$

Essa relação é bastante utilizada no estudo da acústica para distinguir notas musicais. Sabe-se que cada nota musical possui um sinal de frequência distinto que podem ser combinados gerando um sinal não periódico. Independente da complexidade do sinal obtido, os sinais originais podem ser decompostos em senóides. Desse modo, voltado essa relação para o cálculo da Transformada de Fourier, pode-se afirmar que o $f(x)$ representa a onda complexa gerada, o exponencial as senóides originais e a integral concede a possibilidade de escolher um intervalo de frequência de cada senóide original para representar o sinal combinado. Por fim, o $F(\omega)$ será definida como a magnitude e o atraso de cada sinal que foi combinado.

2.3 A Transformada Discreta de Fourier (TDF)

Como foi visto anteriormente, a Transformada de Fourier é uma ótima aplicação para sinais de tempos contínuos. Porém, quando utilizada em processamento de sinais digitais utiliza-se a Transformada Discreta de Fourier (TDF). Ela se trata de uma sequência de várias amostras de frequência de um determinado sinal em certos instantes de tempo. Matematicamente, ela é dada por:

$$F[k] = \; < f, e_k > \; = \sum_{n=0}^{N-1} f[n]e^{\frac{-2\pi ink}{N}}$$

onde $k \in Z$ e N é a quantidade de pontos da transformada. A Figura 2 mostra a representação de um sinal discreto e periódico que pode ser analisado utilizando a transformada.

Figura 2 - Amostra de um sinal discreto e periódico.

Uma simplificação do uso de uma TDF é o algoritmo de Goertzel que é um método bastante utilizado na área de processamento digital de sinais, com o objetivo de obter uma avaliação mais refinada dos mecanismos da TDF.

2.4 A Transformada Rápida de Fourier (TRF)

A Transformada Rápida de Fourier (TRF) é um algoritmo que realiza o cálculo da Transformada Discreta de Fourier (TDF). Ele é bastante importante para a medição acústica, já que ele tem a capacidade de decompor um sinal combinado em partes individuais, com o objetivo de obter informações sobre a constituição desse sinal gerado. A TRF foi implementada pelo estatístico estadunidense John Tukey (1915-2000). Em 1982, John foi contemplado com a Medalha de Honra IEEE por causa das suas contribuições, mas também pela implementação do algoritmo TRF.

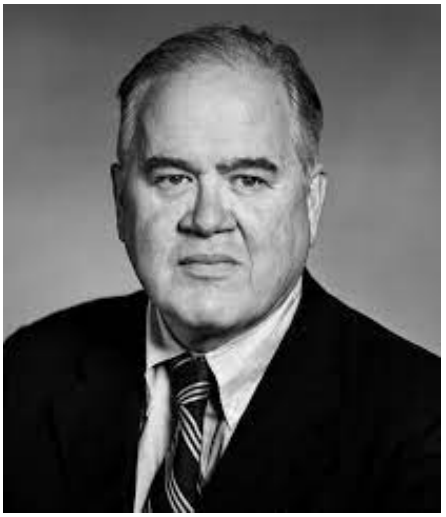

Figura 3 - Estatístico John Tukey (1915-2000).

Com o TRF é possível observar quais faixas de frequências são dominantes em qualquer sinal analógico. Para isso, basta observar qual a amplitude é a mais alta do sinal para classificá-lo como dominante. A Figura 4 mostra um sinal combinado decomposto em suas senóides originais.

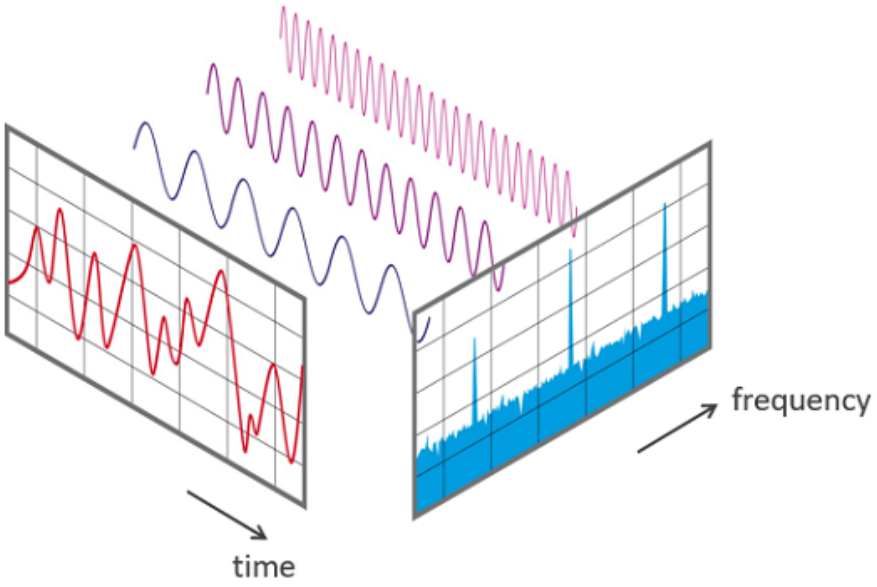

Figura 4 - Decomposição de um sinal combinado (vermelho) em suas senóides originais.

Suponha que exista um eixo z na Figura 5 representando a amplitude desse sinal. Enquanto isso, o eixo x é representado pela frequência desses sinais e o eixo y pelo tempo. Nota-se que existem vários picos e o dominante é o que possui maior amplitude. Dessa maneira, é possível enxergar que o algoritmo TRF possui 3 passos importantes:

1. A decomposição de um ponto N no domínio do tempo em diversos sinais, onde cada um detém um valor.
2. A busca do espectro de cada ponto N;
3. A obtenção de um espectro de N em uma faixa de frequência.

Uma aplicação prática do algoritmo TRF é na remoção de frequências indesejadas durante a edição de vídeos e transmissões ao vivo. Por exemplo, durante a transmissão de jogos em estádios de futebol, uma espécie de filtro de frequência aplicando o TRF consegue eliminar boa parte do barulho das arquibancadas para sobrepor de forma predominante a frequência da voz do locutor que fica narrando durante toda a transmissão do jogo.

2.5 Eletricidade Instrumental

Após uma breve abordagem teórico-matemática importante para o entendimento do projeto, torna-se necessário compreender os fundamentos básicos da eletricidade instrumental que foram aplicadas ao sinalizador de afinação do projeto. Para a construção desse dispositivo, utilizou-se leds, resistores e jumpers macho-macho. Os leds são conhecidos popularmente como diodos emissores de luz. Quando um certo valor de corrente passa pelos seus terminais ele acende indicando esse evento. Ele é amplamente utilizado como sinalizador de avisos, por isso eles podem ser encontrados de várias cores. Além disso, ele é um semicondutor polarizado, como mostra a Figura 5.

