CENTRO UNIVERSITÁRIO UNIVATES
CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO

# SISTEMA AUTOMATIZADO PARA AFINAÇÃO DE VIOLÃO ELÉTRICO

Natan Gabriel Becchi

Lajeado, junho de 2017

Natan Gabriel Becchi

# SISTEMA AUTOMATIZADO PARA AFINAÇÃO DE VIOLÃO ELÉTRICO

Monografia apresentada na disciplina de Trabalho de Conclusão de Curso – Etapa II, do Curso de Engenharia de Controle e Automação, do Centro Universitário UNIVATES, como parte da exigência para a obtenção do título de Bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Me. Henrique Worm

Lajeado, junho de 2017

Natan Gabriel Becchi

# SISTEMA AUTOMATIZADO PARA AFINAÇÃO DE VIOLÃO ELÉTRICO

A Banca examinadora abaixo aprova a Monografia apresentada na disciplina de Trabalho de Conclusão de Curso – Etapa II, do Curso de Engenharia de Controle e Automação, do Centro Universitário UNIVATES, como parte da exigência para a obtenção do grau de Bacharel em Engenharia de Controle e Automação:

Prof. Me. Henrique Worm – orientador
Centro Universitário UNIVATES

Prof. Me. Anderson Antônio Giacomolli
Centro Universitário UNIVATES

Prof. Me. Juliano Schirmbeck
Centro Universitário UNIVATES

Lajeado, 13 de julho de 2017

# RESUMO

Este trabalho objetiva o desenvolvimento de um sistema automatizado para afinação de um violão elétrico, fazendo uso de uma placa microprocessada. Um violão elétrico convencional é equipado com um captador ativo, que converte a vibração mecânica produzida pelas cordas, ao serem tocadas, em uma tensão alternada de alguns milivolts. Esta tensão, presente no terminal de saída do instrumento, é polarizada e amplificada para que possa ser corretamente interpretada pelo microcontrolador. Neste controlador, o sinal é processado digitalmente, de modo que a frequência referente à nota musical emitida pelo violão esteja disponível ao sistema em formato numérico. Uma vez obtida esta frequência, o programa desenvolvido para a placa compara-a a valores de frequência conhecidos, correspondentes às notas musicais válidas para as cordas do instrumento. Ao determinar o quão mais grave ou mais agudo é o som produzido em relação às frequências de referência, o sistema comuta saídas digitais ligadas a uma interface de potência para fazer o controle de um motor acoplado às tarraxas do violão, de modo a ajustar a tensão mecânica das cordas tornando o som mais grave ou mais agudo, conforme for o caso. Os resultados obtidos validam a aplicação do programa desenvolvido de forma genérica, sendo requeridas as adequações ao hardware utilizado.

**Palavras-chave**: DSP. Microcontrolador. Música.

## ABSTRACT

This paper aims to develop an automated electro-acoustic guitar tuning system by using a microprocessor board. A standard electro-acoustic guitar is provided with an active pickup, which transduces the mechanical vibration created by the strings, whilst played, into a few-millivolt alternating voltage. This voltage, existing in the guitar's output jack, is polarized and amplified so it may be properly read by the microcontroller. Within this controller, the signal is digitally processed, so the frequency of the played musical note is numerically available to the system. Once the frequency is acquired, the software compares it to known frequency values, which are the valid musical notes for the strings of a guitar. By defining how higher or lower is the produced pitch in comparison to reference frequencies, the system toggles digital outputs connected to a power interface in order to have a motor actuating on the guitar's tuning pegs, as to adjust the strings' mechanical tension having them sound in a higher or lower pitch, accordingly. Results validate the application of the developed program in a generic way, leaving the required adjustments to the selected hardware.



**Keywords**: DSP. Guitar. Microcontroller. Music. Tuning.

# LISTA DE FIGURAS

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| ADC | Analog-to-Digital Converter |
| DFT | Discrete Fourier Transform |
| DSP | Digital Signal Processing |
| DTMF | Dual-Tone Multi-Frequency |
| FFT | Fast Fourier Transform |
| HPS | Harmonic Product Spectrum |
| I/O | Input/Output |
| LCD | Liquid Crystal Display |
| LED | Light-Emitting Diode |
| PC | Personal Computer |
| PCI | Placa de Circuito Impresso |
| PWM | Pulse Width Modulation |
| SRAM | Static Random-Access Memory |

# SUMÁRIO

# 1 INTRODUÇÃO

O violão é um dos instrumentos musicais mais populares do mundo. Há décadas, a versatilidade do violão permite que músicos o utilizem em canções de rock, jazz, blues e inúmeros outros estilos (LIMA, 2013). Para tocar uma música agradável aos ouvidos, entretanto, é essencial que todos os instrumentos estejam perfeitamente afinados.

A afinação da corda de um instrumento consiste em girar uma tarraxa que regula a tensão mecânica da corda, assim aumentando ou diminuindo a frequência que a mesma produz ao ser tocada. O processo é repetido para todas as cordas – seis, no caso do violão (ROEDERER, 1998; BRAIN, 2014).

Este trabalho objetiva o desenvolvimento de um sistema de aquisição das frequências emitidas pelas cordas, para então compará-las com os valores correspondentes às notas musicais válidas e determinar o quão mais grave ou aguda está a afinação da corda. Assim, torna-se possível corrigir esta afinação fazendo uso de um atuador.

Um violão elétrico convencional tem em seu interior um sensor piezoelétrico ativo, alimentado por uma bateria de 9 V, ligado ao *jack* de saída do instrumento: um conector J10 (fêmea do conector P10) (LIMA, 2013). O sinal de saída deve ser condicionado para uma entrada analógica de uma placa microcontrolada.

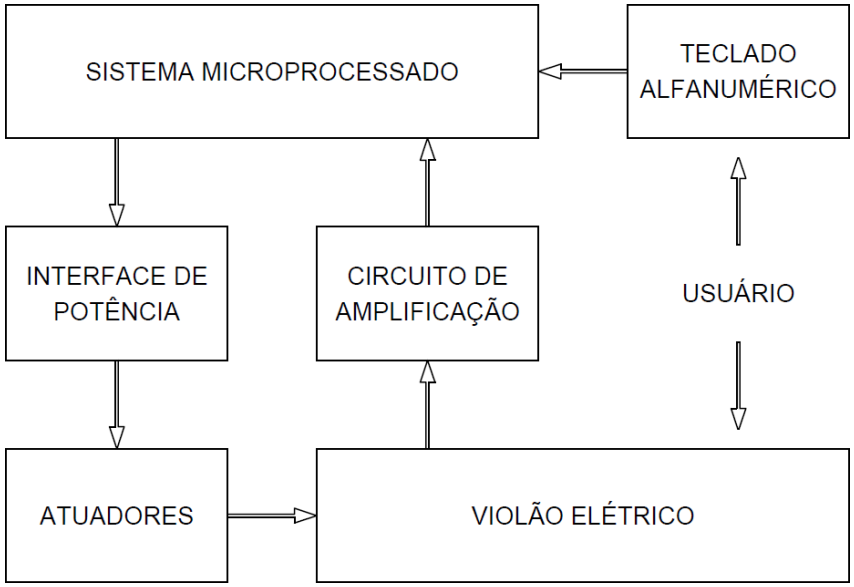
O sinal na entrada analógica do microcontrolador será processado por meio de técnicas de processamento digital de sinais (DSP, do inglês *Digital Signal Processing*). A frequência obtida deverá estar disponível para o sistema em formato digital, de

modo que o mesmo possa comparar estas informações com outras predeterminadas para tomar decisões e atuar diretamente nas tarraxas do violão, por meio de uma interface de potência.

Ao levar estes sinais à placa, também é possível compará-los com dados personalizados na memória do sistema, atribuindo uma nota específica a cada corda do violão conforme a afinação desejada, criando assim um banco de afinações para que o instrumento possa ser regulado de modos alternativos à afinação padrão, de acordo com o que pede a música ou o estilo musical.

A Figura 1 dispõe o sistema proposto em um diagrama de blocos.

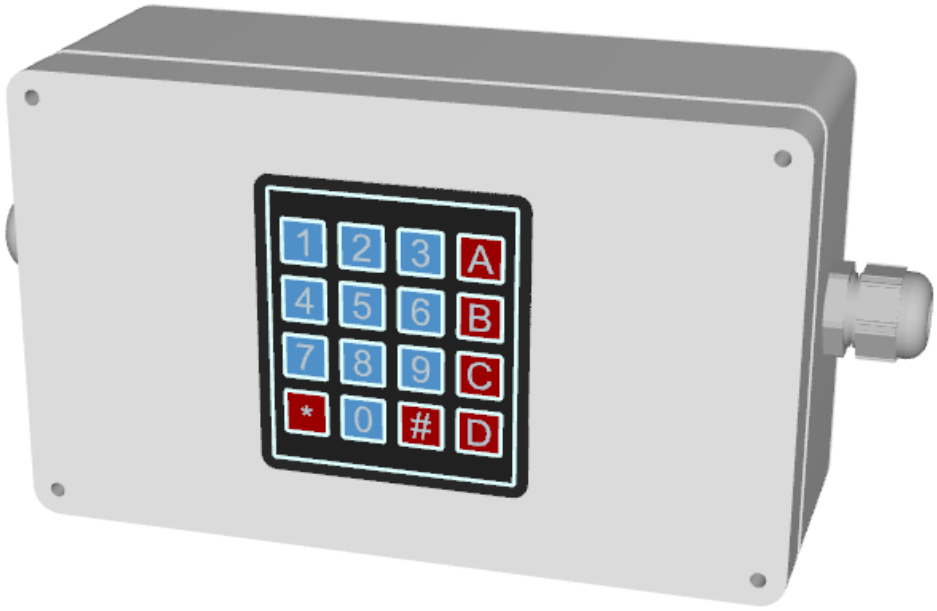
Figura 1 – Diagrama de blocos do sistema proposto




Fonte: Autor.

Os componentes são acomodados em uma caixa plástica de onde saem dois cabos, sendo um o cabo de áudio ligado ao violão e outro para o acionamento do motor. A alimentação é interna, por meio de bateria. Na tampa da caixa está alocado o teclado matricial autoadesivo de dezesseis teclas. A Figura 2 ilustra o equipamento.

Figura 2 – Sistema em caixa plástica ABS




Fonte: Autor.

O software desenvolvido é estruturado de acordo com as funções que o sistema deve executar. Foi criada uma estrutura para leitura de entradas do usuário, como a afinação desejada ou a corda a ser ajustada; uma estrutura para aquisição do sinal do violão condicionado por circuito apropriado; a estrutura principal, que ordena as demais; e uma estrutura de saída, que comanda a interface de potência.

Os experimentos práticos deste trabalho foram realizados com um violão da marca Tagima, série Memphis, modelo Folk MD 18, com cordas de aço. São avaliadas as características do sinal disponibilizado pelo instrumento para que possa ser feito o correto condicionamento e processamento do mesmo, além de análises da forma de onda gerada pelo som para que possa ser feita a interpretação adequada pelo microprocessador.

O Capítulo 2 apresenta a fundamentação teórica envolvendo o violão elétrico, aquisição e processamento de sinais de áudio, motores de passo, bem como trabalhos e produtos relacionados a esta ideia. No Capítulo 3 é descrito o desenvolvimento do sistema desde a primeira análise dos sinais provenientes do violão até o código desenvolvido para a aplicação. O Capítulo 4 exibe os resultados obtidos após o desenvolvimento do sistema. Por fim, no Capítulo 5 são apresentadas as conclusões e considerações finais.

# 2 REFERENCIAL TEÓRICO

Neste capítulo, são revisados conceitos gerais sobre música, contemplando notas musicais e frequências do som, violão elétrico e afinação, além da digitalização de um sinal de áudio. Também é abordada teoria acerca do condicionamento e do processamento destes dados, objetivando a manipulação de sinais responsáveis pela atuação. Por fim, são citados trabalhos desenvolvidos nesta mesma área e produtos comerciais disponíveis atualmente.

## 2.1 Música

Segundo Roederer (1998), a percepção da música está essencialmente baseada no processamento de informações acústicas. Constituída pela combinação de vários sons e ritmos, a música pode ser definida como uma sobreposição bem organizada de várias frequências. Embora sons em qualquer frequência possam fazer parte de uma música, os instrumentos musicais denominados “temperados” buscam reproduzir pontos específicos na faixa contínua de frequências audíveis, conhecidos como notas musicais.

Uma nota musical é um som emitido em uma frequência específica. Embora sejam conhecidas e pronunciadas como *dó*, *ré*, *mi*, *fá*, *sol*, *lá* e *si*, as notas musicais são representadas internacionalmente por símbolos, conforme a Tabela 1.

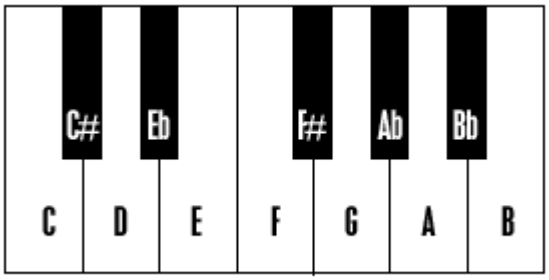
Tabela 1 – Nomes e símbolos das notas musicais

| Nome | Símbolo |
|---|---|
| Dó | C |
| Ré | D |
| Mi | E |
| Fá | F |
| Sol | G |
| Lá | A |
| Si | B |

Fonte: Autor.

Existem doze notas musicais. Além das sete fundamentais exibidas anteriormente, há outras cinco intermediárias, ou “acidentes”, caracterizadas pelo sufixo sustenido (#) ou bemol (*b*). A Figura 3 mostra a disposição das doze notas nas teclas de um teclado.

Figura 3 – Notas musicais em um teclado




Fonte: Mandaqui (2009).

Um conjunto de doze notas musicais é chamado de “oitava”. Uma oitava é caracterizada por ter cada uma de suas notas representando o dobro da frequência da mesma nota na oitava anterior, assim como a metade da frequência em relação às notas na próxima oitava. A frequência da nota lá, que é de 110 Hz em determinada oitava, será de 55 Hz na oitava anterior e de 220 Hz na próxima (ROEDERER, 1998).

Para diferenciar notas musicais em diferentes oitavas, pode ser atribuído um sufixo numérico à nota, iniciando-se em C1 e incrementando em uma unidade ao passar de um si (B) a um dó (C). Assim, após o G3 há o Ab3, e após o B3 há o C4. A nota musical C1, a mais grave, é produzida em 32,7 Hz.

Por convenção, adotou-se a nota lá da quarta oitava (A4, 440 Hz) como a nota a partir da qual se calcula a frequência das demais, seguindo a Equação 1 (VONK, 2015):

$$f = 2^{\frac{n}{12}} \cdot 440 \quad (1)$$

Onde:

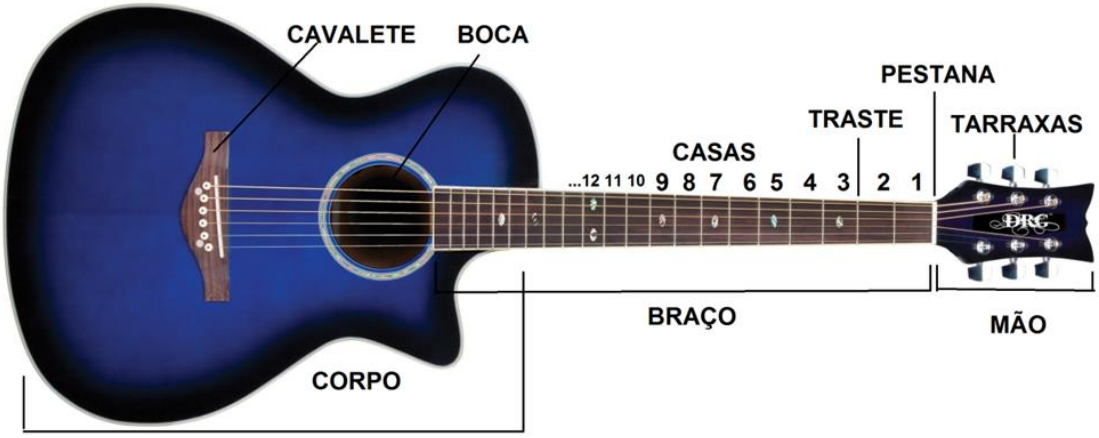
- ***f*** é a frequência da nota musical, em hertz;
- ***n*** é o número (positivo ou negativo) de notas entre a nota musical calculada (inclusive) a partir de A4.

## 2.2 Violão elétrico

O violão é um instrumento musical temperado da família das cordas. O modelo clássico, que representa a grande maioria dos instrumentos, possui seis cordas. O violão emite som a partir da vibração destas, sendo que cada corda emite uma nota musical específica relacionada à casa pressionada pelo músico no instante em que o acorde é tocado.

Esta vibração é amplificada pelo corpo oco do violão. Cada corda possui também uma espessura específica, assim emitindo notas musicais diferentes mesmo submetidas à mesma tração (BRAIN, 2014). As partes básicas de um violão, comuns a modelos acústicos e elétricos, são exibidas na Figura 4.