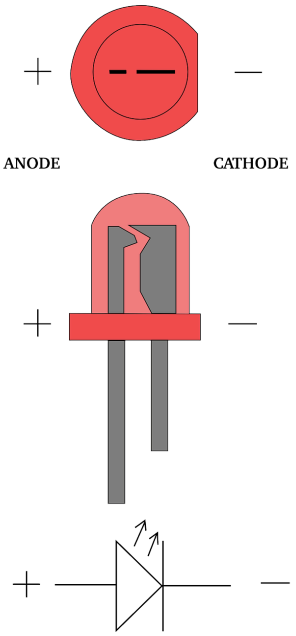

Figura 5 - Representação da polarização de um led.

Na Figura 5 observa-se na primeira representação a polarização do led. Em seguida, observa-se a sua configuração física e por fim o seu esquemático. As setas para fora indicam a emissão de luz. O outro componente utilizado em conjunto com o led foi o resistor. Como seu próprio nome sugere, ele é usado para impedir a passagem de corrente elétrica em determinado percurso do circuito. Diferente do led, o resistor não possui polarização. Dependendo da finalidade do projeto, existem vários tipos de leds que se comportam de maneiras distintas. Para o caso do sinalizador de afinação foi necessário utilizar resistores comuns como o exposto na Figura 6.

Figura 6 - Resistores de firme metálico.

2.6 Python

O Python é uma linguagem de programação de alto nível bastante utilizada em desenvolvimento web, aprendizado de máquina, desenvolvimento de aplicativos entre outros. Ela foi desenvolvida pelo matemático e programador Guido van Rossum em 1989. O Python é atualmente uma das linguagens de programação mais utilizadas na computação. Ela é bastante conhecida por ser uma linguagem simples, uma vez que possui abstrações de tipos de variáveis e ponteiros, características exigidas por linguagens como C e C++. No que se refere a sistemas embarcados que possuem recursos de hardware reduzidos, o Python não se torna uma boa opção. Um dos motivos é o fato de o seu processamento demandar muita energia do sistema, tornando-o sua implementação ineficiente. No caso da aplicação do projeto do sinalizador, o código criado em Python fica sendo executado em um computador enquanto envia sinais para a placa do Arduino que será detalhada a seguir.

Figura 7 - Símbolo do Python.

Figura 8 - Guido van Rossum.

2.7 Arduino

O Arduino é uma placa microcontroladora de hardware único e código aberto amplamente utilizada na prototipação de projetos devido ao seu baixo custo. Criado pelos desenvolvedores Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis, a ideia do projeto é até hoje um sucesso, principalmente para os hobbistas do mundo da programação e da eletrônica.

Figura 9 - Criadores do Arduino.

Existem vários tipos de Arduino, porém o que foi usado no projeto do sinalizador de afinação foi o Arduino Uno. A linguagem de programação da placa microcontroladora é uma variação de C e C++. O Arduino dispõe de sua própria IDE para os sistemas operacionais Linux, Windows e MacOs de forma gratuita. A comunidade do Arduino é bastante aquecida, dessa forma, ela se torna uma ótima opção para os iniciantes na área de sistemas embarcados. No projeto do sinalizador ela foi usada para classificar qual led deve acender de acordo com a nota musical calculada pelo programa em Python.

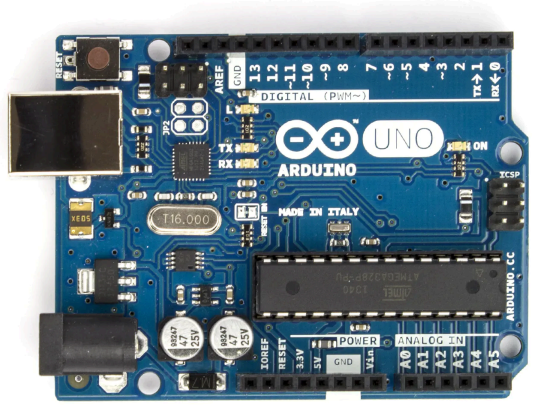

Figura 10 - Placa de um Arduino Uno.

3. Materiais e Métodos

Após uma breve abordagem teórica sobre as ferramentas utilizadas durante a construção do projeto, torna-se possível expor como ocorreu o seu processo de desenvolvimento. Como foi mencionado anteriormente, a ideia consiste na implementação de um afinador corda a corda de violão bastante utilizados leigos

iniciantes no instrumento. Os aplicativos afinadores que existem na internet oferecem duas possibilidades de afinar um violão. A primeira delas é o modo cromático. Com ele é possível afinar o violão através de notas musicais. A segunda opção é o afinador corda a corda. Assim como, o cromático ele também afina o violão, porém usa o ajuste corda a corda usando a abordagem $E_2\ A_2\ D_3\ G_3\ \ B_3\ e\ E_4$, onde:

- E_2 é a 6° corda do violão, àquela que produz o som mais grave e popularmente conhecida como Mi ou Mizão.
- A_2 é a 5° corda do violão conhecida como Lá.
- D_3 é a 4° corda do violão conhecida como Ré.
- G_3 é a 3° corda do violão conhecida como Sol.
- B_3 é a 2° corda do violão conhecida como Si.
- E_4 é a 1° corda do violão, àquela que produz o som mais agudo e popularmente chamada de Mi ou Mizinho.

As imagens a seguir mostram o aplicativo afinador da empresa Cifra Club que é bastante conhecido pelos músicos. Na Figura 11 observa-se a aba do afinador cromático, enquanto na Figura 12 é representado o afinador corda a corda que foi implementado no projeto.

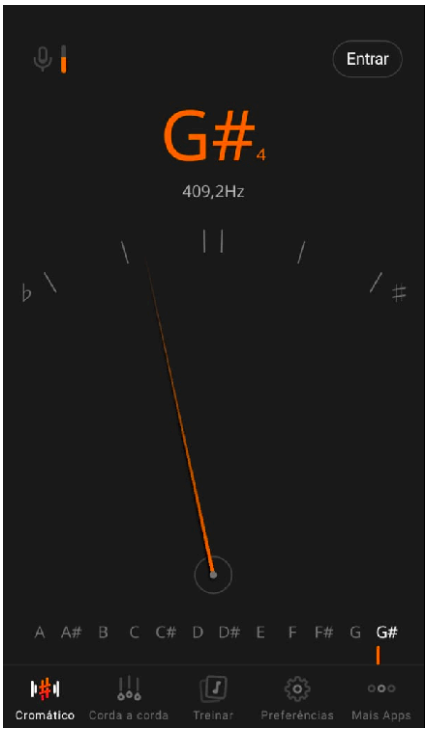

Figura 11 - Afinador cromático.

Figura 12 - Afinador corda a corda.

Sabe-se que um afinador capta a frequência de um instrumento para verificar se ele está afinado ou não. Cada corda tem seu próprio intervalo de frequência estabelecido. Desse modo, se uma determinada corda não estiver soando na frequência predefinida, o afinador exige que a corda seja ajustada. Tomando o afinador do Cifra Club como referência, foi possível verificar os intervalos de frequência em que cada corda deve trabalhar enquanto um violão era afinado. Com isso, tornou-se possível implementar um código em Python que classifica o intervalo em que cada corda deve trabalhar. Vale ressaltar que dependendo dos ruídos externos do ambiente, pode ocorrer interferência durante o ajuste do violão, uma vez que essas pequenas ondas ruidosas são somadas à frequência da corda do instrumento. A Tabela 1 a seguir mostra as faixas de frequências configuradas para cada corda do violão de acordo com o aplicativo Cifra Club.

Notação	Frequência (Hz)
E_4	329,6
B_3	246,9
G_3	196
D_3	146,8
A_2	110
E_2	82,4

Tabela 1 - Frequências exigidas em cada corda do violão para está afinado de acordo com o afinador Cifra Club.

Desse modo, foi possível ajustar o código em Python com esse valores, aproximadamente. Isso acontece porque utiliza-se um microfone conectado ao computador que gera algumas variações enquanto as frequências são captadas.

3.1 Montagem do Circuito do Sinalizador

A fim de verificar se cada corda estava na frequência adequada, foi implementado um sinalizador de afinação usando leds. A Figura 13 mostra como ficou a montagem do circuito.

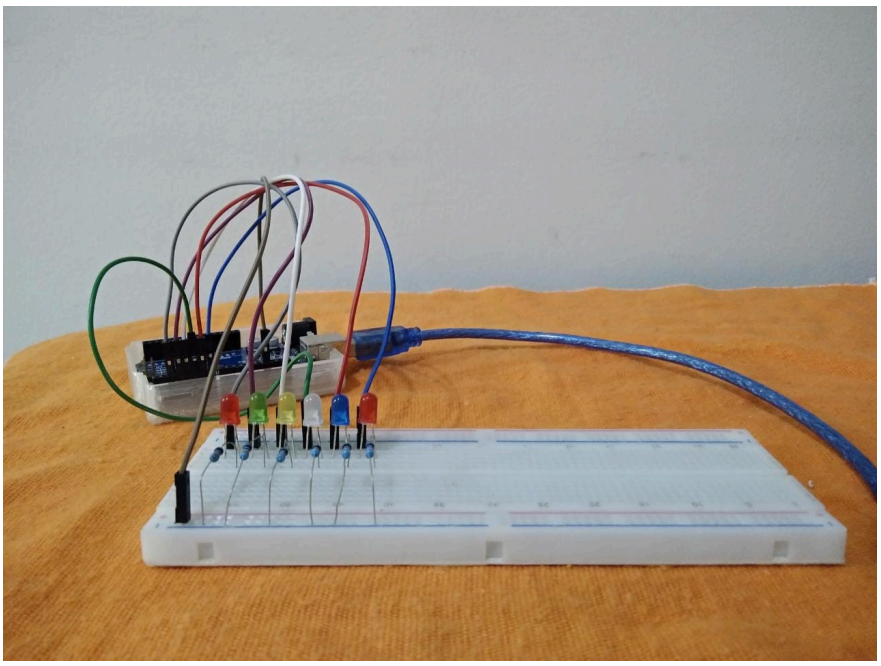

Figura 13 -Circuito do sinalizador de afinação.

O circuito é composto de 6 leds. Cada um representa uma corda do violão. A ordem segue da esquerda para a direita, da 1° à 6° corda. De acordo com a Tabela 1, percebe-se que a 6° corda, que é a mais grave, possui a menor frequência em comparação com as outras cordas. Isso significa que quanto mais agudo o som, mais alta é a frequência. Desse modo, fica perceptível que cada corda do violão tem uma frequência específica que não interferem uma na outra. Por exemplo, se a 1° corda estiver corretamente afinada, o led vermelho na extrema esquerda irá acender e os outros permanecerão apagados, uma vez que a frequência dominante é a do Mizinho e assim sucessivamente.

Para implementar o sinalizador de afinação foi usado resistores de 330Ω em conjunto com os leds para evitar curto circuito, mas também evitar que o led possa queimar, devido a alta tensão advindas das portas do Arduino. Além disso, cada led está conectado em uma porta digital do Arduino, como será mostrado no código mais adiante.

3.2 Implementação do Código Classificador de Frequência em Python

Devido a quantidade de bibliotecas prontas que realizam cálculos matemáticos, plotagem de gráficos e outros mecanismos, o Python foi escolhido para compor a parte da aplicação prática da equação da Transformada de Fourier no domínio da frequência. A Figura 14 mostra como ficaram programadas as primeiras linhas de código feitas na IDE do software Visual Studio Code.