Figura 4 – Componentes de um violão convencional

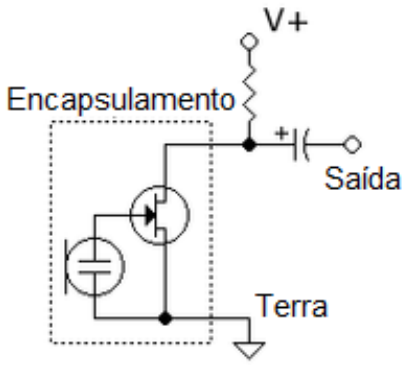



Fonte: Cervi (2013).

Internamente ao corpo, o violão elétrico possui um circuito eletrônico de captação ativa (FIGURA 5). O elemento piezoelétrico é fixado no corpo, próximo à boca do violão, enquanto o conector de saída encontra-se normalmente na parte inferior do mesmo. Próximo ao conector, está o *slot* para a bateria de 9 V que alimenta o circuito.

O sensor piezoelétrico é único para o violão, ou seja, não há a captação individual de cada corda, como no caso de uma guitarra (captação passiva indutiva). Seu funcionamento se dá convertendo a vibração da madeira do corpo do instrumento em tensão elétrica, que é então condicionada neste circuito interno, resultando em uma saída alternada sem *offset* de alguns milivolts.

Figura 5 – Circuito equivalente de um captador ativo




Fonte: Adaptado de Lima (2013).

Nota-se que a disposição das casas no braço do violão (FIGURA 4) segue um padrão exponencial, de acordo com a evolução das notas musicais apresentada no Item 2.1, valendo-se do princípio físico que diz que a frequência emitida por uma corda, entre outros fatores, é inversamente proporcional a seu comprimento útil (ROEDERER, 1998).

Percebe-se, também, que a 12ª casa tem seu traste localizado exatamente na metade do comprimento das cordas (entre a pestana e o cavalete) do instrumento, caracterizando o dobro da frequência em que cada uma vibra quando tocada solta, ou seja, à mesma nota musical, uma oitava mais aguda.

As seis cordas de um violão são numeradas da mais aguda para a mais grave. Na Tabela 2, são exibidas as notas musicais dentro de cada oitava e suas frequências, atribuídas a cada uma das seis cordas soltas do instrumento na afinação padrão.

Tabela 2 – Afinação padrão do violão

| Corda | Nota | Frequência [Hz] |
|---|---|---|
| 6ª | E2 | 82,4 |
| 5ª | A2 | 110,0 |
| 4ª | D3 | 146,8 |
| 3ª | G3 | 196,0 |
| 2ª | B3 | 246,9 |
| 1ª | E4 | 329,6 |

Fonte: Adaptado de Lourde R. e Saji (2009).

Além da afinação padrão, entretanto, é comum que o violão seja configurado em afinações alternativas, cada uma com características únicas que soam melhor em determinadas músicas. Uma das afinações alternativas mais comum é a “E bemol” (TABELA 3), que consiste em afinar cada corda uma nota musical mais grave.

Tabela 3 – Afinação “E bemol”

| Corda | Nota | Frequência [Hz] |
|---|---|---|
| 6ª | Eb2 | 77,8 |
| 5ª | Ab2 | 103,8 |
| 4ª | C#3 | 138,6 |
| 3ª | F#3 | 185,0 |
| 2ª | Bb3 | 233,1 |
| 1ª | Eb4 | 311,1 |

Fonte: Adaptado de Deambulatory (2010).

Outra configuração derivada da afinação padrão é a “Sexta em D” (TABELA 4), obtida com o ajuste da sexta corda em D2, mantendo as demais inalteradas.

Tabela 4 – Afinação "Sexta em D"

| Corda | Nota | Frequência [Hz] |
|---|---|---|
| 6ª | D2 | 73,4 |
| 5ª | A2 | 110,0 |
| 4ª | D3 | 146,8 |
| 3ª | G3 | 196,0 |
| 2ª | B3 | 246,9 |
| 1ª | E4 | 329,6 |

Fonte: Adaptado de Lourde R. e Saji (2009) e Deambulatory (2010).

Muito utilizada em canções dos Rolling Stones, a afinação “G aberto” (TABELA 5), como o nome sugere, consiste em afinar as cordas de modo que o violão produza um acorde G quando tocado sem nenhuma casa pressionada (GIBSON, 2014).

Tabela 5 – Afinação "G aberto"

| Corda | Nota | Frequência [Hz] |
|---|---|---|
| 6ª | D2 | 73,4 |
| 5ª | G2 | 98,0 |
| 4ª | D3 | 146,8 |
| 3ª | G3 | 196,0 |
| 2ª | B3 | 246,9 |
| 1ª | D4 | 293,7 |

Fonte: Adaptado de Lourde R. e Saji (2009) e Deambulatory (2010).

## 2.3 Aquisição de áudio

Da concepção do som à leitura pelo processador, a informação de áudio é transformada, condicionada, convertida e interpretada. Um som é uma vibração mecânica, e esta vibração é captada pelo transdutor acoplado ao instrumento, resultando em um sinal elétrico (BRAIN, 2014) que é então polarizado e amplificado para os níveis de tensão adequados ao processador.

Este sinal é contínuo no tempo, portanto, para que o processador possa interpretá-lo, é necessário que sejam tomadas amostras a cada determinado período.

Para realizar a interface entre sinais externos e o processador, este é acondicionado em uma placa, denominada, assim, placa microprocessada, que comporta circuitos digitais e analógicos e pinos de entradas e saídas, sendo que alguns destes pinos são capazes de ler um valor analógico e convertê-lo em um valor digital, realizando amostras do sinal de acordo com o que o processador estiver programado para realizar (PUHLMANN, 2015).

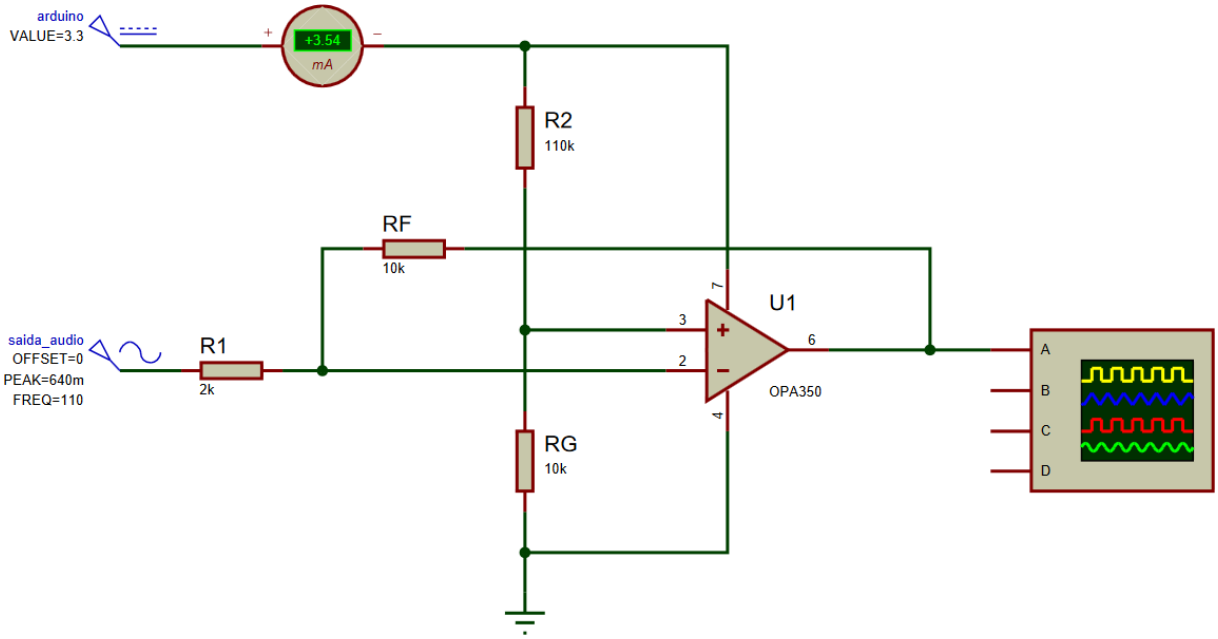
As seções a seguir descrevem o processo de aquisição de áudio, desde a geração até a interpretação do sinal: condicionamento, conversão analógico-digital e processamento.

### 2.3.1 Circuito de condicionamento

Condicionar os sinais de saída de áudio de um violão a um sistema microprocessado envolve determinar a amplitude máxima do sinal como a máxima tensão permitida no pino de entrada da placa e adicionar ao sinal um *offset* de metade desta tensão.

No circuito da Figura 6, é apresentado um amplificador operacional de modo *single supply* (dispensa a necessidade de fontes de tensão negativa), com alimentação de 2,7 V a 5,5 V, compatível com a maioria dos sistemas microprocessados (PUHLMANN, 2015). São adequados os modelos de amplificadores operacionais OPA350, LM358-N, AD8051A, AD8091, OP196S ou outros com características semelhantes (FARNELL, 2016), com tensões de operação entre 3 V e 5 V.

Figura 6 – Circuito de condicionamento de sinal




Fonte: Adaptado de Wendling (2010).

O circuito apresentado (FIGURA 6) contempla a ampliação do sinal de áudio em cinco vezes e a adição do *offset* na média da tensão fornecida pelo sistema microcontrolado utilizado. Para ajustar estes números, devem ser atribuídos novos valores aos resistores do circuito, de acordo com a Equação 2 (WENDLING, 2010):

$$V_{saída} = \left(\frac{R1 + RF}{R1}\right)\left(\frac{RG}{RG + R2}\right)V_{placa} - \frac{RF}{R1}V_{audio} \qquad (2)$$

Nota-se que a tensão $V_{audio}$ é invertida neste circuito, característica desta configuração de amplificação, o que não causa nenhum problema na aquisição.

### 2.3.2 Conversor analógico-digital

As grandezas físicas com as quais se deseja trabalhar raramente são de natureza elétrica. Por este motivo, existem sensores e transdutores. Entretanto, os sinais elétricos criados por estes elementos são contínuos no tempo, e é necessário que sejam amostrados para que possam ser processados digitalmente. Após o sinal do elemento primário ser adequadamente condicionado ao nível de tensão do processador, as leituras podem ser realizadas (PUHLMANN, 2015).

Cada conversor analógico-digital (ADC) possui uma resolução, ou seja, um número de bits com o qual será representada a variável analógica. Um ADC de dez bits, por exemplo, converte o nível de tensão do sinal lido em um número de 0 a 1.023. Este nível de tensão varia de acordo com o controlador, sendo os valores mais comuns 3,3 V, 5 V e 10 V (PUHLMANN, 2015).

A frequência de amostragem do sinal é determinada pelo processador (software) e deve ser adequada ao tipo de sinal com o qual se está lidando. Conforme Item 2.4, o teorema fundamental da amostragem de sinais analógicos para posterior processamento digital é o teorema da amostragem de Nyquist-Shannon.

### 2.3.3 Placas de processamento

Dentre os sistemas microprocessados disponíveis no mercado, os mais utilizados são plataformas microcontroladas como PIC e Arduino e sistemas operacionais embarcados que rodam em placas como Raspberry Pi e BeagleBone (SUEIRO, 2014). Apesar de os sistemas embarcados possuírem *clocks* de processamento com velocidades de até alguns gigahertz, esta característica não é determinante ao trabalhar com as baixas frequências produzidas no violão. Além disso, o sistema operacional (normalmente Linux) onde roda o programa pode causar atrasos nas aquisições de dados devido a processos paralelos (SUEIRO, 2014).

## 2.4 Processamento de áudio

Além da frequência fundamental, a corda de um violão produz harmônicas. São estas harmônicas que, somadas à fundamental, caracterizam o timbre único de cada instrumento (BRAIN, 2014). Estes sinais são contínuos no tempo, enquanto os microcontroladores funcionam em ciclos, isto é, amostrando o sinal a cada determinado período de tempo, armazenando-o em uma sequência discreta de valores.

O teorema da amostragem de Nyquist-Shannon diz que para que um sinal analógico seja adequadamente amostrado em meio digital, é necessário que a frequência de amostragem seja, ao menos, o dobro da maior frequência presente no sinal (PROAKIS; MANOLAKIS, 2007). Considerando a corda mais aguda do violão, que vibra em 330 Hz e harmônicas, por exemplo, até a sexta ordem, a máxima frequência de amostragem do sistema seria de ao menos 3,96 kHz.

Apesar de estes valores serem preliminares, percebe-se que a frequência de amostragem deve ser baixa para basicamente todos os microprocessadores disponíveis no mercado.

Há uma variedade de métodos de processamento do sinal adquirido para se determinar a frequência do mesmo. Abaixo são apresentados alguns dos mais utilizados: *zero-crossing*, autocorrelação e transformada discreta de Fourier, sendo que este último é mais comum em aplicações de áudio.

### 2.4.1 *Zero-crossing*

*Zero-crossing* é a técnica de detecção de frequência que consiste em analisar os pontos onde um sinal alternado passa do semiciclo positivo para o negativo e vice-versa, isto é, onde o sinal instantaneamente não apresenta tensão. Conhecendo os tempos onde isto ocorre e verificando padrões de repetitividade, é possível determinar o período total do mesmo e também as frequências fundamentais que o compõem (VONK, 2015).

Como instrumentos musicais podem ter harmônicas de amplitude maior que a própria frequência fundamental, este método não é o mais recomendado para música, sendo indicado para sinais formados por ondas senoidais mais uniformes, como tons DTMF (*dual-tone multi-frequency*).

### 2.4.2 Autocorrelação

De acordo com Oppenheim e Schafer (2010) e Vonk (2015), autocorrelação é a ferramenta matemática de detecção de repetição de padrões, assim sendo capaz de determinar quando um sinal se repete no tempo, obtendo seu período e, consequentemente, sua frequência. Para um sistema de afinação de cordas, a frequência fundamental é a única que deve necessariamente ser analisada, portanto este método poderia ser válido para tal aplicação. Porém, o som de um violão apresenta amplitude variável, que diminui a cada ciclo, conforme as cordas permanecem vibrando e atritando-se com o ar.

A autocorrelação de um sinal discreto pode ser expressa pela Equação 3 (OPPENHEIM; SCHAFER, 2010):

$$c_{hh}[l] = \sum_{k=-\infty}^{\infty} h[k]\, h[l+k] \tag{3}$$

Onde:

***h[n]*** é o sinal discreto;
***l*** é o período do sinal;
$c_{hh}$ é a autocorrelação.

### 2.4.3 Transformada discreta de Fourier

Segundo Nalon (2009), qualquer sequência discreta, salvo poucas restrições, pode ser representada como uma soma de senoides de frequências apropriadas. Conforme Antoniou (2005) e Oppenheim e Schafer (2010), sinais no tempo discreto podem ser caracterizados no domínio da frequência por sua transformada discreta de Fourier (DFT). Assim, o sinal pode ser representado por uma sequência de números, sendo o uso da DFT altamente recomendado para processamento em computadores e microcontroladores.

Mais complexa que a autocorrelação, a transformada discreta de Fourier retorna uma representação mais completa do sinal, informando a amplitude de cada componente do mesmo, considerando a frequência fundamental e as harmônicas mais influentes.

A transformada discreta de Fourier para amostras de um sinal contínuo se dá conforme a Equação 4 (DINIZ; SILVA; NETTO, 2002):

$$\boldsymbol{X}\left(\mathbf{e}^{j\frac{2\pi}{N}k}\right) = \sum_{n=0}^{N-1} \boldsymbol{x}(n)\, \mathbf{e}^{-j\frac{2\pi}{N}kn} \tag{4}$$

Onde:

***N*** é o número de amostras;

***k*** limita-se a ***0 ≤ k ≤ N-1***.

### 2.4.3.1 Algoritmo de Goertzel

O algoritmo de Goertzel, introduzido em 1958, é baseado na transformada discreta de Fourier, avaliando um componente específico da DFT. Este algoritmo varre as amostras do sinal realizando somente cálculos com valores reais e, ao final, executa apenas um cálculo com coeficiente imaginário, para então obter a resposta em frequência (LIMA, 2013). Por se limitar a alguns termos da frequência selecionada, este procedimento pode ser numericamente mais eficiente que as transformadas rápidas de Fourier (LYONS, 2012).

### 2.4.3.2 Transformada rápida de Fourier

Segundo Diniz, Silva e Netto (2002), o grande número de operações aritméticas envolvidas no cálculo da DFT, especialmente em sequências longas, pode limitar seu uso prático. Assim, foram desenvolvidos algoritmos mais eficientes para realizar estes cálculos, conhecidos como transformadas rápidas de Fourier (FFT), introduzidas em

1965 por Cooley e Tukey. Desde então, a DFT tem sido amplamente utilizada em processamento digital de sinais, fazendo uso das FFT (OPPENHEIM; SCHAFER, 2010). Esta técnica retorna a resposta em frequência de forma mais completa que o algoritmo de Goertzel, sendo, portanto, mais confiável.