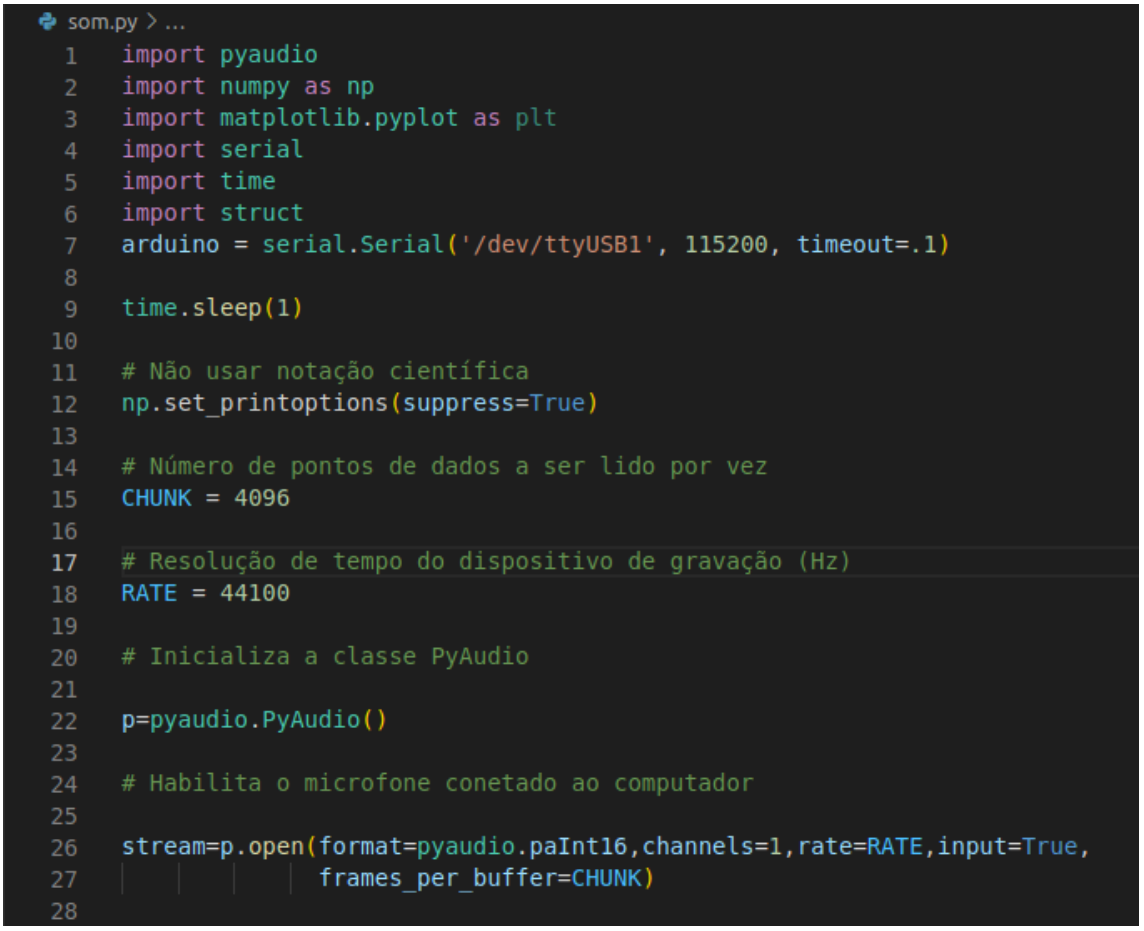

Figura 14 - Primeiras linhas de código do Python.

A priori, foi necessário importar as bibliotecas usadas durante o programa. A primeira delas é o *PyAudio*. Ela é responsável pelo reconhecimento de voz que é captado pelo microfone do computador. A segunda biblioteca é o *Numpy*. Ela contém diversos métodos matemáticos, dentre eles os responsáveis pelo cálculo da Transformada Rápida de Fourier (TRF). Em seguida, foi necessário acrescentar a biblioteca *Matplotlib.* Ela é bastante usada para plotar gráficos numéricos em Python. No código implementado, ela foi usada para gerar os gráficos correspondentes às amostras dos sinais discretos captados durante certo intervalo de tempo. Depois, foi chamada a biblioteca *Serial* que é a responsável por enviar dados através da porta USB do computador para o Arduino. Além disso, foi também chamada a biblioteca *Time* que trabalha com valores relacionados ao tempo, como as datas do calendário, por exemplo. A última biblioteca importada é a *Struct.* Ela é a responsável por converter estruturas em Python para a linguagem C.

Na linha 7 foi criada uma variável chamada *arduino.* Ela recebe a configuração serial de acordo com os parâmetros apresentados para se comunicar com o Arduino. Em seguida, na linha 9 a função *time.sleep()* estabelece um tempo para que a conexão com a porta ocorra. Enquanto isso, na linha 12 a função se responsabiliza por limitar os valores do cálculo sem notação científica. Ademais, foram criadas as variáveis *CHUNK* e *RATE* com os valores de 4096 e 44100, respectivamente. O primeiro estabelece a quantidade

de pontos de dados a ser lido por vez e o segundo gera um tempo de gravação da frequência em Hertz (Hz).

Na linha 22, a classe do *PyAudio* é chamada para inicializar a conexão com o microfone do computador na linha 26.

```
29  # Cria um array numpy contendo uma única leitura de dados de áudio
30  # Fica executando o código infinitamente
31
32  while True:
33
34      # Realiza a leitura dos dados de 16 bit inteiro
35      data = np.fromstring(stream.read(CHUNK),dtype=np.int16)
36
37      # Suaviza o TRF por janelas de dados
38      data = data * np.hanning(len(data))
39
40      # Filtra apenas a parte positiva do sinal
41      fft = abs(np.fft.fft(data).real)
42
43      # Calcula a frequência usando o TDF
44
45      fft = fft[:int(len(fft)/2)]
46      freq = np.fft.fftfreq(CHUNK,1.0/RATE)
47      freq = freq[:int(len(freq)/2)]
48      freqPeak = freq[np.where(fft==np.max(fft))[0][0]]+1
49
```

Figura 15 - Laço infinito que realiza os cálculos da frequência através da Transformada Rápida de Fourier (TRF).

Lendo atentamente os comentários percebe-se que o laço *while* realiza os cálculos de acordo com dados captados pelo microfone. Nota-se que ele utiliza várias funções do módulo *Numpy* para aplicar os cálculos referentes a Transformada Discreta de Fourier (TDF) e o algoritmo da Transformada Rápida de Fourier. Antes disso, o código realiza a leitura de dados de 16 bits inteiros, transformando o cálculo do TRF em janela de dados para finalmente as faixas de dados no domínio da frequência durante certas amostras de tempo.

```
50      # Imprime os valores de frequências em (Hz)
51      print("Frequência: %d Hz"%freqPeak)
52
53      # Classifica as frequências e envia o sinal para o Arduino
54
55      if (freqPeak > 300 and freqPeak <= 340):
56          print("Mizinho")
57          arduino.write(struct.pack('>B', 1))
58      elif (freqPeak > 200 and freqPeak <= 255):
59          print("Si")
60          arduino.write(struct.pack('>B', 2))
61      elif (freqPeak > 170 and freqPeak <= 199):
62          print("Sol")
63          arduino.write(struct.pack('>B', 3))
64      elif (freqPeak > 120 and freqPeak <= 157):
65          print("Ré")
66          arduino.write(struct.pack('>B', 4))
67      elif (freqPeak > 90 and freqPeak <= 115):
68          print("Lá")
69          arduino.write(struct.pack('>B', 5))
70      elif (freqPeak >= 65 and freqPeak <= 100 and freqPeak != 162):
71          print("Mizão")
72          arduino.write(struct.pack('>B', 6))
73      else:
74          arduino.write(struct.pack('>B', 0))
75
76  # Mostra os gráficos das amostras
77
78  # plt.plot(freq,fft)
79  # plt.axis([0,4000,None,None])
80  # plt.show()
81  # plt.close()
82
83  # Encerra a execução do programa normalmente
84
85  stream.stop_stream()
86  stream.close()
87  p.terminate()
```

Figura 16 - Parte do código onde foi implementada a classificação das frequências.

Como foi apresentado anteriormente, cada corda do violão possui um intervalo definido de frequência sonora para está corretamente afinado. Para determinar as faixas de valores definidas no código, foi necessário afinar um violão através do aplicativo do Cifra Club e observar em quais frequências cada corda do instrumento trabalhava. Isso gerou os dados da Tabela 1.

A posteriori, um microfone de boa precisão foi conectado no computador com a finalidade de verificar se as frequências da Tabela 1 eram identificadas pelo código em Python. Devido às baixas frequências ruidosas do ambiente, mas também da captação do próprio microfone, as faixas lidas pelo programa variaram um pouco, porém não fugiram muito do resultado real. Isso pode ser visto na linha 70 do código, onde existe a

restrição do *freqPeak != 162*, já que ela tenta sobrepor a frequência do Ré. Após essa análise, foram implementados vários desvios condicionais que verificam se aquela frequência está realmente relacionada àquela corda. Caso a condição seja verdadeira, o Arduino receberá um sinal que varia de 0 à 6, onde 0 significa que o som emitido por umas das cordas do violão estão em uma faixa desconhecida e inaceitável. Se, por exemplo, o Arduino receber 1, significa que a corda tocada emite uma frequência equivalente à primeira corda do violão e assim sucessivamente. Dessa maneira, fica nítido que se a primeira corda estiver afinada, o led vermelho na extrema esquerda, visto na Figura 13 irá acender.

Os métodos comentados entre as linhas 78 e 81 plotam amostras dos gráficos da amplitude x frequência do sinal. Finalmente, as linhas de código mais abaixo encerram a execução do código quando ele é interrompido.

3.3 Implementação do Código do Classificador de Leds no Arduino

Durante a apresentação do Arduino, foi mencionado que ele possui a sua própria IDE. Tendo isso em vista, fica evidente que ela foi usada para o desenvolvimento do classificador dos leds. Sabe-se que o Arduino possui portas analógicas e digitais. Como os leds são sensíveis a variações bruscas de corrente, utiliza-se os pinos digitais para trabalhar adequadamente com esses componentes. A Figura 17 mostra as portas onde os leds foram inseridos.

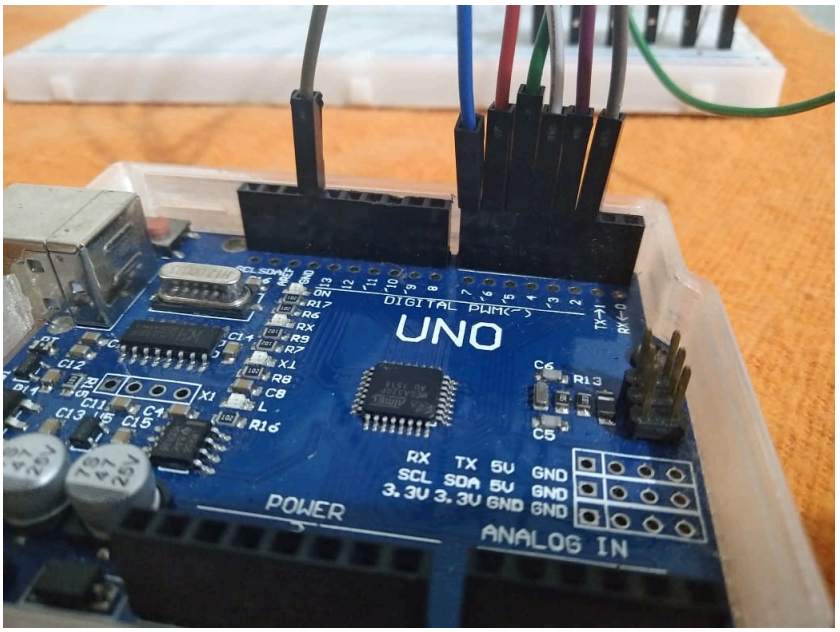

Figura 17 - Portas onde estão conectados os 6 leds e o GND.

As portas digitais escolhidas são da 2 até a 7 seguindo a sequência de leds da esquerda para a direita. Nesse caso, não é necessário alimentar o circuito com algum valor de tensão, uma vez que as próprias portas digitais fornecem energia para os componentes. Na Figura 18, observa-se as primeiras linhas de código desenvolvidas em C/C++ na IDE do Arduino.

som_ino

```
// Recebe o byte de entrada

int byte_entrada = 0;

// Usando na verificação do sinal recebido

int valor;

// Configuração dos pinos dos leds

const int led_mizinho = 2;
const int led_si = 3;
const int led_sol = 4;
const int led_re = 5;
const int led_la = 6;
const int led_mizao = 7;

// Interruptor de depuração

int opc_debug = false;

// Caso verdadeiro para testar os leds antes de usá-los

int cond_led = true;

// Armazena o valor recebido do código em Python

int sinal = 0;
```

Figura 18 - Configuração das primeiras linhas de código no Arduino.

A Figura 18 mostra como ficou elaborada as primeiras linhas de código no Arduino. Primeiramente, foi criada uma variável inteira chamada *byte_entrada* igual a 0. Ele recebe um valor de 0 a 6 do código em Python explicado anteriormente. Além disso, a variável *valor* é criada para fazer a comparação mais adiante que permite a classificação dos estados ativos dos leds. Mais adiante, foram implementadas variáveis inteiras constantes sinalizando as portas digitais onde cada led está presente.

Nesse viés, foram criadas também os valores *opc_debug* e *cond_led.* As suas finalidades são de ativar o estado de debug do sistema e permitir que os leds presentes na protoboard sejam testados um a um, como será mostrado mais adiante. Outra variável de extrema importância para o programa é o *sinal.* Ele recebe a variável *byte_entrada* que é responsável por pegar o valor bruto do programa em Python para em seguida ser usada nos desvios condicionais que permitem o piscar dos leds dependendo da frequência identificada.

```
void setup(){

  // Usa a mesma taxa de transmissão do Python
  Serial.begin(115200);

  if (cond_led == true) {

    // Testa os leds
    teste_leds();

  }
}

void loop() {

  if(opc_debug == true){
    debug();
  }else{
    Serial.println("Erro no debug!");
  }

  if(Serial.available() > 0) {
    byte_entrada = Serial.read();
    sinal = byte_entrada ;
    Serial.print("Eu recebi: ");
    Serial.println(byte_entrada);
  }

  // Realiza a animação dos leds

  classificador();

}
```

Figura 19 - Implementação das funções *void setup()* e *void loop().*

Na função *void setup()* é feita a calibragem da taxa de transmissão entre o código Python e o Arduino, através da função *Serial.begin(115200)*. Em seguida, é possível observar um desvio condicional que permite que os leds sejam testados antes de serem acesos de acordo com os valores de 0 a 6 recebidos do código em Python. Enquanto isso, a função *void loop()* possui um *if* que é executado caso a variável que ativa o debug seja verdadeira. Com isso, ele permite o funcionamento da função *debug(),* caso contrário, retorna uma mensagem de falha. Em seguida, é verificado se a conexão serial está disponível. Se a resposta for sim, o valor advindo do código em Python é lido, armazenado em *sinal* e impresso através do monitor serial da placa microcontroladora. Tendo isso em vista, é possível chamar a função *classificador()* que é a responsável por causar a animação dos leds.

```
void debug() {

  Serial.print("Valor do Som: ");

  if (sinal >= valor) {
    Serial.print(sinal);
    Serial.println(" Led Ligado!");

  }else{
    Serial.println(sinal);
  }
}

void teste_leds(){

  digitalWrite(led_mizinho, HIGH);
  delay(1000);
  digitalWrite(led_si, HIGH);
  delay(1000);
  digitalWrite(led_sol, HIGH);
  delay(1000);
  digitalWrite(led_re, HIGH);
  delay(1000);
  digitalWrite(led_la, HIGH);
  delay(1000);
  digitalWrite(led_mizao, HIGH);
  delay(1000);

}
```

Figura 20 - Escopo das funções *debug()* e *teste_leds()*.