Existem diversos algoritmos de transformada rápida de Fourier, sendo que as respostas podem ser mais ou menos precisas dependendo das características do sinal de entrada (SMITH, 2011).

#### 2.4.3.3 Teorema de Parseval

O teorema de Parseval-Plancherel, ou simplesmente teorema de Parseval, refere-se à unicidade do sinal, afirmando que a potência média de um sinal discreto é igual à soma das potências médias de seus componentes de Fourier. Ou seja, retorna um número correspondente à energia do sinal, e pode ser expresso pela Equação 5 (ROBERTS, 2010):

$$\frac{1}{L}\int_{L} |x(t)|^2 dt = \sum_{n=-\infty}^{\infty} |X(k)|^2 \qquad (5)$$

Assim, ao realizar o cálculo do teorema de Parseval para todas as amostras do sinal, é possível determinar sua potência como um todo. Da mesma maneira, ao realizar este cálculo utilizando apenas uma frequência de interesse (possivelmente incluindo algumas harmônicas), é possível determinar a contribuição desta potência ao sinal (ROBERTS, 2010).

Comparando com a potência total do mesmo, obtém-se uma proporção que representa a razão da intensidade de dada frequência e componentes relacionados, determinando a pureza e o nível de ruído do sinal, para então validá-lo ou não como representação confiável da entrada.

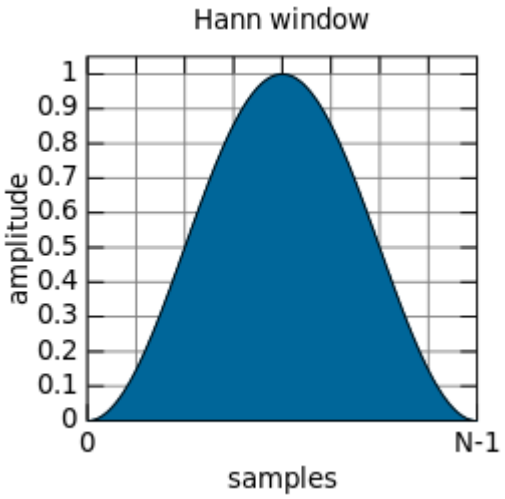
### 2.4.4 Função janela

O janelamento de sinais atua como um filtro digital e permite que operações como a transformada discreta de Fourier retornem resultados mais confiáveis (WEISSTEIN, 2002).

Uma função janela aplicada a um sinal discreto opera com os valores em um determinado intervalo, zerando todos os valores fora deste. Uma janela retangular, por exemplo, é uma janela de delimitação apenas, multiplicando por 1 os valores dentro de um intervalo e por zero os valores fora deste.

Em análise de sinais, é comum o uso de janelas com formas de sino, conforme Figura 7. As janelas mais utilizadas, além da retangular, são Hann (FIGURA 7), Hamming, Blackman, Kaiser, gaussiana, entre outras, além de combinações entre mais de um tipo. As janelas citadas são semelhantes entre si, sendo que a aplicação de cada uma se dá de acordo com as propriedades do sinal (WEISSTEIN, 2002).

Figura 7 – Janela de Hann




Fonte: Adaptado de Weisstein (2002).

A Janela de Hann, acima, é uma das mais utilizadas para aplicações genéricas, iniciando-se em zero, com pico em 1 à metade da amostragem, e finando-se em zero novamente.

## 2.5 Motor de passo

Elucidados os conceitos acerca da aquisição e processamento de sinais de áudio, esta seção apresenta a fundamentação teórica acerca do atuador que pode ser utilizado para, de fato, afinar um violão elétrico com base nessas informações.

Motor de passo é um dispositivo eletromecânico que converte pulsos elétricos em movimentos angulares discretos, podendo ser utilizado em aplicações onde é necessário controlar fatores como rotação, velocidade e posição. Normalmente, um motor de passo possui quatro bobinas que, ao serem energizadas alternadamente, causam o movimento de rotação do eixo, que busca alinhar-se ao campo magnético criado por cada bobina. Ao inverter a ordem de energização das mesmas, o sentido de rotação do eixo do motor torna-se o contrário (BRITES, 2008).

Motores de passo possuem boa precisão e resposta rápida a comandos de aceleração e desaceleração, porém, não atingem altas velocidades e não respondem adequadamente a inércia mecânica. Portanto, seu uso é recomendado onde posicionamento é uma variável mais prioritária em relação a velocidade e carga, ou seja, em aplicações de controle (BRITES, 2008).

A sequência de excitação das bobinas de um motor de passo se dá de modo comparável a qualquer outro motor, porém digitalmente. Isto é, ora a tensão aplicada em uma bobina é total, ora é zero. Podem ser excitadas uma ou mais bobinas a cada passo, permitindo diferentes formas de controle do atuador (ELETRO, 2013).

O modo mais simples de acionar o dispositivo é através da excitação de uma bobina a cada passo, mantendo as demais desenergizadas. Assim, uma sequência de atuação se dá em quatro passos. Também podem ser acionadas duas bobinas adjacentes simultaneamente em cada operação, gerando um aumento de torque de 30% a 40%, entretanto, consumindo o dobro de corrente. Da mesma forma, um ciclo de operação se dá em quatro passos (ELETRO, 2013). Este tipo de operação é denominado “passo completo”.

Ainda é possível combinar os acionamentos simples e simultâneos em um modo de controle chamado “meio passo”. Neste caso, a corrente drenada é maior que

nos demais, e a velocidade também é reduzida, porém, a precisão é maior. São necessários oito passos para completar um ciclo de atuação (ELETRO, 2013).

## 2.6 Trabalhos relacionados

Principalmente na última década, foi realizada uma variedade de estudos e trabalhos, por diversos autores, que abordam temas relacionados ao uso do processamento digital de sinais com o objetivo de interpretar frequências de som ou mesmo afinar um instrumento.

Lourde R. e Saji (2009) propuseram um sistema de afinação motorizada de violão. Foram desenvolvidos algoritmos no software MATLAB utilizando FFT e a análise do espectro de frequências para determinar a fundamental. Segundo as autoras, esta análise é necessária pois muitas vezes uma frequência harmônica tem amplitude maior que a fundamental. Embora teórico, o artigo apresenta estudos sobre a atuação nas tarraxas de um violão, prevendo que um giro de 180° na sexta corda altera a frequência desta em 4 Hz, enquanto na primeira corda, a mesma atuação provoca uma alteração de cerca de 15 Hz. O trabalho não foi testado em microcontrolador.

Code (2009) desenvolveu um programa em C para determinar a frequência fundamental de um fragmento de áudio capturado. Foi utilizado o algoritmo de FFT de Cooley-Tukey para adquirir este valor e compará-lo com uma tabela de frequências correspondentes a notas musicais válidas.

Deambulatory (2010) apresenta um algoritmo de detecção de frequência baseado em autocorrelação desenvolvido para o Arduino Uno. De acordo com o autor, o código, um tanto complexo, apresentou bons resultados na estimativa da frequência fundamental de um sinal.

Ghassaei (2012) publicou no *website* Instructables algumas técnicas de processamento de áudio de diferentes formatos de onda para plataformas Arduino, incluindo um circuito condicionador para entradas analógicas de zero a 5 V. A partir desta publicação, Grimwood (2012) desenvolveu um afinador tradicional de guitarra

elétrica, utilizando sete sinais luminosos para indicar o quão mais grave ou aguda está a corda que se deseja afinar, na afinação padrão.

Salvi (2013) apresenta fundamentos para um sistema de análise de áudio genérico para Arduino. São exibidas diversas técnicas de condicionamento do sinal, envolvendo amplificadores operacionais, transistores e circuitos específicos para microfones de eletreto. O autor também publicou no *website* Instructables sugerindo uma matriz luminosa que responde de acordo com o som como saída de seu sistema.

Lima (2013) propõe um sistema de afinação para violão acústico, fazendo uso de um microfone de eletreto, que apresenta um circuito semelhante ao sensor piezoelétrico de um violão elétrico, porém com saída de cerca de 20 mV. O autor desenvolveu um programa no software MATLAB baseado no algoritmo de Goertzel. O sistema foi desenvolvido de modo a fazer a leitura da nota musical tocada e compará-la com uma das seis frequências predeterminadas da afinação padrão. Com o auxílio do software de afinação AP Tuner, o código foi validado no MATLAB. Porém, ao importar o software para o Arduino, o mesmo não apresentou resultados satisfatórios. De acordo com o autor, o sistema não funcionou adequadamente no Arduino provavelmente por limitações de memória e de velocidade de resposta da placa.

Vonk (2015) utilizou algoritmos de autocorrelação em seu sistema de detecção de frequência de um som. O programa foi testado com sons de piano e clarinete, e apresentou boas respostas na faixa média de frequências. Os sons mais agudos foram detectados incorretamente, como se fossem uma oitava mais graves, o que não inviabiliza a aplicação do mesmo ao violão, que não apresenta tais notas musicais. Porém, para algumas notas da segunda oitava, encontradas nas sexta e quinta cordas do violão, o sistema não apresentou resposta.

## 2.7 Produtos existentes

Afinadores simples de instrumentos existem nas mais variadas formas. Há afinadores onde é conectado o cabo da guitarra ou do contrabaixo, afinadores de

violão presos por pressão para detectar a vibração ou mesmo incorporados na lateral do violão, e afinadores a microfone, que detectam notas musicais de diversos instrumentos. Como se tratam de produtos comerciais, o modo de funcionamento dos mesmos não é aberto.

Afinadores automáticos também podem ser encontrados no mercado, como o Roadie Tuner, composto por um motor que o músico acopla à tarraxa no momento da afinação (BAND, 2017), e o Min-ETune, composto de seis motores fixos nas tarraxas do violão ou da guitarra (EPIPHONE, 2017).

A proposta deste trabalho é desenvolver passo a passo um sistema de afinação automática onde todos os componentes, inclusive software, possam ser acessados e modificados pelo usuário.

# 3 DESENVOLVIMENTO

Basicamente, o sistema deve consistir em adquirir amostragens da tensão elétrica produzida pelo elemento piezoelétrico do violão, condicionada em circuitos apropriados, e processá-lo de modo a obter um valor numérico correspondente à frequência tocada. O sistema também deve receber informações do usuário que indicam em qual afinação este deseja regular o instrumento e qual corda será tocada para que seja afinada, determinando assim em qual frequência esta deveria soar.

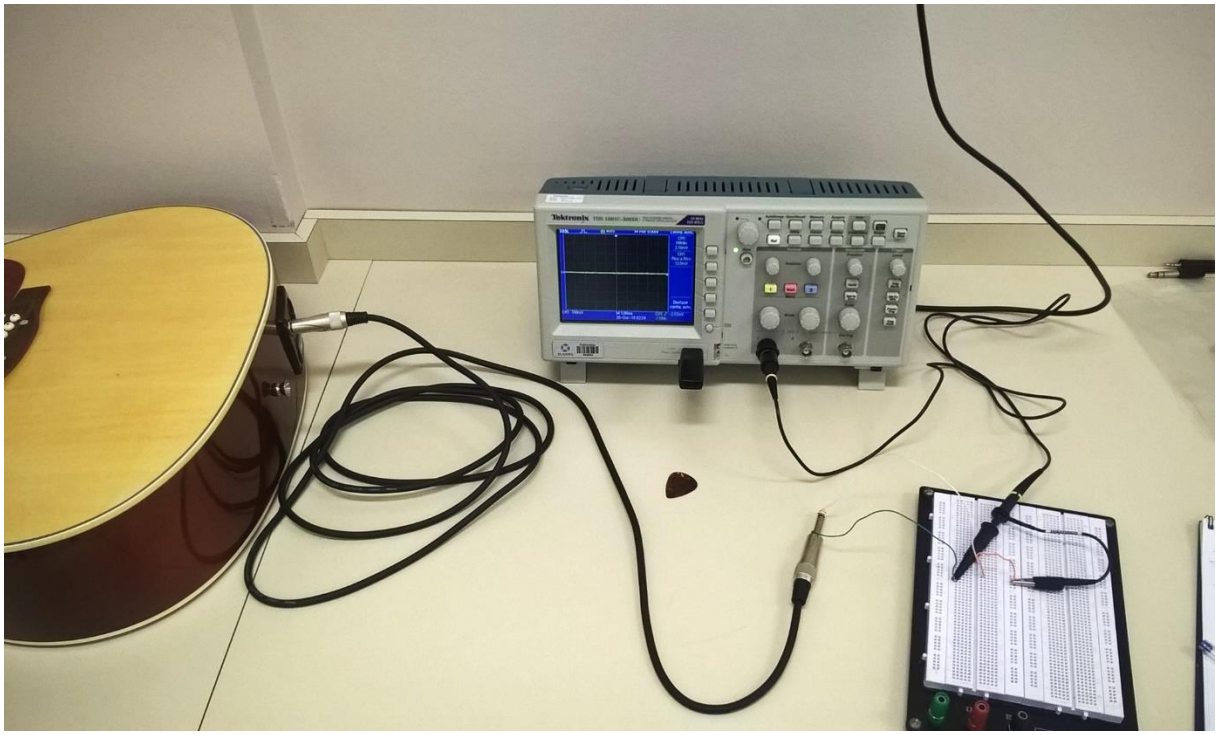
Sabendo se a frequência da corda é mais grave, igual ou mais aguda que a frequência desejada, o sistema deve comutar saídas conectadas por uma interface de potência ao atuador acoplado ao violão com o objetivo de afiná-lo.

Antes de a instalação do sistema ser iniciada, os componentes foram montados e testados individualmente, assim permitindo a configuração específica de cada parte e uma detecção de erros mais direcionada. Além disso, qualquer ajuste necessário ao sistema pudera ser realizado antes de feita a montagem definitiva de elementos, como a fabricação de uma placa de circuito impresso.

## 3.1 Análise dos sinais

Antes de conectar o violão ao circuito de condicionamento e ao Arduino (ITEM 2.3.1), foram realizados alguns testes em osciloscópio (FIGURA 8), para determinar com mais exatidão com qual tipo de sinal se estaria trabalhando. Percebe-se que o sinal não apresenta um *offset*, isto é, possui apenas o componente alternado.

Figura 8 – Osciloscópio conectado ao violão

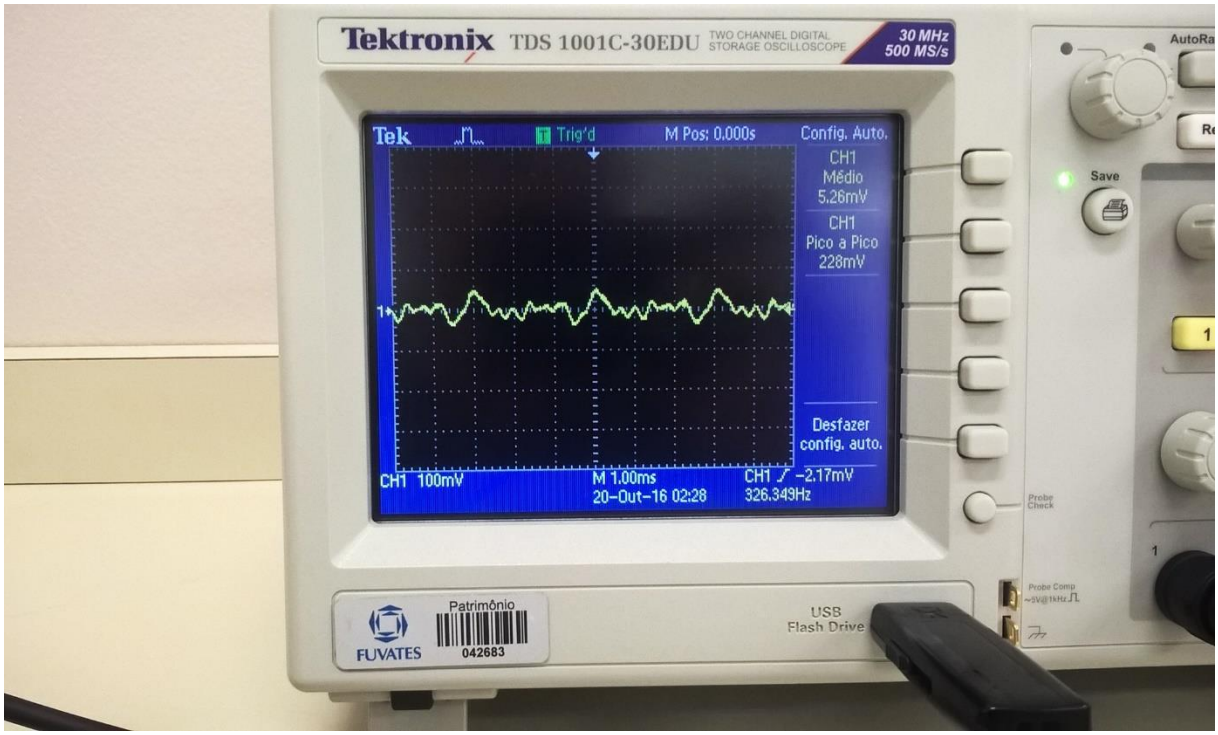



Fonte: Autor.