A função *debug()* tem como finalidade observar o comportamento do sinal recebido do código em Python por meio do monitor serial da IDE do Arduino. A priori, é feita uma comparação. Se o valor de *sinal* for maior ou igual a do *valor*, o led ao sinal correspondente acende ou todos se mantêm apagados, caso o sinal seja igual a 0. Caso contrário, ele só mostra o valor do sinal desconhecido na tela. Já a função *teste_leds()* pisca todos os leds da protoboard uma vez para ver se estão funcionando perfeitamente.

```
void classificador(){

  if(sinal == 0){
    digitalWrite(led_mizinho, LOW);
    digitalWrite(led_si, LOW);
    digitalWrite(led_sol, LOW);
    digitalWrite(led_re, LOW);
    digitalWrite(led_la, LOW);
    digitalWrite(led_mizao, LOW);

  }else if(sinal == 1){
    digitalWrite(led_mizinho, HIGH);
    delay(300);
    digitalWrite(led_si, LOW);
    digitalWrite(led_sol, LOW);
    digitalWrite(led_re, LOW);
    digitalWrite(led_la, LOW);
    digitalWrite(led_mizao, LOW);

  }else if(sinal == 2){

    digitalWrite(led_mizinho, LOW);
    digitalWrite(led_si, HIGH);
    delay(300);
    digitalWrite(led_sol, LOW);
    digitalWrite(led_re, LOW);
    digitalWrite(led_la, LOW);
    digitalWrite(led_mizao, LOW);

  }else if(sinal == 3){

    digitalWrite(led_mizinho, LOW);
    digitalWrite(led_si, LOW);
    digitalWrite(led_sol, HIGH);
    delay(300);
    digitalWrite(led_re, LOW);
    digitalWrite(led_la, LOW);
    digitalWrite(led_mizao, LOW);
```

Figura 21 - Parte da função *classificador().*

A implementação dessa função é bastante simples. Através de vários *if* é possível sinalizar qual led deve acender. Por exemplo, se a variável *sinal* receber 0, todos os leds da protoboard devem ficar apagados. Em contrapartida, se o sinal for 1, significa que a frequência captada é equivalente à intensidade sonora de um Mizinho. Se a 1° corda do violão obedecer esse princípio ela está afinada. Caso outra luz de led pisque enquanto essa corda está sendo manipulada, a mesma encontra-se desafinada. Finalmente, vale ressaltar que a configuração de desvio condicional segue até a comparação do *if* de quando o *sinal* é igual a 6.

4. Resultados

Após a construção do projeto, tornou-se possível realizar testes a fim de verificar se o sistema retorna de fato o comportamento de um afinador corda a corda de um violão. A priori, um microfone de boa precisão foi conectado ao computador. Em seguida, foi necessário gravar o código do Arduino na placa. Com isso, o código em Python foi executado, a fim de captar os valores de frequência do ambiente externo gerado pelas cordas do violão. Durante a explicação do código em Python, foi mencionado que a biblioteca *Matplotlib* era a responsável por gerar gráficos matemáticos. Tendo isso em vista, as linhas responsáveis pela plotagem foram ativadas, com o objetivo de analisar o formato de onda gerado pela frequência. Como exemplo, foi tocada a 2° corda do violão que representa o Si. Assim, o led verde acendeu e o gráfico da Figura 22 foi gerado.

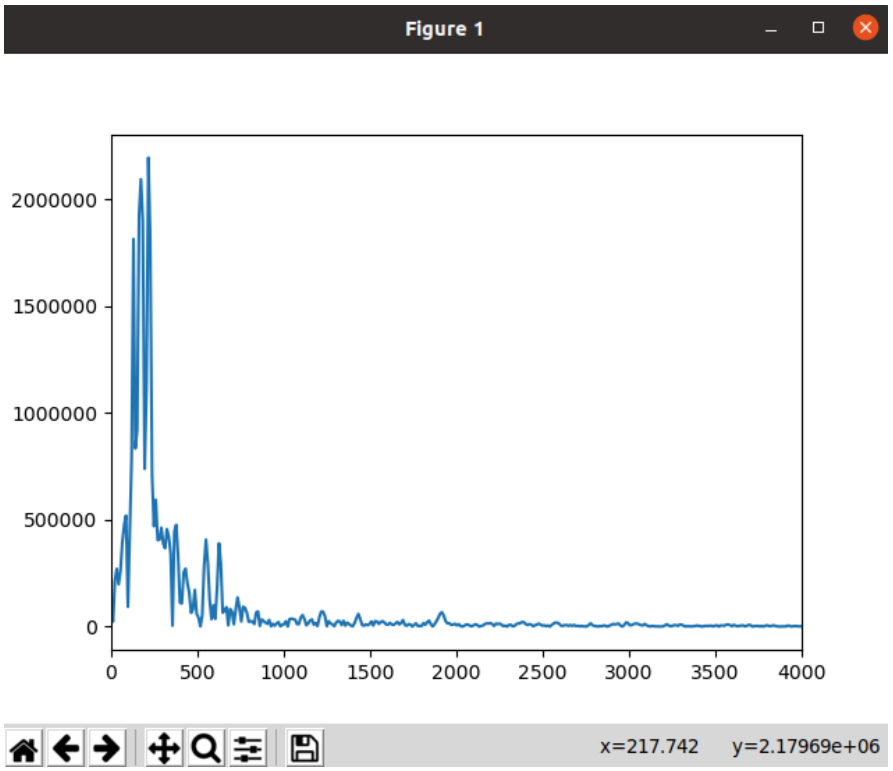

Figura 22 - Onda gerada pela frequência da 2° corda (Si) do violão.