Verificou-se uma variação na amplitude máxima dos sinais, que passou de valores de 260 mV pico a pico quando é tocada a sexta corda, a mais grave (FIGURA 9) para até 640 $mV_{pp}$ no caso da primeira corda, a mais aguda (FIGURA 10), dependendo também da intensidade com a qual as mesmas são tocadas.

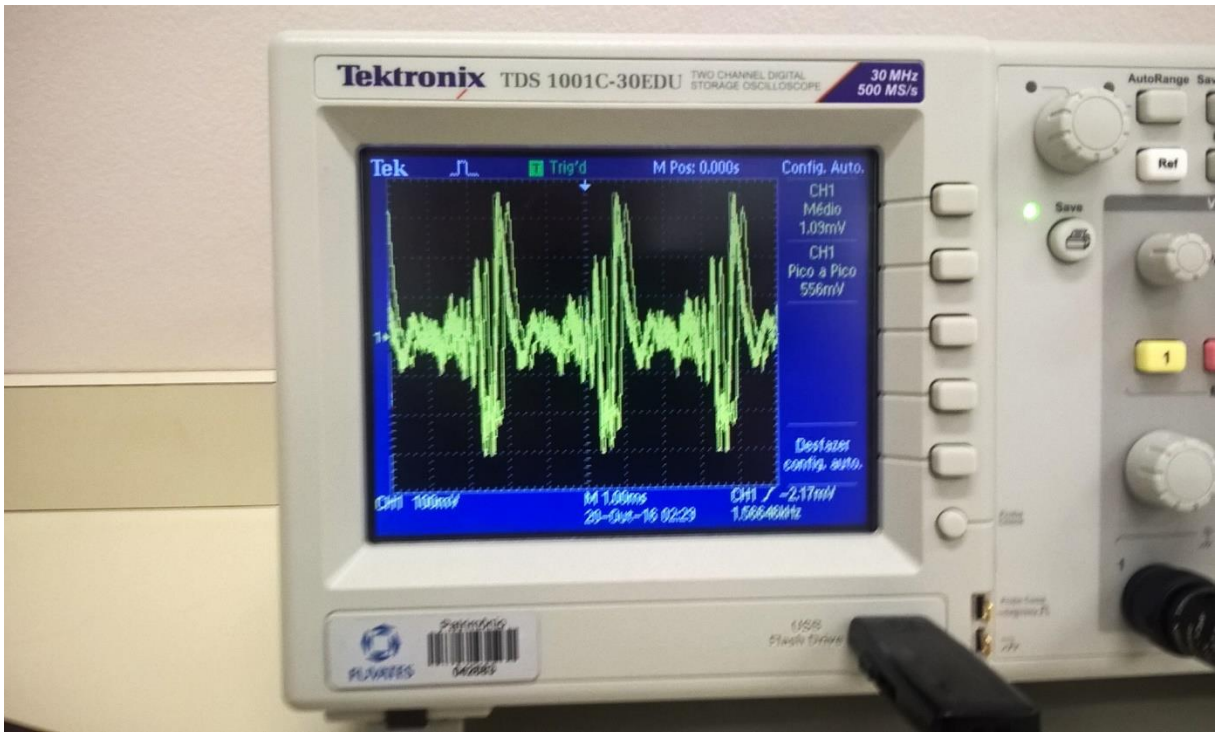
Após ajustes na equalização do instrumento, as magnitudes normalizaram-se um tanto, mas as cordas mais agudas continuam a gerar maiores amplitudes.

Figura 9 – Resposta à sexta corda (E2)




Fonte: Autor.

Figura 10 – Resposta à primeira corda (E4)



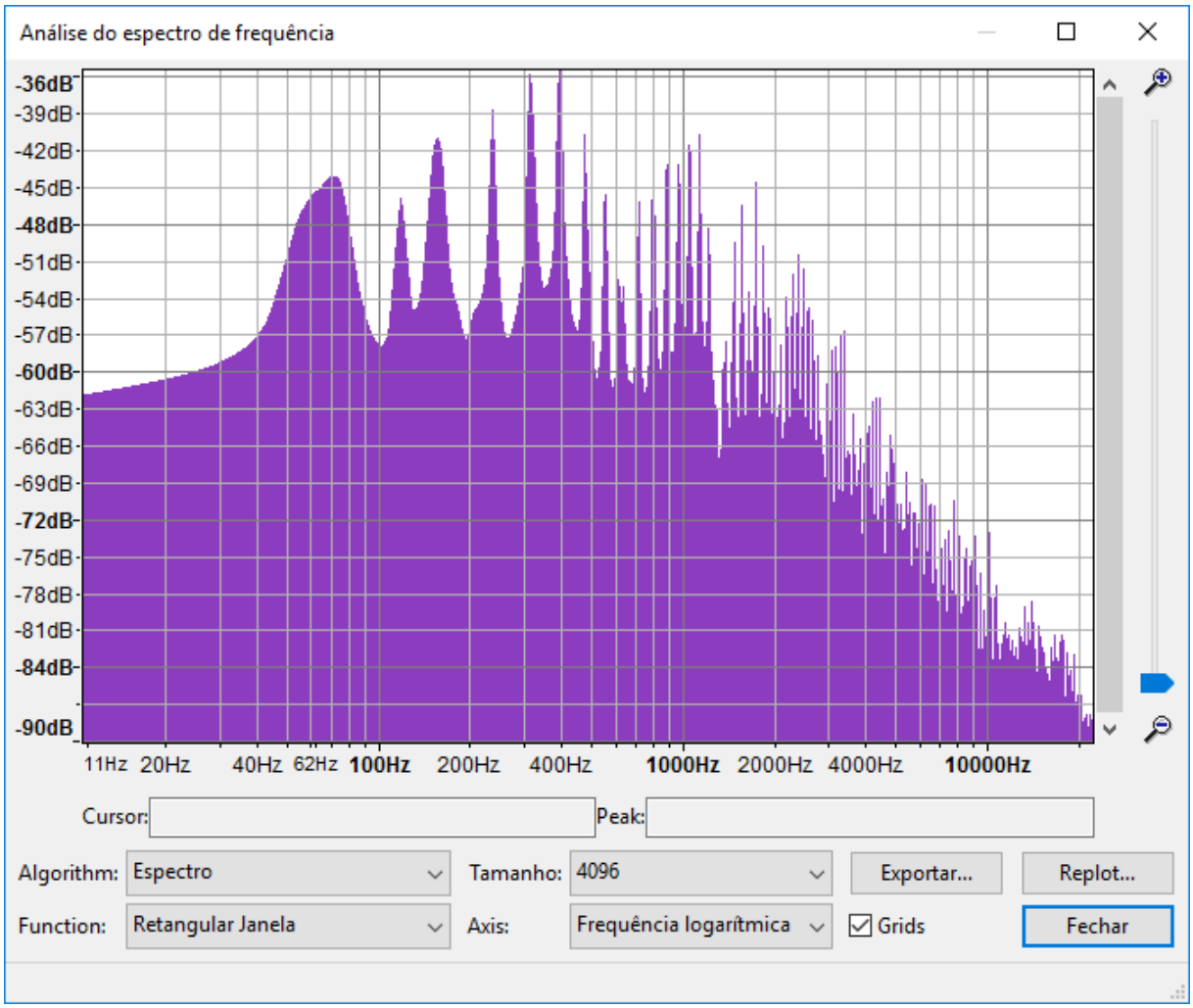

Fonte: Autor.

O osciloscópio utilizado é da marca Tektronix, modelo TDS 1001C-30EDU (TEKTRONIX, 2016).

### 3.2 Análise de som

Para ajudar a determinar qual o melhor método de aquisição da frequência, foram analisadas algumas amostras do som do violão conectado a um computador pessoal (PC) fazendo uso do software Audacity 2.1.2 (AUDACITY, 2016). Na Figura 11 é apresentado o espectro de frequências da nota Eb2, cuja fundamental é 77,8 Hz, ainda sem a implementação de qualquer filtro digital.

Figura 11 – Espectro de frequência da nota Eb2



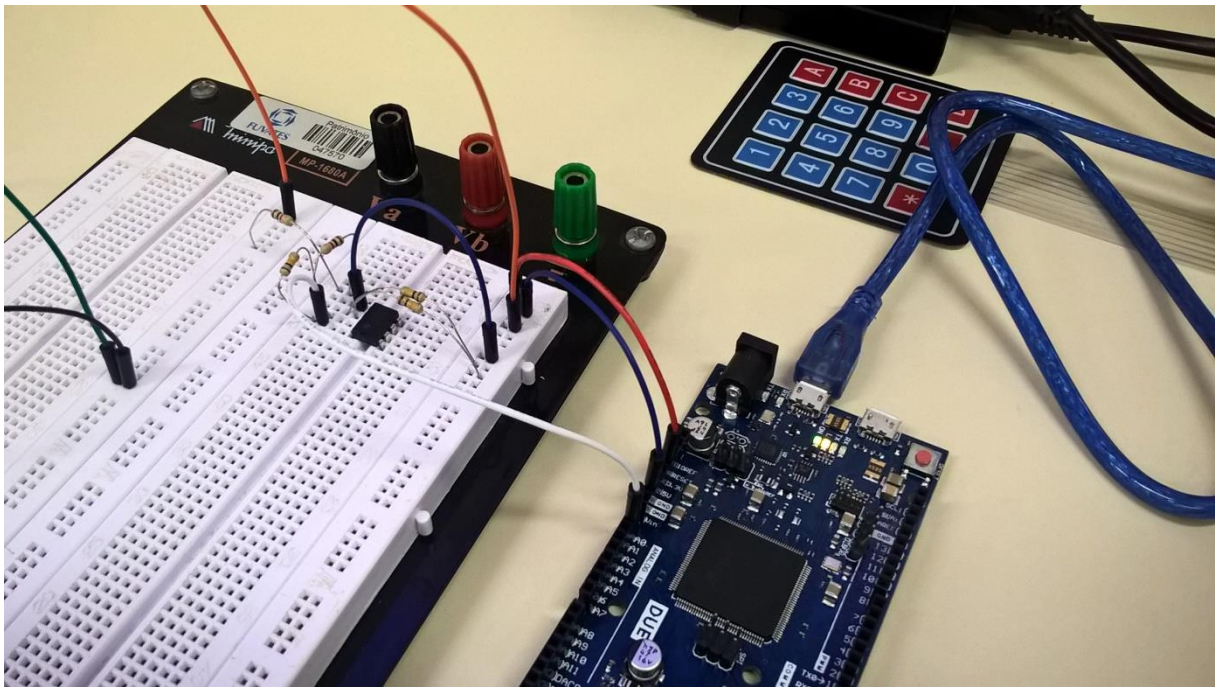

Fonte: Autor.

Mesmo com uma análise preliminar, nota-se que algumas harmônicas têm amplitude maior que da frequência fundamental, confirmando assim que métodos simples como *zero-crossing* não são adequados para a análise deste tipo de sinal.

### 3.3 Polarização e amplificação do sinal

O circuito proposto no Item 2.3.1 foi esquematizado e simulado no software Proteus ISIS Professional V7.8 e, após obtidos os resultados esperados, montado em placa de ensaio (FIGURA 12).

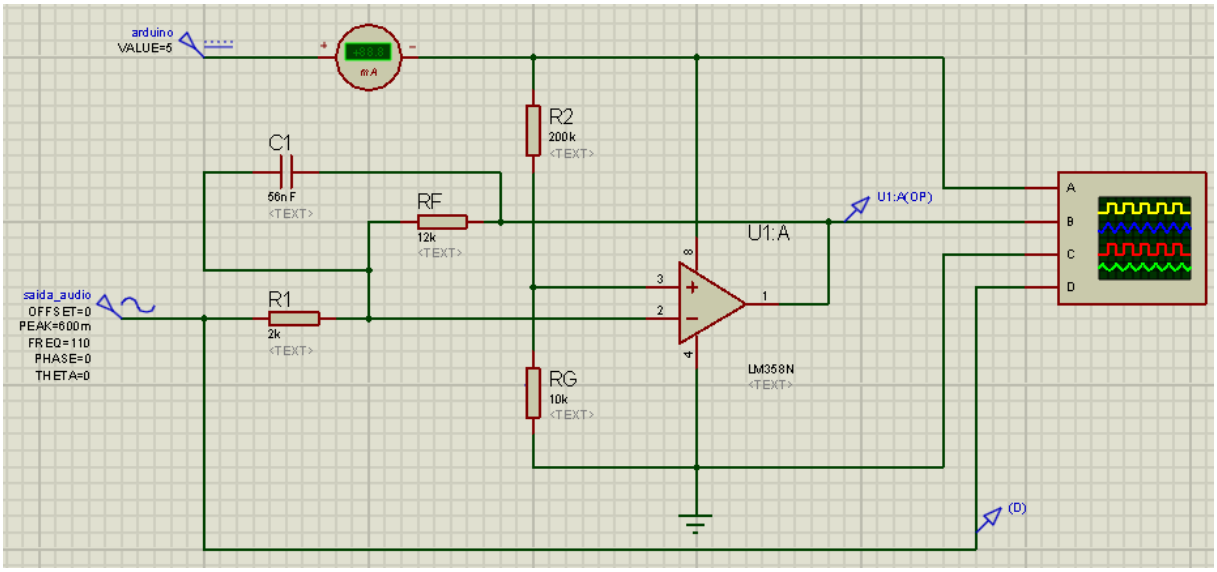
Figura 12 – Circuito de condicionamento em placa de ensaio



Fonte: Autor.

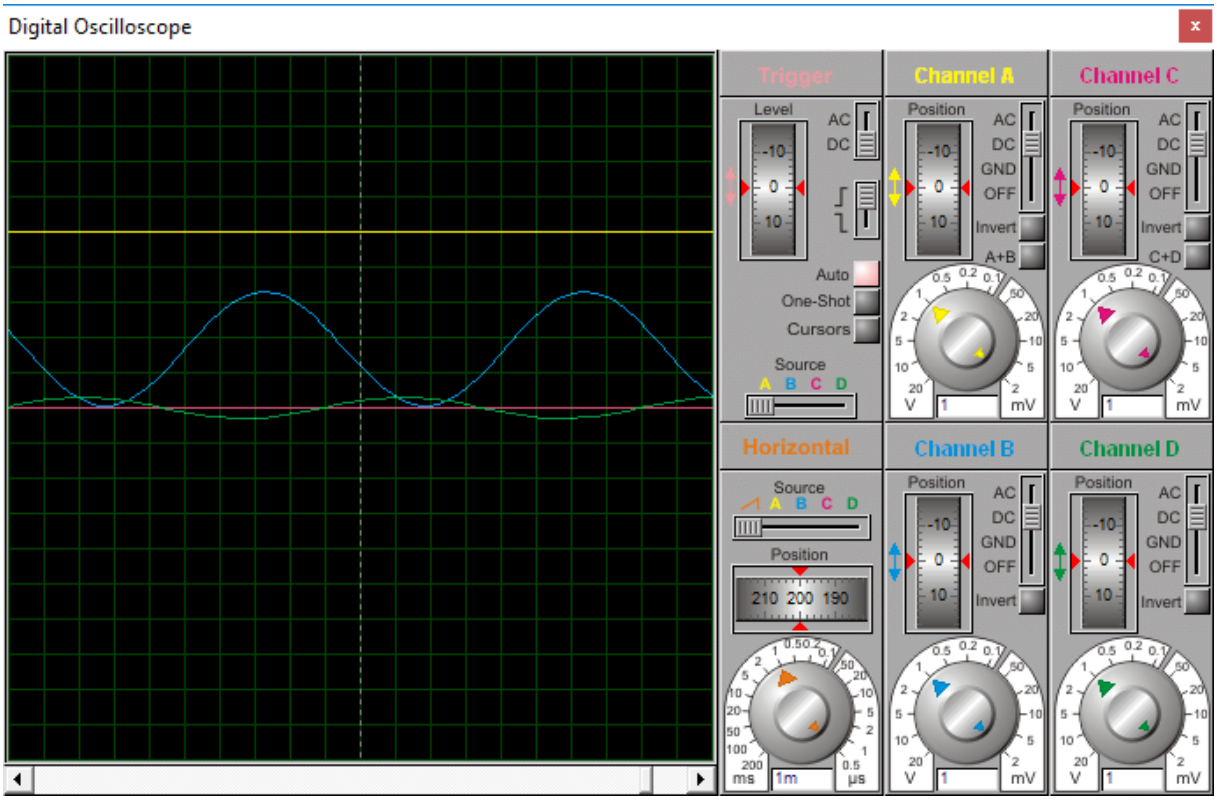
Porém, na ocasião da aquisição do amplificador operacional, o único modelo disponível com as características desejadas foi o LM358-N. Este circuito integrado possui uma configuração *darlington* na saída, o que causa a queda da tensão máxima da saída e consequente saturação do sinal. Portanto, foi utilizado o pino de 5 V do Arduino para que o sinal não saturasse antes dos 3,3 V. Assim também, os valores de resistores tiveram de ser alterados (FIGURA 13) de acordo com a Equação 2 e o circuito foi novamente simulado no software (FIGURA 14) e em placa de ensaio.

Figura 13 – Circuito de aquisição




Fonte: Autor.

Figura 14 – Simulação em software

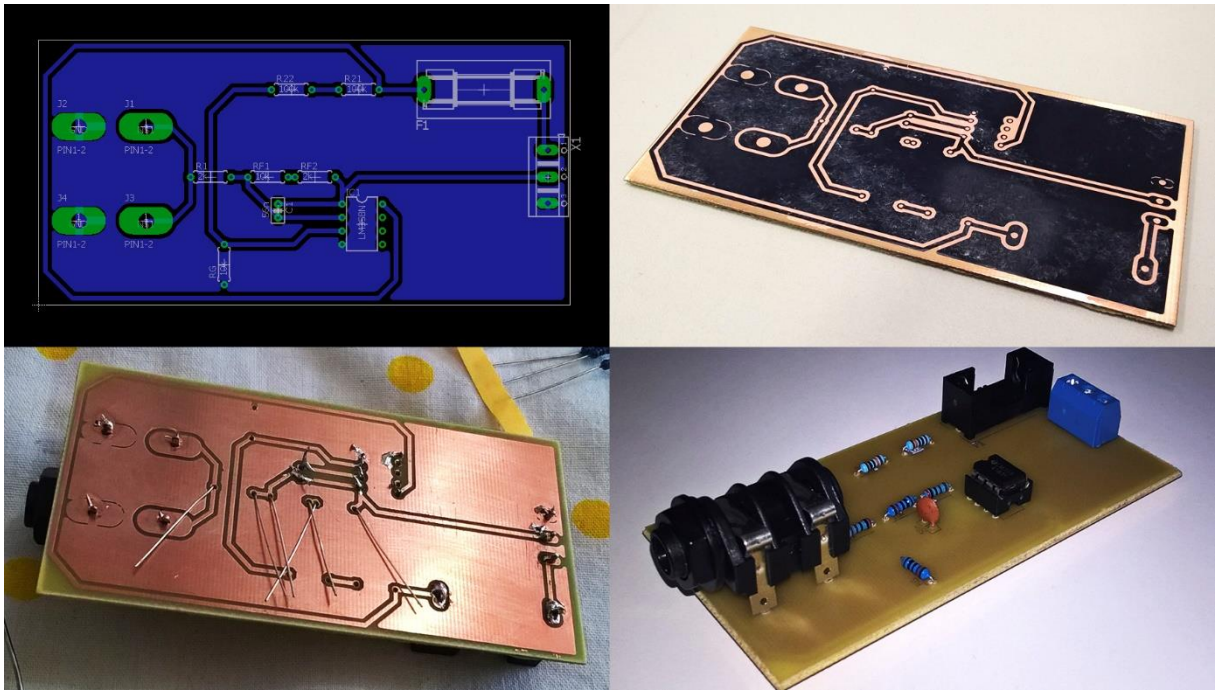



Fonte: Autor.

Uma vez montado, o circuito foi testado com o violão Tagima, mostrando respostas adequadas ao ser excitado pelas frequências e amplitudes produzidas pelas cordas do instrumento, assim como os valores lidos pelo microcontrolador foram satisfatórios.

Então, foi fabricada a placa de circuito impresso (PCI) em placa virgem de fibra de vidro, com o uso de resistores de precisão, conforme exibe a Figura 15. Esta placa foi desenvolvida no software Eagle 7.7.0, e apresentou resultados semelhantes ao circuito em placa de ensaio, com um *offset* de saída ligeiramente mais próximo à metade da tensão máxima de leitura – uma característica desejável.

Figura 15 – Placa de circuito impresso fabricada



Fonte: Autor.

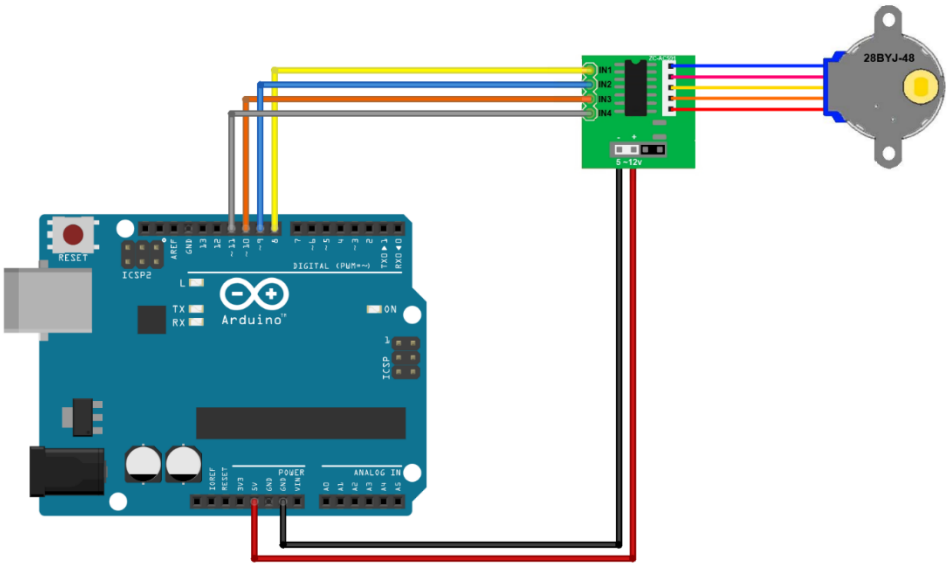
### 3.4 Atuador

O motor de passo implementado para girar automaticamente as tarraxas do violão no sentido horário ou anti-horário, de acordo com a situação da afinação da corda, é o modelo 28BYJ-48, da Kiatronics, de 5 V (KIATRONICS, 2017). O driver utilizado para interfacear o Arduino com a potência é o modelo ULN2003, da Texas Instruments, de até 500 mA (TEXAS, 2017). Este driver é fornecido em placa de circuito impresso apropriada, com todos os conectores necessários.

Assim, são disponibilizados quatro pinos para o comando do motor e mais dois para a alimentação do driver de potência (FIGURA 16). Para aumentar o alcance do

motor, é utilizado um cabo de seis fios tipo manga com um conector de 5 pinos (fêmea) ligado aos terminais do motor, em uma das pontas, e outro conector de 5 pinos (macho) ligado ao driver, na outra extremidade.

Figura 16 – Ligação da placa ao motor



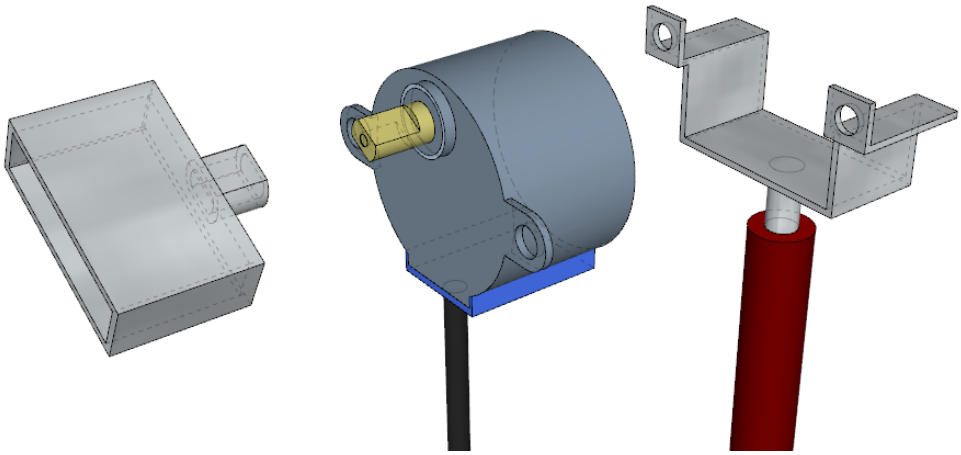

Fonte: Adaptado de Filipe (2013).

Devido às características deste projeto, foi determinado o acionamento do motor em passo completo, energizando uma bobina a cada operação, gerando um ciclo de quatro passos, conforme Item 2.5.

Foi utilizada a biblioteca `stepper.h` disponibilizada pelo próprio fabricante para controlar o motor. Para este motor realizar 360° de rotação, levando em consideração a redução mecânica interna de 1/64, são necessários 4.096 passos. Com esta redução, o eixo de acoplamento perde velocidade, mas tem aumento em torque e retenção (força), características mais desejadas.

Primeiramente, o motor foi conectado à tarraxa da sexta corda do violão (mais espessa) através de um acoplamento mecânico (FIGURA 17) para determinar a velocidade na qual o mesmo poderia girar sem patinar e nem causar sobrecorrente no driver. Foi fixada a velocidade de 60%, e então os testes nas demais cordas foram realizados, com sucesso. A corrente fornecida ao motor jamais passara de 400 mA.

Figura 17 – Estrutura mecânica em torno do atuador



Fonte: Autor.

O próximo experimento foi determinar o número de passos necessários para alterar a frequência de uma corda em 1 Hz. Foi utilizado um afinador comercial no violão e então o motor foi posto a atuar, com os resultados variando conforme a corda, de acordo com a Tabela 6.

Tabela 6 – Número de passos por hertz para cada corda

| Corda | Nota | Passos/Hz |
|---|---|---|
| 6ª | E2 | 144 |
| 5ª | A2 | 153 |
| 4ª | D3 | 161 |
| 3ª | G3 | 167 |
| 2ª | B3 | 173 |
| 1ª | E4 | 178 |

Fonte: Autor.

### 3.5 Placa microcontrolada

Como componente central do sistema, foi escolhido o Arduino Due R3, cujo *clock* é de 84 MHz, principalmente por possuir processamento em 32 bits, enquanto seu custo não é muito superior ao modelo mais simples da linha, o Arduino Uno, de 16 MHz e oito bits (TABELA 7). Uma das vantagens do modelo Due é trabalhar com 32 bits, o que proporciona melhor desempenho ao realizar as operações aritméticas envolvidas no cálculo das DFT – que utilizam números complexos, formados por uma

combinação de dois valores armazenados em formato *float* (ponto flutuante) de 32 bits, sendo um para a parte real do número e outro para a parte imaginária.

O modelo possui doze saídas analógicas controladas por modulação por largura de pulso (PWM) e seus conversores analógico-digital têm resolução de dez bits. Além disso, é o modelo que possui maior quantidade de memória estática de acesso aleatório (SRAM) e memória flash.

Tabela 7 – Comparativo entre placas Arduino

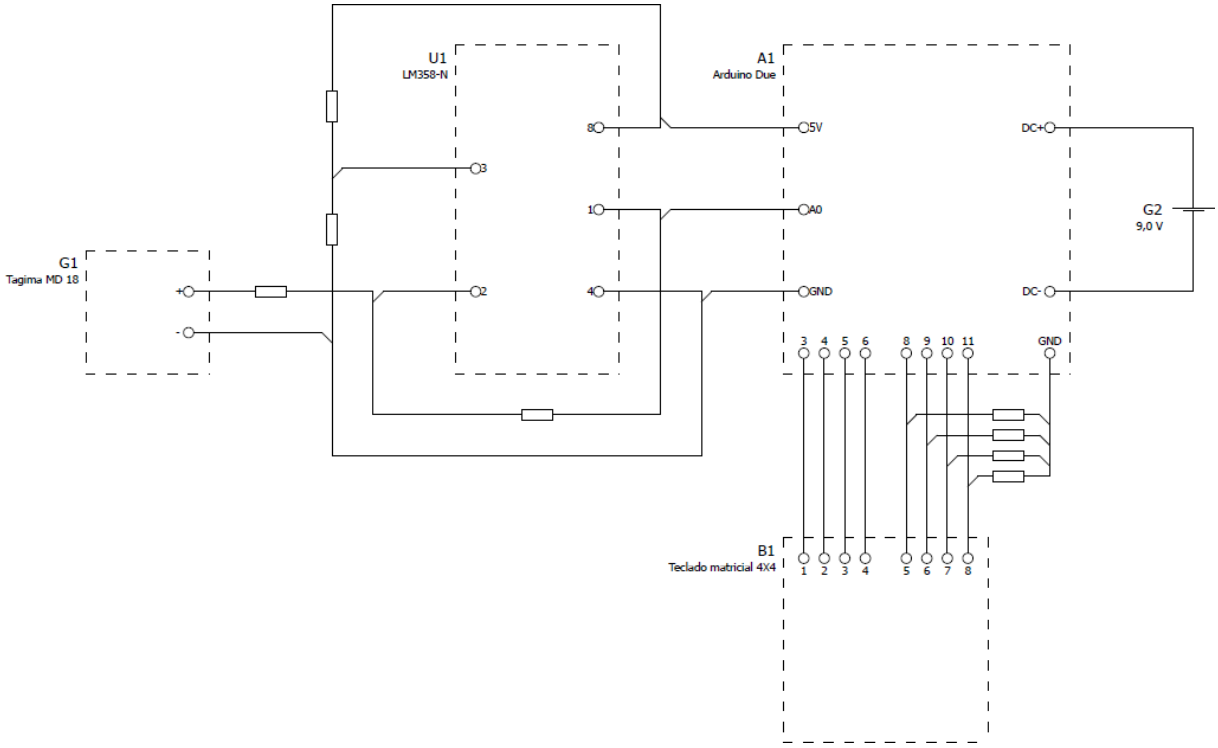
| | **UNO** | **LEONARDO** | **MEGA** | **101** | **ZERO** | **DUE** |
|---|---|---|---|---|---|---|
| Pinos digitais | 14 | 20 | 54 | 14 | 20 | 54 |
| Saídas PWM | 6 | 7 | 15 | 4 | 18 | 12 |
| Entradas analógicas | 6 | 12 | 16 | 6 | 6 | 12 |
| Tensão operação [V] | 5 | 5 | 5 | 3,3 | 3,3 | 3,3 |
| Corrente máx. [mA] | 50 | 50 | 50 | 280 | 140 | 800 |
| Frequência [MHz] | 16 | 16 | 16 | 32 | 48 | 84 |
| Processador [bits] | 8 | 8 | 8 | 32 | 32 | 32 |
| Memória flash [kB] | 32 | 32 | 256 | 196 | 256 | 512 |
| Memória SRAM [kB] | 2 | 2,5 | 8 | 24 | 32 | 96 |

Fonte: Arduino (2016).

O Arduino Due opera em 3,3 V e pode fornecer um total de até 800 mA aos componentes conectados em saídas reguladas de 3,3 V e 5 V. Possui doze entradas analógicas, com 1.024 intervalos de resolução de 3,22 mV cada. A alimentação da placa deve ser de 7 V a 12 V (ARDUINO, 2016) – para evitar ruídos da rede possivelmente causados por uma fonte de tensão contínua, foi utilizada uma bateria de 9 V.

Para receber as informações do usuário, foi conectado um teclado matricial de quatro por quatro teclas à placa. Assim, completa-se o diagrama do sistema de aquisição da frequência (FIGURA 18).

Figura 18 – Diagrama elétrico do sistema de aquisição




Fonte: Autor.

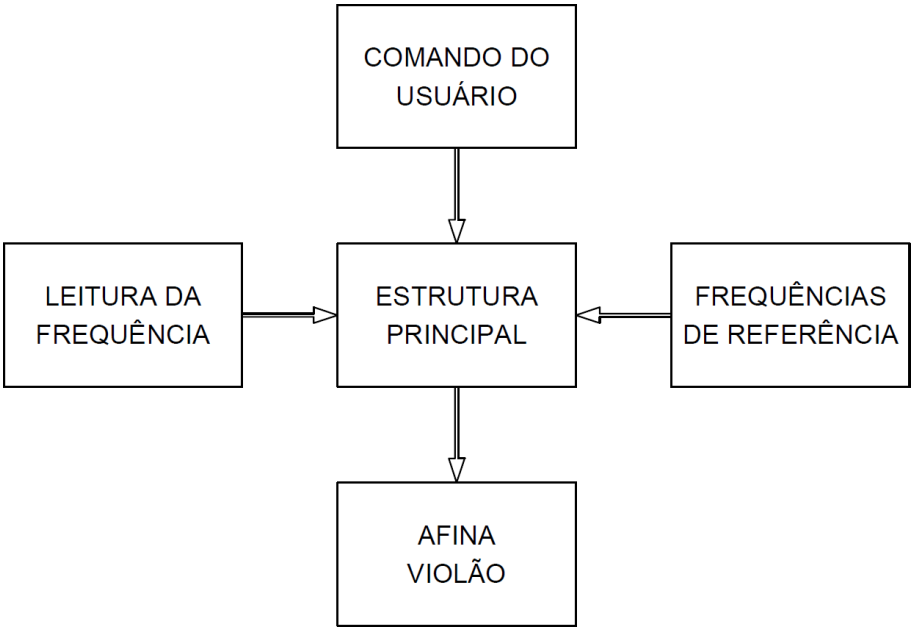
### 3.5.1 Teclado alfanumérico

O pressionamento de uma tecla do periférico fecha um contato entre o terminal correspondente à linha e o terminal correspondente à coluna onde a mesma está posicionada. Assim, são disponibilizadas quatro saídas digitais, cujo acionamento é intercalado, e quatro entradas digitais. Na ocasião de uma das entradas apresentar nível alto, verifica-se qual saída digital foi acionada pelo software neste momento, para então determinar a posição da tecla pressionada. Para garantir que as entradas se mantenham em nível baixo quando não houver tecla correspondente acionada, são utilizados resistores de alta impedância entre estas entradas e o sinal de tensão negativa da placa, caracterizando a configuração *pull-down* (GEEK, 2013).