Como foi argumentado em tópicos anteriores, o sinal captado pelo microfone pode ser somado a pequenos ruídos externos do ambiente. Desse modo, não é possível obter resultados muito precisos, já que até o funcionamento do microfone podem gerar pequenas frequências indesejadas. Para resolver esse problema, seria necessária a implementação de um filtro que pode ser encontrado em outros equipamentos acústicos mais profissionais. Observando atentamente a Figura 22, percebe-se que ao mover o mouse em direção ao pico do sinal, a frequência captada está na faixa de intervalo definida no código em Python para a 2° corda do violão. Uma outra característica observada enquanto o sistema estava fazendo leitura é que quando a corda do violão começa a parar de tocar, os outros leds começam a piscar, indicando para o usuário do

sistema que a frequência está mudando para outras faixas. Quando o sistema encontra o seu pico, o led fica aceso por mais tempo.

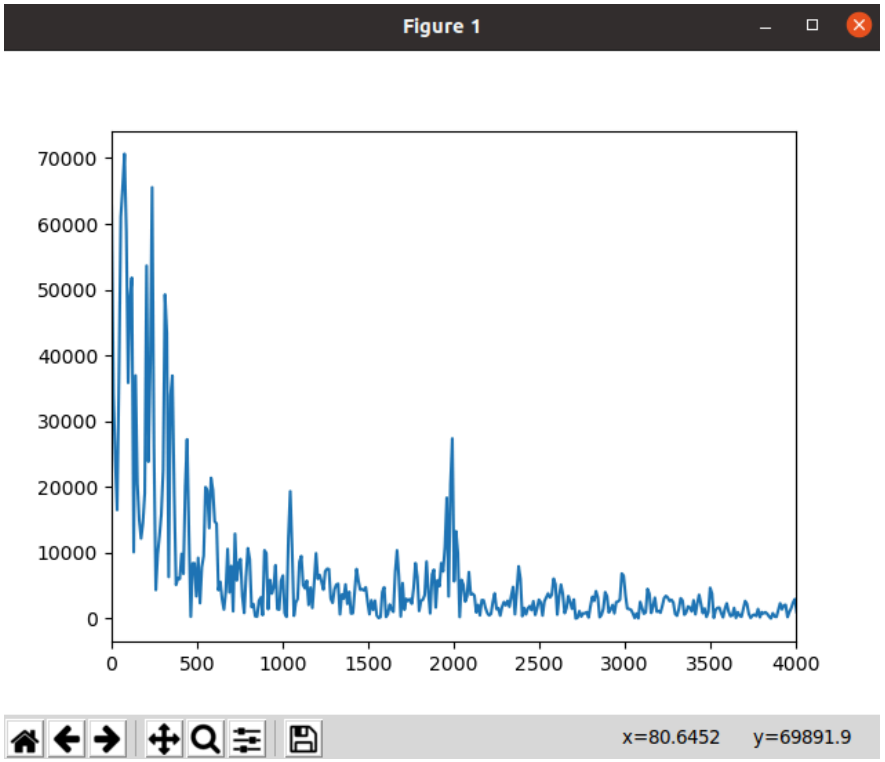

Figura 23 - Onda gerada pela frequência da 6° corda (Mizão) do violão.

Na Figura 23 é mostrado o sinal gerado pela 6° corda do violão conhecida como Mizão. Observa-se que a frequência gerada por ela é bem menor que o Si, uma vez que ela emite uma tonalidade mais grave. Desse modo, o led predominante será o vermelho da extrema direita. As imagens a seguir mostram os leds acesos quando se tocam as 2° e 6° cordas de um violão afinado, respectivamente.

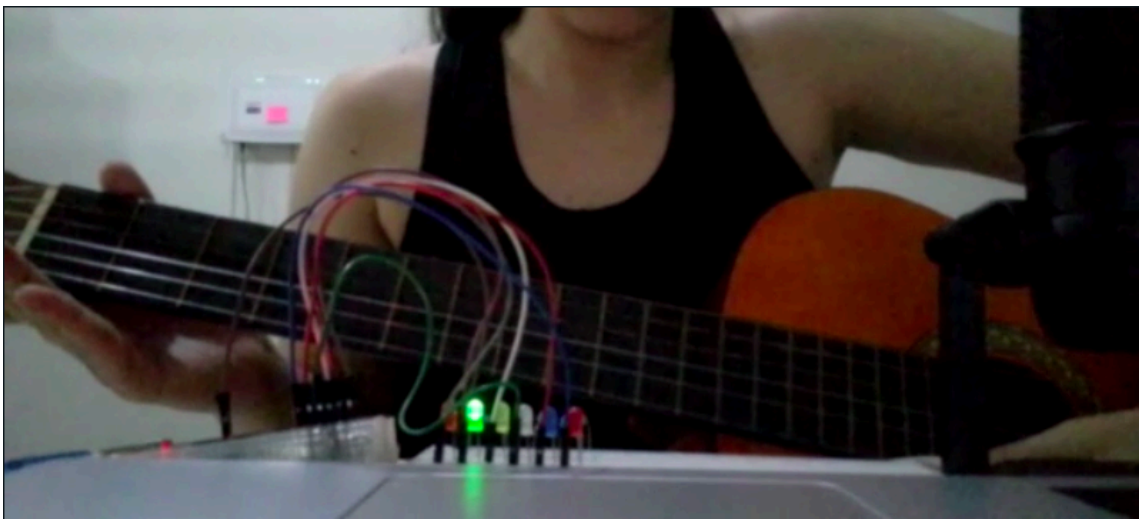

Figura 24 - Led verde aceso indicando que a 2° do violão (Si) está afinada.

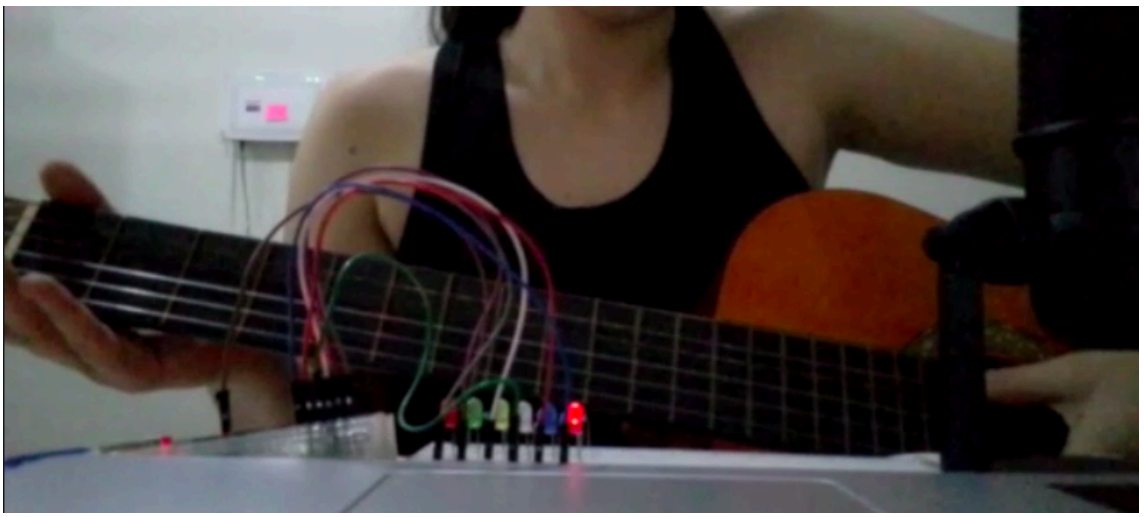

Figura 25 - Led verde aceso indicando que a 6° do violão (Mizão) está afinada.