### 3.5.2 Estrutura do código

O código elaborado para o Arduino é desenvolvido em interface própria, em uma linguagem baseada em C, também composto por estruturas (MCROBERTS, 2013). Cada estrutura é responsável por um elemento do programa. A ordenação dos algoritmos e dados é apresentada na Figura 19.

Figura 19 – Diagrama do software desenvolvido




Fonte: Autor.

Os dados provenientes do teclado alfanumérico conectado, conforme Item 3.5.1, permitem que o usuário informe a afinação desejada pelas teclas A a D e qual a corda que será tocada pelas teclas 1 a 6. Estas informações são processadas na estrutura denominada `teclado` (comando do usuário).

Na estrutura principal, estão armazenados quatro vetores de dados, cada um com as frequências correspondentes às afinações apresentadas no Item 2.2 (afinação padrão; E bemol; sexta em D; G aberto). Ao receber a informação da afinação desejada pelo usuário, o vetor correspondente é carregado aos valores de referência nos quais o sistema deve se basear para atuar.

A estrutura `leitura` (aquisição), foco central do trabalho, foi primeiramente desenvolvida no software MATLAB R2015a e então portada ao Arduino, e está detalhada no decorrer do trabalho.

A estrutura de leitura deve determinar a frequência do sinal de áudio capturado e retornar este valor à estrutura principal. Estes dados também serão disponibilizados na porta serial da placa, para monitoramento durante a programação. À estrutura denominada `afina` serão informados estes mesmos valores para que esta execute a atuação necessária, conforme a diferença de frequências em hertz e o número de passos por hertz determinado no Item 3.4.

## 3.6 Software de aquisição

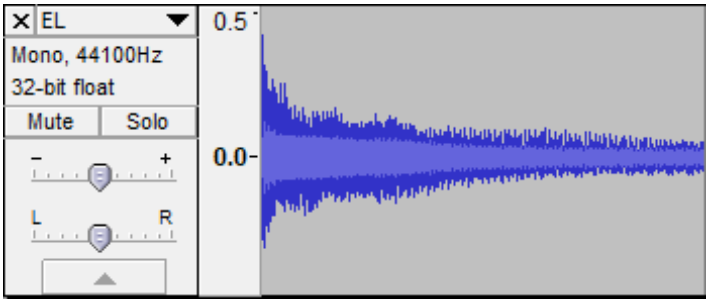
O código para leitura de frequências foi desenvolvido no software MATLAB e alterado constantemente até que fosse possível detectar a frequência de uma nota musical gerada pelo violão Tagima utilizado neste trabalho. A partir desta ocasião, o código começou a ser portado para o Arduino.

### 3.6.1 MATLAB

Primeiramente, o violão foi conectado a um computador pessoal onde o software Audacity está instalado. Todas as cordas foram tocadas, individualmente, e os sons gerados por cada uma foram capturados e gravados em arquivos digitais WAVE (sem compressão), que armazenam 44.100 amostras por segundo em formato *float* de 32 bits, em valores que variam de -1 a 1.

A Figura 20 exibe o espectro do áudio capturado quando tocada a sexta corda solta, onde nota-se que a amplitude do sinal diminui com o tempo.

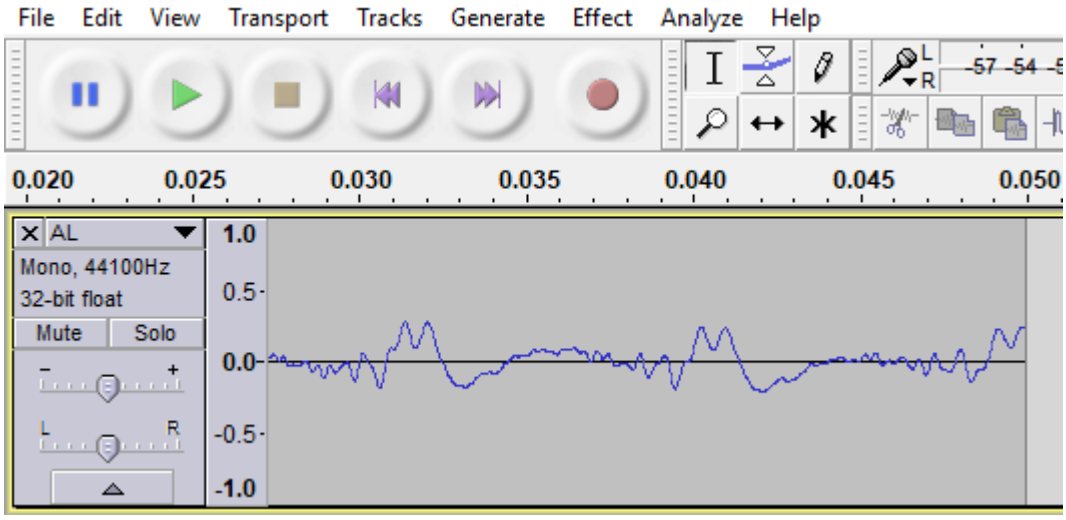
Figura 20 – Áudio capturado no programa Audacity




Fonte: Autor.

Destas amostras de áudio, foram gravados em novos arquivos pequenos intervalos do sinal, de 100 e 50 ms (FIGURA 21), para que então os testes no MATLAB pudessem ser realizados.

Figura 21 – 50 ms de áudio




Fonte: Autor.

O MATLAB pode ler arquivos WAVE e armazená-los como um vetor de dados, ocupando 44.100 posições de memória a cada segundo de som lido (MATHWORKS, 2017). Portanto, um intervalo de 100 ms de áudio resulta em um vetor de 4.410 posições.

Neste momento, foi atribuído um valor inicial de frequência de amostragem que então seria portado para a placa. O Arduino Due consegue realizar uma leitura analógica a cada 100 µs (ARDUINO, 2016), assim, a máxima frequência de amostragem poderia ser de 10 kHz. Considerando o teorema de Nyquist-Shannon (ITEM 2.4), o espectro harmônico do sinal capturado (ITEM 3.2), e a máxima

frequência fundamental emitida – 330 Hz – adotou-se a amostragem de 4.096 Hz, ainda ponderando que serão realizadas transformadas rápidas de Fourier, que operam com ordens de potências de 2, para manter a resolução da frequência capturada um número inteiro.

No código elaborado, a função de carregamento das amostras de áudio foi inserida diversas vezes, sendo uma declarada normalmente e as demais em forma de comentários. Assim, tornou-se possível a rápida troca do sinal analisado, permitindo que as alterações feitas ao decorrer do desenvolvimento do programa pudessem ser testadas em todas as frequências.

O MATLAB possui uma função de subamostragem (MATHWORKS, 2017). Ou seja, após a leitura do sinal de 44,1 kHz é possível amostrá-lo em frequências menores, para corresponder à leitura do Arduino. De 44.100 Hz não é possível amostrar em exatamente 4.096 Hz, porém foi utilizada uma frequência próxima (44.100 / 11 = 4.009,1), e ao final dos cálculos é aplicado um fator de correção.

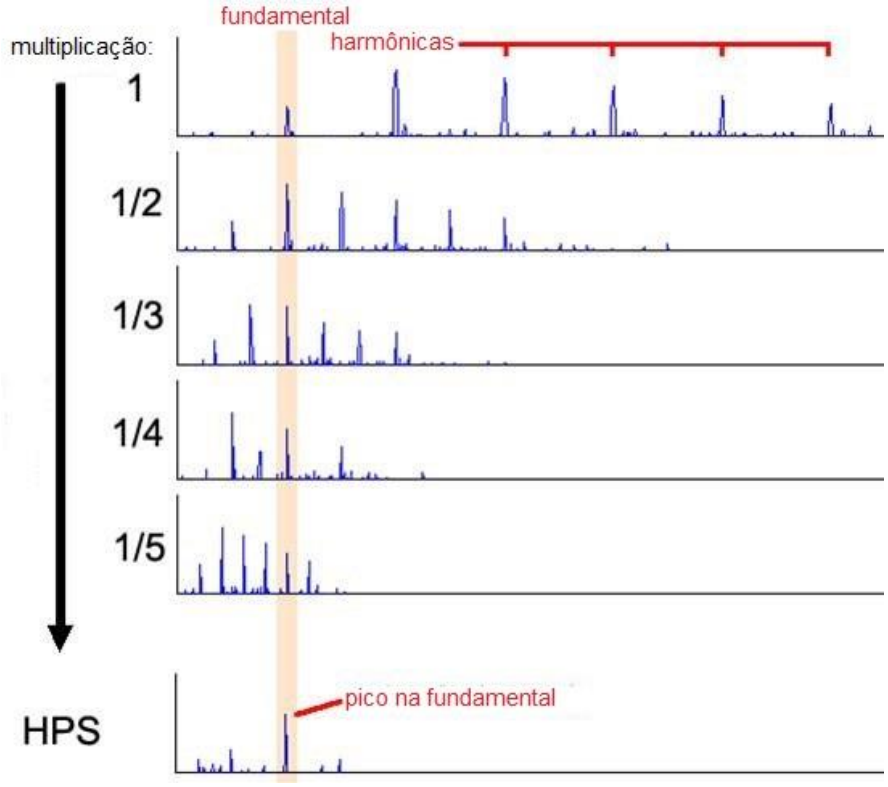
Para aplicar uma função janela, como apresentado no Item 2.4.4, o MATLAB também possui funções prontas (MATHWORKS, 2017). É necessário informar o tipo de janela – neste caso, Hann – e o número de amostras do sinal, que é 401 nas amostras de 100 ms e 201 nas amostras de 50 ms. A análise dos sinais após a aplicação das janelas mostrou, de fato, melhores resultados.

Após o condicionamento digital dos sinais, foi aplicada a transformada rápida de Fourier. A FFT lê os sinais discretos no domínio do tempo e retorna os componentes do mesmo no domínio da frequência, separando-os de acordo com a ordem em que a função foi invocada. É possível determinar qualquer número para a ordem da transformada, porém o processamento será o mesmo que para a próxima potência de 2, se este não for o caso. Então, como a mínima frequência de leitura foi escolhida como 4.096 Hz, a ordem da FFT deve ser também 4.096 para que a resolução de leitura seja de 1 Hz.

São analisados os componentes do resultado da FFT para indicar em qual intervalo de 1 Hz de frequência está a fundamental do sinal analisado. Se o componente mais intenso for o primeiro, por exemplo, sabe-se que o sinal tem entre zero e 1 Hz, ou seja, trata-se de um sinal majoritariamente contínuo.

Para aumentar a confiabilidade destas respostas, a técnica de DSP conhecida como espectro do produto harmônico (HPS) pode ser aplicada nos sinais analisados pelas FFT. O HPS consiste em subamostrar o sinal em números inteiros para que as harmônicas nestas ordens se alinhem com a frequência fundamental, destacando-a quando as amostragens são multiplicadas (FIGURA 22).

Figura 22 – Espectro do produto harmônico até a quinta ordem




Fonte: Adaptado de Naiadseye (2013).

O HPS só pode ser aplicado se o sinal analisado possuir harmônicas intensas, sendo inclusive recomendado quando alguns componentes superam a intensidade da fundamental (NAIADSEYE, 2013). Analisando os sinais do violão no Audacity, considerou-se o produto do espectro até a terceira harmônica.

Por fim, após determinar qual o componente mais intenso da FFT, foi aplicado o teorema de Parseval (EQUAÇÃO 5) para determinar a porcentagem de energia desta frequência e suas primeiras harmônicas em relação à potência total do sinal. Para todas as cordas, os intervalos de 50 ms retornaram números maiores (mais desejáveis) que os de 100 ms. Novos valores de tempo foram testados, mas o intervalo de 50 ms mostrou-se o mais adequado.

A porcentagem mínima para considerar a frequência capturada como válida pelo teorema de Parseval, baseada nos dados retornados pelo MATLAB, foi definida em 60%. Quanto mais puro o sinal na entrada do sistema, maior este número pode ser.

### 3.6.2 Arduino Due

A primeira alteração entre o código no MATLAB e o código no Arduino é a aquisição do sinal. Enquanto o MATLAB lê arquivos WAVE de 44,1 kHz e subamostra os dados a 4 kHz, o Arduino lê os valores diretamente do pino de entrada analógica (ADC) a 4.096 Hz. Com esta frequência, uma leitura deve ocorrer a cada 244 µs. Como uma aquisição leva cerca de 100 µs, foi inserido um *delay* de 144 µs entre as amostras. Após testes, verificou-se que este número deveria ser 146 µs. Também seria possível a utilização de um *timer* nesta etapa. Adicionalmente, o primeiro acesso a uma porta analógica pode levar mais tempo (ARDUINO, 2016), então é realizada uma leitura na rotina de setup da placa, apenas para inicialização do ADC.

Além disso, os valores lidos são números inteiros que variam de 0 a 1.023, diferentemente dos números com ponto flutuante de -1 a 1 interpretados pelo MATLAB. A amplitude destes números não tem influência nos resultados da FFT, porém um *offset* de 488, lido na porta quando a placa do circuito de condicionamento está conectada, pode interferir nos resultados. Desta forma, este valor foi subtraído da amostragem, criando números positivos e negativos, de forma a retirar digitalmente o componente contínuo do sinal de entrada gerada pela placa de condicionamento.

Como o número de amostragens já fora determinado – 200 – foi chamada a função `Hann(200)` no MATLAB e o resultado, um vetor de duzentos dados, foi inserido no código do Arduino para que seja feita a multiplicação pelo sinal de entrada.

Para o cálculo da FFT, foi importada a biblioteca `complex.h`, comum em linguagens de programação como C, para lidar com os números complexos requeridos na execução das transformadas, assim como utilizada uma função recursiva da transformada rápida de Fourier pelo algoritmo de Cooley-Tukey, uma vez

que as funções de FFT específicas em bibliotecas para Arduino não se mostraram adequadas, sendo a maioria limitada a ordem de 256.

O desenvolvimento do algoritmo foi baseado no pseudocódigo para sistemas microprocessados apresentado por Smith (2011).

O espectro do produto harmônico foi realizado com laços de repetição e a equação do teorema de Parseval foi escrita no código, de modo semelhante ao software no MATLAB. Ainda foi adicionado um sistema de validação onde as três últimas frequências devem ser válidas e não distantes entre si para que só então o sistema possa atuar.

## 3.7 Materiais

Na Tabela 8 estão listados os principais materiais utilizados neste trabalho.

Tabela 8 – Lista de materiais

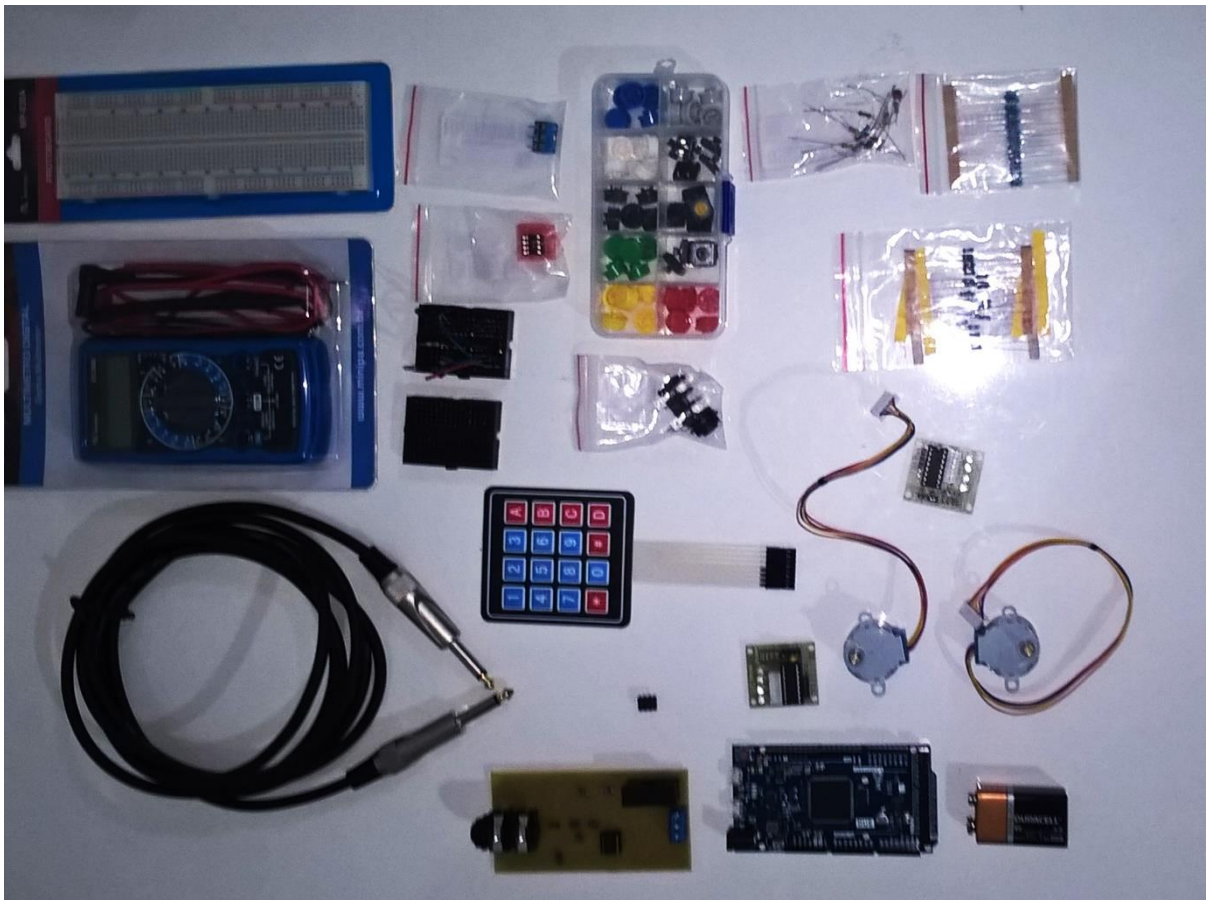
| **Descrição** | **Qtde.** | **Val. unitário** | **Valor total** |
|---|---|---|---|
| Bateria 9V Duracell | 1 | R$ 39,90 | R$ 39,90 |
| Cabo áudio P10 Stagg | 1 | R$ 27,00 | R$ 27,00 |
| Caixa plástica ABS 200x120x75mm | 1 | R$ 29,90 | R$ 29,90 |
| Circuito integrado LM358-N | 2 | R$ 0,73 | R$ 1,46 |
| Conector borne KRE 3 vias | 2 | R$ 1,90 | R$ 3,80 |
| Jack J10 mono | 2 | R$ 1,83 | R$ 3,66 |
| Mini placa de ensaio 170 pontos | 2 | R$ 6,90 | R$ 13,80 |
| Motor de passo com driver | 1 | R$ 19,90 | R$ 19,90 |
| Placa Arduino Due | 1 | R$ 129,90 | R$ 129,90 |
| Placa de ensaio com jumpers | 1 | R$ 89,90 | R$ 89,90 |
| Placa de fibra de vidro 5x10cm | 1 | R$ 2,47 | R$ 2,47 |
| Porta fusível para PCI | 2 | R$ 0,37 | R$ 0,74 |
| Prensa-cabos | 2 | R$ 0,99 | R$ 1,98 |
| Resistores de precisão diversos | - | - | R$ 1,90 |
| Soquete 8 pinos estampado DIP-8 | 2 | R$ 0,90 | R$ 1,80 |
| Suporte bateria 9V | 1 | R$ 1,90 | R$ 1,90 |
| Teclado matricial 16 teclas | 1 | R$ 12,90 | R$ 12,90 |
| | | Total: | R$ 382,91 |

Fonte: Autor.

Determinados componentes sem custo considerável, como capacitores de disco, fios de cobre, fusíveis de vidro e semelhantes, bem como equipamentos utilizados durante algumas etapas dos testes, como multimedidor, osciloscópio, fonte chaveada de tensão e gerador de funções, não estão inclusos nesta tabela.

Parte dos materiais está disposto na Figura 23.

Figura 23 – Componentes utilizados



Fonte: Autor.

Os softwares utilizados no decorrer deste trabalho, em ocasiões de testes ou desenvolvimento, estão listados na Tabela 9.

Tabela 9 – Lista de softwares

| **Programa** | **Versão** |
|---|---|
| AP Tuner | 3.08 |
| Arduino | 1.8.2 |
| Audacity | 2.1.2 |
| Eagle | 7.7.0 |
| MATLAB | R2015a |
| Proteus ARES | V7.8 |
| Proteus ISIS | V7.8 |
| SkecthUp Make | 2017 |

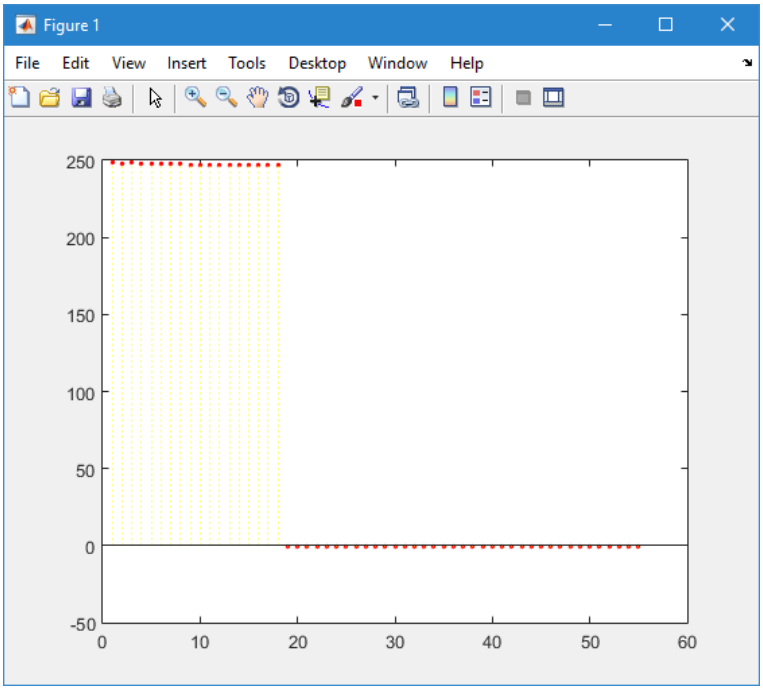
Fonte: Autor.

# 4 RESULTADOS

## 4.1 Validação

Conforme definido no Item 3.6.1, o intervalo de tempo usado para análise do sinal foi de 50 ms. Importando alguns segundos de áudio, como na Figura 21, subamostrando o sinal e dividindo-o nos intervalos de 50 ms, tem-se o resultado apresentado na Figura 24 quando executado o algoritmo após carregada a amostragem correspondente à segunda corda (B3, 247 Hz).

Figura 24 – Resultados para captura da nota B3

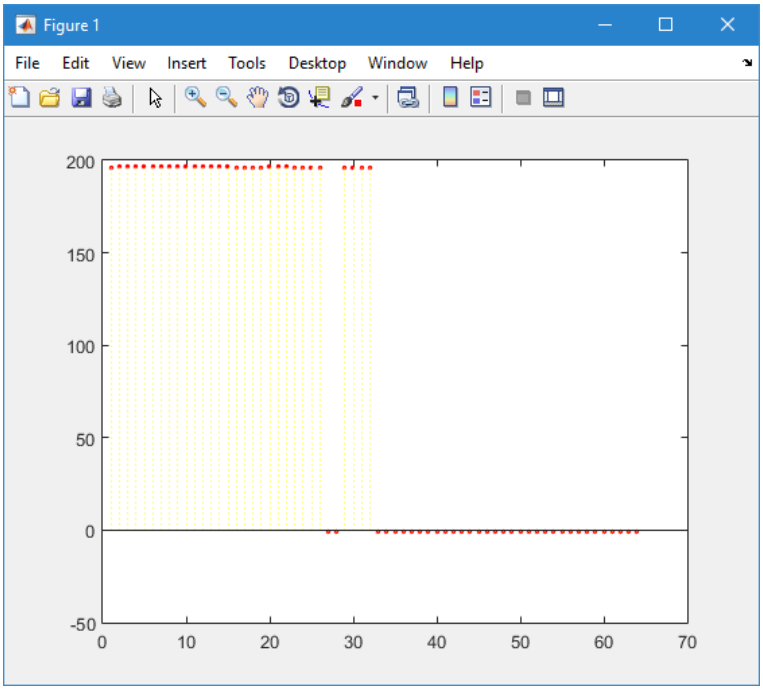



Fonte: Autor

Nota-se que as primeiras dezoito amostragens retornam frequências que variam pouco entre si e que correspondem a mais de 60% da energia presente no sinal, validando assim a frequência correta para esta análise.

A Figura 25 mostra o resultado para a execução do código após carregada a amostragem da terceira corda (G3, 196 Hz), onde as 26 primeiras análises foram feitas com sucesso, além de mais quatro após duas de insucesso.

Figura 25 – Resultados para captura da nota G3




Fonte: Autor

Para as demais cordas, os resultados foram semelhantes aos acima apresentados, com taxas de sucesso variando entre estes dois valores. Esta diferença deve-se principalmente à intensidade com a qual a corda foi tocada no momento da aquisição no software Audacity.

Verificou-se então que, com quanto mais intensidade a corda é excitada pelo músico, maior a chance de a leitura retornar um valor válido. Este comportamento é esperado, pois o nível de ruído mantém-se o mesmo durante toda a aquisição, assim, o sinal torna-se mais puro conforme os componentes válidos do mesmo se destacam.

Após conferir que o código desenvolvido e testado no MATLAB mostrou resultados satisfatórios, todos os parâmetros modificados durante a execução do mesmo foram portados à plataforma Arduino.

## 4.2 Sistema

A interação do microprocessador com os periféricos ocorreu da forma esperada. A aquisição de dados por meio de placa de circuito impresso funcionou corretamente, bem como a varredura e leitura do teclado matricial alfanumérico. O controle do motor de passo aplicado às tarraxas do violão também ocorreu com sucesso.

Entretanto, antes de o sistema ser posto a rodar, ainda seria necessário testar a estrutura de leitura, que já havia funcionado no MATLAB. Neste momento verificou-se que a placa para de responder ao tentar realizar a função de transformada rápida de Fourier. Após tentativas de alteração no código, percebeu-se que o Arduino realizava a FFT até a ordem de 512, apenas. Esta ordem faz com que cada componente do resultado da transformada tenha a largura de 8 Hz, em comparação ao 1 Hz desejado. Ou seja, a exatidão do sistema sobe para ±4 Hz.

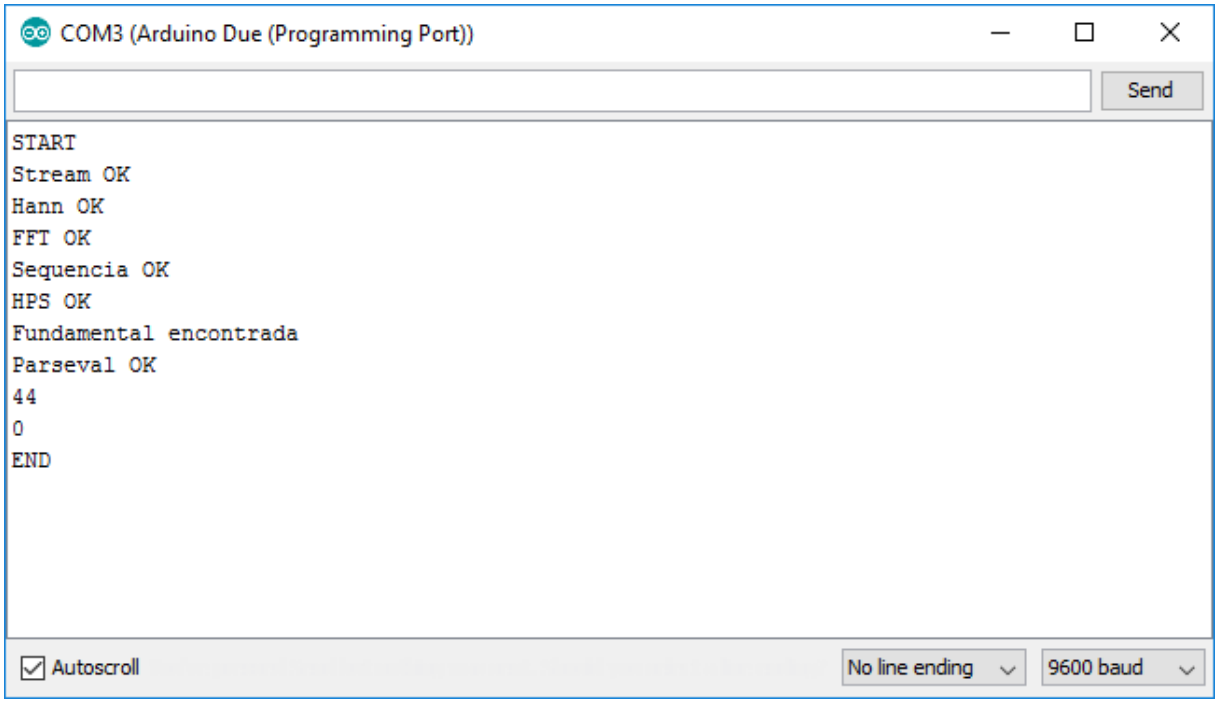
Então, foram realizadas avaliações, desafinando as cordas do violão 4 Hz abaixo e acima das frequências padrão, monitorando o processo com um afinador comercial. O som resultante é nitidamente desagradável, sendo o ideal que a desafinação não passe de ±1,5 Hz, e o almejado que não passe de ±0,5 Hz.

Ainda assim, para que desenvolvimento do trabalho não fosse completamente interrompido, o sistema continuou a ser testado com a FFT em ordem de 512, utilizando os mesmos dados fornecidos pela subamostragem do MATLAB ao invés da aquisição da própria placa, para que os resultados pudessem ser comparados.

A Figura 26 mostra o processo monitorado pela porta serial do Arduino, onde a FFT é aplicada com ordem de 512 em uma janela de 50 ms de áudio. Os dois dados após a confirmação de que o teorema de Parseval foi executado tratam-se do índice

do componente da FFT de maior intensidade e a tentativa de validação do mesmo, que, por não ter tido sucesso, retorna zero.

Figura 26 – Monitor serial do Arduino



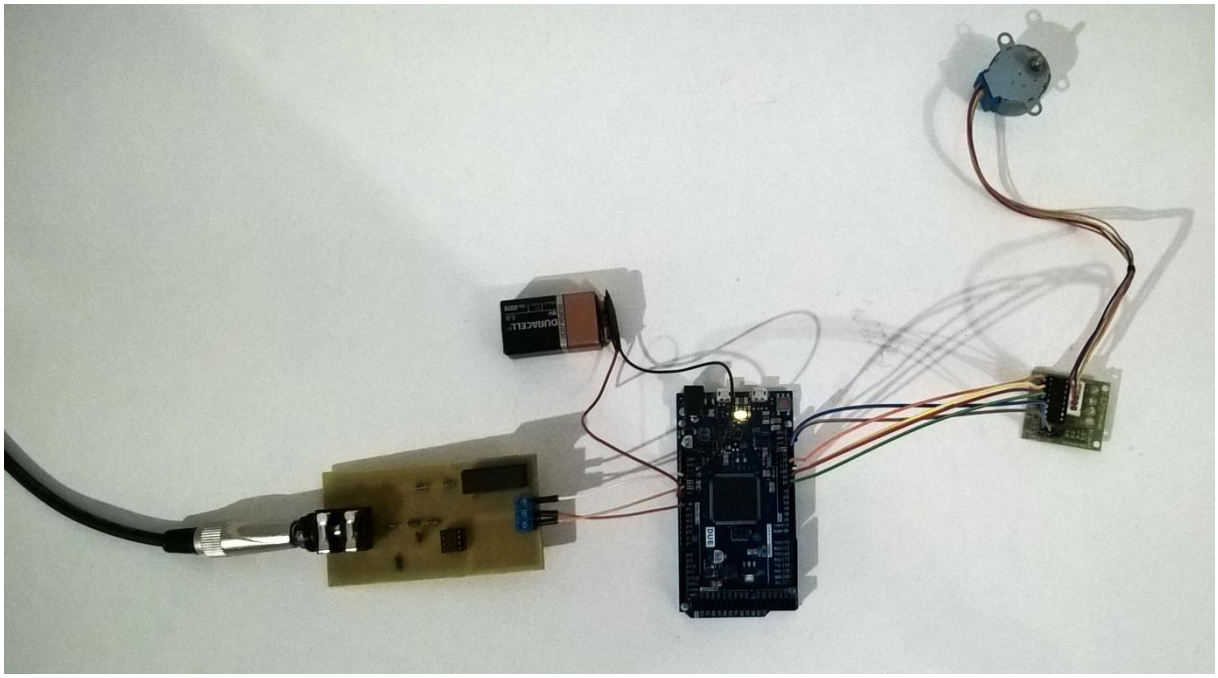

Fonte: Autor.

Notou-se então que, com estes parâmetros, a equação de Parseval dificilmente valida o componente encontrado, e então seria necessário desativá-la para continuar os experimentos.