O tempo que o led fica aceso quando identifica o seu pico de domínio é de 300 ms. Com isso, pode-se concluir que o projeto apresentou resultados satisfatórios. Como foi exposto anteriormente, o sistema pode ser aperfeiçoado incrementando um filtro de ruídos para obter uma afinação mais precisa. Porém, para fins didáticos, as saídas do sistema foram suficientes para obter um conhecimento prático sobre a aplicabilidade dos sinais em nosso cotidiano.

5. Conclusão

A implementação deste projeto é de extrema importância, uma vez que ele permite uma maior familiarização prática dos conhecimentos adquiridos durante a disciplina de sinais e sistemas. Devido às várias pesquisas realizadas durante o desenvolvimento do afinador, ficou evidente que os cálculos da Transformada de Fourier são de grande relevância para a maioria dos sistemas que são utilizados no dia a dia. Pode-se afirmar que esse projeto é constituído de uma certa interdisciplinaridade com cálculo e eletrônica. Portanto, no que se refere a engenharia de sistemas embarcados torna-se impossível não entrar em contato com o mundo dos sinais, sejam eles contínuos ou discretos, já que vários sensores, como por exemplo, ultrassônico, temperatura, infravermelho, entre outros, necessitam de amostras de sinais para reagir de forma apropriada aos estímulos do ambiente externo.

No caso do projeto do afinador corda a corda, a sua função é ajudar estudantes iniciantes de violão que não sabem as notas afinar o seu instrumento. Com isso, é notório que a importância dos estudos dos sinais devem ser consideradas sempre que um projeto for desenvolvido, já que eles são responsáveis por informar aos usuários as condições presentes no ambiente, a fim de auxiliar na realização das tarefas do cotidiano

6. Referências Bibliográficas

AUTOMAÇÃO E CONTROLE DE PROCESSOS: CONHEÇA O ARDUINO. Empeltec Jr, 2020. Disponível em: https://empeltecjr.com/automacao-controle-e-arduino/. Acesso em 19 de novembro de 2022.

ARDUINO. Wikipédia, 2022 Disponível em: https://pt.wikipedia.org/wiki/Arduino#:~:text=Massimo%20Banzi%2C%20David%20Cuartielles%2C%20Tom,Gianluca%20Martino%20e%20David%20Mellis. Acesso em 19 de novembro de 2022.

TRANSFORMADA DE FOURIER DE TEMPO DISCRETO. Wikipedia, 2022. Disponível em: https://pt.wikipedia.org/wiki/Transformada_de_Fourier_de_tempo_discreto. Acesso em 19 de novembro de 2022.

SERIE DE FOURIER. Wikipedia, 2022. Disponível em: https://pt.wikipedia.org/wiki/S%C3%A9rie_de_Fourier. Acesso em 19 de novembro de 2022.

JOHN TUKEY. Wikipedia, 2022. https://pt.wikipedia.org/wiki/John_Tukey. Acesso em 19 de novembro de 2022.

TRASNFORMADA DE FOURIER. Wikipedia, 2022. https://pt.wikipedia.org/wiki/Transformada_de_Fourier#:~:text=A%20transformada%20de%20Fourier%20%C3%A9%20chamada%20de%20representa%C3%A7%C3%A3o%20do%20dom%C3%ADnio,frequ%C3%AAncia%20a%20uma%20fun%C3%A7%C3%A3o%20temporal. Acesso em 19 de novembro de 2022.

TRANSFORMAÇÃO RÁPIDA DE FOURIER FFT - NOÇÕES BÁSICAS. Nti, 2022. https://www.nti-audio.com/pt/suporte/saber-como/transformacao-rapida-de-fourier-fft#:~:text=Passo%20a%20passo&text=Dois%20par%C3%A2metros%20s%C3%A3o%20relevantes%3A,o%20comprimento%20de%20bloco%20BL. Acesso em 20 de novembro de 2022.

YNOGUTI, Carlos Alberto. "Transformada Discreta de Fourier"; Inatel. Disponível em: https://www.inatel.br. Acesso em 20 de novembro de 2022.

OLIVEIRA, Carlos de Oliveira. "O passo a passo de como afinar o violão", Cifra Club. Disponível em: https://www.cifraclub.com.br/blog/como-afinar-o-violao/. Acesso em 21 de novembro de 2022.

SANTOS, Camila Stenico. "Usando a transformada de Fourier para construir um visualizador de música com Python e Arduino"; Medium. Disponível em: https://medium.com/@cafelouco/usando-a-transformada-de-fourier-para-construir-um-visualizador-de-m%C3%BAsica-com-python-e-ardu%C3%ADno-57c98e723cdc. Acesso em 21 de novembro de 2022.

JEAN BAPTISTE JOSEPH FOURIER. Wikipedia, 2022. Disponível em: https://pt.wikipedia.org/wiki/Jean_Baptiste_Joseph_Fourier. Acesso em 21 de novembro de 2022.

JOHN W. TUNKEY. National Academy of Sciences, 2018. Disponível em: http://www.nasonline.org/publications/biographical-memoirs/memoir-pdfs/tukey-john.pdf. Acesso em 21 de novembro de 2022.

MARTINS, Andressa Martins. "Como calcular o resistor correto para um LED", Nerd Rosa. Disponível em: https://www.nerd-rosa.com.br/post/como-calcular-o-resistor-correto-para-um-led. Acesso em 21 de novembro de 2022.

RESISTOR DE PRECISÃO 5,6Ω DE 1% 1/4 PTH 5K6. Ryndack Componentes, 2022. Disponível em: https://www.ryndackcomponentes.com.br/resistor-5-6k-1-1-4w-5-6k-5k6.html. Acesso em 21 de novembro de 2022.

PYTHON (PROGRAMMING LANGUAGE). Wikipedia, 2022. Disponível em: https://en.wikipedia.org/wiki/Python_%28programming_language%29. Acesso em 21 de novembro de 2022.

GUIDO VAN ROSSUM. Wikipedia, 2022. Disponível em: https://pt.wikipedia.org/wiki/Guido_van_Rossum. Acesso em 21 de novembro de 2022.

MOREIRA, Alexandre Harayashiki. "Como calcular o resistor correto para um LED", Instituto Conhecimento Liberta. Disponível em: https://icl.com.br/curso/robotica-i-introducao-ao-arduino/. Acesso em 21 de novembro de 2022.