Porém, com a validação desativada, o sistema tentaria atuar baseando-se em ruídos quando não houvesse sinal de entrada. Assim, foi desenvolvido um modo de operação onde o sistema compara a frequência lida e determina automaticamente qual corda foi tocada baseando-se em um intervalo de frequência de 12 Hz no entorno do valor nominal da corda. Por exemplo, se uma frequência entre 98 Hz e 122 Hz for detectada, tenta-se corrigir a afinação para 110 Hz, da quinta corda (A2).

Desta forma, é dispensado temporariamente o uso do teclado e o sistema jamais atua mais do que 12 Hz, evitando assim que uma corda seja danificada. Também foi implementada a alimentação definitiva de 9 V por bateria. No primeiro momento, o motor foi testado desacoplado do violão (FIGURA 27).

Figura 27 – Sistema parcial alimentado por bateria



Fonte: Autor.

Neste caso, o motor movimentou-se corretamente em sentido horário e anti-horário dependendo de a corda estar afinada de forma mais grave ou aguda em relação ao valor alvo, porém isto não ocorria todas as vezes e a precisão do ajuste ficou aquém do esperado.

# 5 CONCLUSÃO

Este trabalho apresentou as etapas do desenvolvimento de um sistema objetivado a detectar a frequência de um som no conector de saída de um violão para girar um motor de passo acoplado a uma das tarraxas do violão, de modo a ajustar as frequências das cordas em valores padrão.

O principal empecilho ao sucesso total deste experimento foi o fato de a placa microcontrolada selecionada não aceitar a ordem necessária de 4.096 no cálculo das transformadas rápidas de Fourier. Limitado à ordem de 512, o sistema microcontrolado mostrou resultados insatisfatórios, o que, entretanto, não inviabiliza a metodologia aplicada e desenvolvida no software MATLAB, onde tal limitação não ocorreu.

Santos (2012) conclui que a máxima ordem para a realização da FFT em um Arduino equipado com processador ATMega1280 é 512, porém o processador utilizado neste trabalho supera o citado em todas as configurações de memória, e ainda assim não foi possível o aumento deste limite. Até a ordem de 512, o retorno é imediato, sem atrasos no processamento. A partir de 1.024, não há resposta.

A função utilizada nestes cálculos foi escrita também em linguagem C e testada em ambiente Linux, com a ordem desejada de 4.096, tendo o mesmo retorno que o sistema desenvolvido e testado no MATLAB, concluindo que não se trata de um erro de código, mas de uma provável limitação da placa.

O código de leitura, incluindo as funções utilizadas para o cálculo das transformadas rápidas de Fourier, pode ser analisado no Apêndice A.

## 5.1 Melhorias

Para aumentar a confiabilidade da amostragem, pode ser incorporado um filtro passa-baixas no circuito de condicionamento do sinal para diminuir erros na leitura por efeito de *aliasing*. Também é possível substituir os *delays* entre as amostras da aquisição por um *timer* que estoura a cada 244 µs.

Como melhoria à interface com o usuário, pode ser criado um sistema de indicação baseado em diodos emissores de luz (LED) que informe ao usuário os estados do sistema, como leitura, afinação, espera, e informações quanto ao estado da corda selecionada. Ainda poderiam ser utilizados displays de cristal líquido (LCD) – como o Arduino Due opera em 3,3 V, não há *shields* compatíveis com este modelo da placa, porém é possível desenvolver uma interface a partir de um LCD genérico.

O circuito de varredura e leitura do teclado matricial, com resistores em configuração *pull-down* e conectores, pode ser impresso em placa de fibra de vidro.

Ainda é possível elaborar um algoritmo mais complexo de controle em malha fechada, que combina ações de aquisição e atuação, de modo a tornar a afinação automática mais rápida e precisa.

## 5.2 Continuidade

Uma vez que o algoritmo criado no MATLAB se mostrou válido, propõe-se para a continuidade do trabalho a implementação de tal código (operações de DSP), juntamente com o que foi desenvolvido diretamente no Arduino (periféricos), em outras plataformas.

Como possibilidades, há o sistema microprocessado Raspberry Pi, cujas configurações de hardware superam a placa utilizada, além da opção de implementar o código diretamente em um processador ARM como o deste sistema (o Arduino Due utiliza o microprocessador Atmel SAM3X8E ARM Cortex-M3 CPU), fazendo uso de dispositivos de memória e interface conforme a necessidade.

# REFERÊNCIAS

ANTONIOU, Andreas. **Digital Signal Processing**: Signals, Systems, and Filters. Nova Iorque: McGraw-Hill, 2005.

ARDUINO. **Arduino Board Due**. 2016. Disponível em: <https://www.arduino.cc/en/Main/ArduinoBoardDue>. Acesso em: agosto 2016.

AUDACITY Team. **Audacity 2.1.2 Manual**. 2016. Disponível em: <http://manual.audacityteam.org/index.html>. Acesso em: novembro 2016.

BAND Industries. **Roadie Automatic Guitar Tuner**. 2017. Disponível em: <https://www.roadietuner.com/>. Acesso em: janeiro 2017.

BRAIN, Marshall. How Stuff Works: **How Acoustic Guitars Work**. 2014. Disponível em: <http://entertainment.howstuffworks.com/guitar.htm>. Acesso em: agosto 2016.

BRITES, Felipe Gonçalves. **Motor de Passo**. Niterói: Universidade Federal Fluminense, 2008.

CERVI, Gabriel. Tudo o que você sempre quis saber sobre: **Aulas de Violão**. Curitiba, 2013. Disponível em: <http://escolabackstage.blogspot.com.br/2013_11_01_archive.html>. Acesso em: setembro 2016.

CODE Project. **FFT Guitar Tuner**. Toronto, 2009. Disponível em: <http://www.codeproject.com/Articles/32172/FFT-Guitar-Tuner>. Acesso em: setembro 2016.

DEAMBULATORY Matrix, The. **Digital Chromatic Guitar Tuner**. 2010. Disponível em: <http://deambulatorymatrix.blogspot.com.br/2010/11/digital-chromatic-guitar-tuner-2008.html>. Acesso em: setembro 2016.

DINIZ, Paulo Sérgio Ramirez; SILVA, Eduardo Antônio Barros da; NETTO, Sergio Lima. **Processamento Digital de Sinais**: Projeto e Análise de Sistemas. São Paulo: ARTMED, 2002.

ELETRO Física. **Motor de Passo**. 2013. Disponível em: <http://eletrofisica1.blogspot.com.br/2013/05/motor-de-passo_2023.html>. Acesso em: junho 2017.

EPIPHONE Guitar Corp. **Min-ETune**. 2017. Disponível em: <http://www.epiphone.com/Min-ETune.aspx>. Acesso em: janeiro 2017.

FARNELL element14. **3V to 12V Operational Amplifiers**. Leeds, 2016. Disponível em: <http://uk.farnell.com/operational-amplifiers-op-amps/supply-voltage-range/3v-to-12v/pg/110182464>. Acesso em: outubro 2016.

FILIPE Flop. **Controlando um Motor de Passo de 5V com Arduino**. 2013. Disponível em: <http://blog.filipeflop.com/motores-e-servos/controlando-um-motor-de-passo-5v-com-arduino.html>. Acesso em: março 2017.

GEEK Factory. **Teclado Matricial con PIC**. México, 2013. Disponível em: <http://www.geekfactory.mx/tutoriales/tutoriales-pic/teclado-matricial-con-pic/>. Acesso em: junho 2017.

GHASSAEI, Amanda. Instructables: **Arduino Frequency Detection**. 2012. Disponível em: <http://www.instructables.com/id/Arduino-Frequency-Detection/>. Acesso em: agosto 2016.

GIBSON Guitar Corporation. **Play in Open G Tuning Like Keith Richards**. Nashville, 2014.

GRIMWOOD, Nicole. Instructables: **Arduino Guitar Tuner**. 2012. Disponível em: <http://www.instructables.com/id/Arduino-Guitar-Tuner/>. Acesso em: agosto 2016.

KIATRONICS Electronic Design and Manufacture. **28BYJ-48**: 5V Stepper Motor. Cherrywood, 2017. Disponível em: <http://img.filipeflop.com/files/download/Datasheet_28BYJ-48.pdf>. Acesso em: fevereiro 2017.

LIMA, Rafael. Projetos de Medição de Grandezas Físicas: **Afinador para violão**. Salvador: UFBA, 2013. Disponível em: <https://engc49.wordpress.com/2013/09/06/afinador-para-violao/>. Acesso em: agosto 2016.

LOURDE R., Mary; SAJI, Anjali Kuppayil. A Digital Guitar Tuner. **International Journal of Computer Science and Information Security**. Dubai: BITS, vol. 6, num. 2, 2009. ISSN 1947-5500.

LYONS, Richard. **Single tone detection with the Goertzel algorithm**. Mountain View, 2012. Disponível em: <http://www.embedded.com/design/real-world-applications/4401754/Single-tone-detection-with-the-Goertzel-algorithm>. Acesso em: outubro 2016.

MANDAQUI Musical. **Teclado**. Mandaqui, 2009. Disponível em: <http://www.oocities.org/br/mandaquimusical/teclado.htm>. Acesso em: setembro 2016.

MATHWORKS Inc, The. **MATLAB Documentation**. Natick, 2017. Disponível em: <https://www.mathworks.com/help/matlab/>. Acesso em: janeiro 2017.

MCROBERTS, Michael. **Arduino Básico**. 4. ed. São Paulo: Novatec, 2013.

NAIADSEYE. **Fast Fourier Transform Tuner**. 2013. Disponível em: <https://naiadseye.wordpress.com/2013/10/09/fast-fourier-transform-tuner/>. Acesso em: fevereiro 2017.

NALON, José Alexandre. **Introdução ao processamento digital de sinais**. Rio de Janeiro: LTC, 2009.

OPPENHEIM, Alan V; SCHAFER, Ronald W. **Discrete-Time Signal Processing**. 3. ed. Upper Saddle River: Pearson, 2010.

PROAKIS, John G; MANOLAKIS, Dimitris G. **Digital Signal Processing**: Principles, Algorithms, and Applications. 4. ed. Nova Jérsei: Pearson, 2007.

PUHLMANN, Henrique. Trazendo o mundo real para dentro do processador: **Conversor A/D**. São Paulo, 2015. Disponível em: <http://www.embarcados.com.br/conversor-a-d/>. Acesso em: outubro 2016.

ROBERTS, Michael J. **Fundamentos em sinais e sistemas**. Tradução de Carlos Henrique Nogueira de Resende Barbosa. Porto Alegre: AMGH, 2010.

ROEDERER, Juan G. **Introdução à física e psicofísica da música**. Tradução de Alberto Luis da Cunha. São Paulo: EDUSP, 1998.

SALVI, Dario. **Sound analysis in Arduino**. Madri, 2013. Disponível em: <https://bochovj.wordpress.com/2013/06/23/sound-analysis-in-arduino/>. Acesso em: agosto 2016.

SANTOS, Nuno Pessanha. Arduino e o cálculo da FFT. **Revista Programar**. ed. 35. Portugal, 2012.

SMITH, Steven W. **The Scientist and Engineer's Guide to Digital Signal Processing**. Poway: California Technical Publishing, 2011.

SUEIRO, Diego**. Arduino vs. Raspberry Pi**: entenda as diferenças. São Paulo, 2014. Disponível em: <http://www.embarcados.com.br/arduino-vs-raspberry-pi/>. Acesso em: setembro 2016.

TEKTRONIX, Inc. Digital Storage Oscilloscopes: **TDS1000C-EDU Series Datasheet**. Beaverton, 2016. Disponível em: <http://www.tek.com/sites/tek.com/files/media/media/resources/TDS1000C-EDU-Oscilloscope-Datasheet-3GW265156.pdf>. Acesso em: novembro 2016.

TEXAS Instruments. **High-Voltage, High-Current Darlington Transistor Array**. Dallas, 2017. Disponível em: <http://www.ti.com/lit/ds/symlink/uln2003a.pdf>. Acesso em: fevereiro 2017.

VONK, Johan. **Pitch Detection on Arduino using Autocorrelation**. Los Altos, 2015. Disponível em: <http://www.coertvonk.com/technology/embedded/arduino-pitch-detector-13252/6>. Acesso em: outubro 2016.

WEISSTEIN, Eric W. **CRC Concise Encyclopedia of Mathematics**. 2. ed. Boca Raton: CRC Press, 2012.

WENDLING, Marcelo. **Amplificadores Operacionais**. v. 2.0. Guaratinguetá: UNESP, 2010.

# APÊNDICE A – CÓDIGO DE AQUISIÇÃO PARA ARDUINO

```
#include <complex.h>
typedef double _Complex cplx;

int result0, result1, result2;
float HANN[200];
//gravação do vetor de 200 dados gerado pelo MATLAB
//omitido nesta impressão do código por ser muito extenso

void _fft(cplx buf[], cplx out[], int n, int step) {
  if (step < n) {
    _fft(out, buf, n, step*2);
    _fft(out+step, buf+step, n, step*2);
    for (int i = 0; i < n; i += 2*step) {
      cplx t = cexp(-I*PI*i/n) * out[i+step];
      buf[i/2] = out[i]+t;
      buf[(i+ n)/2] = out[i]-t;
    }
  }
}

void fft(cplx buf[], int n) {
  cplx out[n];
  for (int i = 0; i < n; i++) out[i] = buf[i];
  _fft(buf, out, n, 1);
}
```

```
int leitura() {
  cplx stream[4096]; //cplx 4096 pois FFT retorna aqui
  for(int i=0; i<200; i++) {
    stream[i] = analogRead(A0);
    //244us - descontar tempo de leitura e gravação (~98us)
    delayMicroseconds(146);
  }
  Serial.println("Stream gravado");
  delay(3000);

  //aplicar Hann
  for(int i = 0; i < 200; i++) {
    stream[i] = stream[i] * HANN[i];
  }
  Serial.println("Hann OK");
  delay(3000);

  //ERRO OCORRE NESTA FUNÇÃO:
  //fft(stream, 4096);
  //até ordem 512 funciona ok, inclusive resultados
  fft(stream, 512);
  Serial.println("FFT OK");
  delay(3000);

  //converter em números reais (módulo) (primeira metade apenas)
  float fft2[2048];
  for(int i = 0; i < 2048; i++) {
    fft2[i] = sqrt(pow(creal(stream[i]),2)+pow(cimag(stream[i]),2));
  }
  Serial.println("Sequencia OK");
  delay(3000);

  //subamostrar o sinal (1024, 683)
  float hps1[1024];
  for(int i = 0; i < 1024; i++){
    hps1[i] = fft2[i*2];
  }
  float hps[683];
  for(int i = 0; i < 683; i++){
    hps[i] = fft2[i*3];
  }
  //multiplicar as amostras - poderia juntar no "for" acima
  for(int i = 0; i < 683; i++){
    //x(683) = x(683)*x(1024)*x(2048)
    hps[i] = fft2[i]*hps1[i]*hps[i];
  }
  Serial.println("HPS OK");
  delay(3000);
```

```
//encontrar pico
int index = 0;
float peak = 0;
//caso não seja um valor válido mantém zero
for(int i = 60; i < 360; i++) {
  if(hps[i] > peak) {
    peak = hps[i];
    index = i;
  }
}
Serial.println("Fundamental encontrada");
delay(3000);

//energia total
float e_t = 0;
float hps2[683];
for(int i = 683; i < 683; i++) {
  hps2[i] = pow(hps[i],2);
  e_t = e_t + hps2[i];
}
//energia da frequência - amostragem considera até sexta harmônica
int a = 30;
float e_h = 0;
//deve validar apenas se o índice for válido
if(index != 0) {
  for(int i = index-a; i <= index+a; i++){
    e_h = e_h + hps2[i];
  }
  for(int i = 2*index-a; i <= 2*index+a; i++){
    e_h = e_h + hps2[i];
  }
  for(int i = 3*index-a; i <= 3*index+a; i++){
    e_h = e_h + hps2[i];
  }
  for(int i = 4*index-a; i <= 4*index+a; i++){
    e_h = e_h + hps2[i];
  }
  for(int i = 5*index-a; i <= 5*index+a; i++){
    e_h = e_h + hps2[i];
  }
  for(int i = 6*index-a; i <= 6*index+a; i++){
    e_h = e_h + hps2[i];
  }
}
Serial.println("Parseval OK");
delay(3000);
```

```
  //aceitar a fundamental se e_h > 60%
  int out = 0;
  if (e_h/e_t > 0.6) {
    out = index+1; //índice FFT
  }
  Serial.println(index+1); //encontrada
  Serial.println(out); //validada
  delay(3000);

  //comparar três últimos resultados (variáveis globais)
  result2 = result1;
  result1 = result0;
  result0 = out;
  int f_ok = 0;
  if(abs(result0-result1)<3 && abs(result1-result2)<3) {
    f_ok = result1;
  }

  return f_ok;
  //fim
  //Serial.println("END");
  //delay(30000);
}

void setup(){
  analogRead(A0); //inicializa ADC
  Serial.begin(9600);
  Serial.println("START");
  delay(3000);
}

void loop(){
  int freq = leitura();
  Serial.println(freq);
  Serial.println("END");
  delay(10000);
}
```