UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE GRADUAÇÃO EM ENGENHARIA ELETRÔNICA

THAYLINE VALÉRIO DA SILVA

DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE DIGITAL PARA AFINAÇÃO DE INSTRUMENTOS DE CORDA

TRABALHO DE CONCLUSÃO DE CURSO

TOLEDO
2018

THAYLINE VALÉRIO DA SILVA

DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE DIGITAL PARA AFINAÇÃO DE INSTRUMENTOS DE CORDA

Projeto de Pesquisa do Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do curso de Graduação em Engenharia Eletrônica da Coordenação do Curso de Engenharia Eletrônica – COELE – da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito para a obtenção do título de Bacharel em Engenharia Eletrônica.

Orientador: Prof. Dr. Andrés Eduardo Coca Salazar

TOLEDO

2018

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Toledo
Coordenação do Curso de Engenharia Eletrônica

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

TERMO DE APROVAÇÃO

Título do Trabalho de Conclusão de Curso Nº 73

Desenvolvimento de um sistema de controle digital para afinação de instrumentos de corda

por

Thayline Valério da Silva

Esse Trabalho de Conclusão de Curso foi apresentado às 10h do dia **03 de dezembro de 2018** como requisito parcial para a obtenção do título **Bacharel em Engenharia Eletrônica**. Após deliberação da Banca Examinadora, composta pelos professores abaixo assinados, o trabalho foi considerado **APROVADO**.

_____
Prof. Alberto Vinicius de Oliveira
(UTFPR-TD)

_____
Prof. Daniel Cavalcanti Jeronymo
(UTFPR-TD)

_____
Prof. Andrés Eduardo Coca Salazar
(UTFPR-TD)
Orientador

Visto da Coordenação

_____
Prof. Fabio Rizental Coutinho
Coordenador da COELE

O original deste documento encontra-se na coordenação do curso

Dedico este trabalho à minha família e aos meus amigos.

AGRADECIMENTOS

Agradeço ao meu orientador Prof. Dr. Andrés Eduardo Coca Salazar, pela sabedoria com que me guiou nesta trajetória.

Ao Prof. Dr. Gerson Filippini, por me ajudar com a modelagem matemática deste projeto.

Aos meus amigos Isabela Laufer, Vinicius Amano e Gabriel Matias por estarem ao meu lado nos momentos difíceis, sendo braços de apoio e excelentes distrações nos momentos de lazer.

Ao meu irmão Maykon Valério, por me ajudar a superar os obstáculos que surgiram durante este período.

Aos meus pais Rute Martins e Ezequias Valério, por sempre acreditarem em mim.

Sem o apoio de vocês teria sido muito mais difícil vencer esse desafio.

RESUMO

SILVA, Thayline Valério da. **Desenvolvimento de um sistema de controle digital para afinação de instrumentos de corda**. 2018. 77f. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Eletrônica) – Universidade Tecnológica Federal do Paraná. Toledo, 2018.

A qualidade acústica da interpretação de um instrumentista está relacionada com a correta afinação do instrumento musical. No entanto, a afinação dos instrumentos de cordas sofrem variações no decorrer do tempo devido a fatores ambientais e naturais. Por isso, o instrumentista deve conferir constantemente a afinação, o que requer conhecimento e tempo. Portanto, o desenvolvimento de um sistema automático para a afinação de instrumentos de cordas se torna necessário. No mercado podem ser encontrados alguns dispositivos comerciais que prometem a afinação automática, por exemplo o TronicalTune e o RoadieTuner. Entretanto, não são produtos populares por serem muito custosos. Neste projeto, foi desenvolvido um sistema de controle automático para afinar um instrumento acústico de corda com precisão, rapidez e baixo custo. Tal sistema possui um suporte que encaixa na tarraxa da corda que será afinada e através de um *software* o usuário escolhe a frequência desejada para a corda escolhida. O sinal acústico da corda é captado por um microfone; tal sinal é interpretado por uma entrada analógica do microcontrolador do Arduino UNO e manipulado por um algoritmo que estima a frequência fundamental da amostra de áudio mediante o cálculo da autocorrelação. Além disso, o microcontrolador é responsável por determinar a diferença entre a frequência do sinal recebido e a frequência da nota musical desejada. Dois controles digitais, o PID e outro baseado na equação do sistema (obtida através de uma modelagem matemática), foram implementados no microcontrolador para calcular e enviar ao motor de passo o sinal que muda o sentido de rotação, esticando ou afrouxando a corda, e a quantidade de passos necessários para a corda ser afinada. Foram realizados experimentos usando o ukulele, obtendo resultados satisfatório quanto a tempo, inferior a 30 segundos, e precisão, uma diferença máxima de 10 Hz. O sistema pode ser usado tanto por instrumentistas iniciantes quanto avançados, e minimiza o tempo de afinação.

Palavras-chave: Processamento Digital de Sinais. Estimação da frequência fundamental. Controle PID digital.

ABSTRACT

SILVA, Thayline Valério da. **Development of a Digital Control System for Tuning String Instruments**. 2018. 77f. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Eletrônica) – Universidade Tecnológica Federal do Paraná. Toledo, 2018.

The acoustic quality performance of an musical instrument is related to its correct tuning. However, the tuning of string musical instruments varies over time due to environmental and natural factors. Therefore, the musician must constantly check the tuning, which requires technical knowledge and time. Thus, the development of an automatic tuning system for string instruments had become necessary. Some commercial devices that ensure automatic tuning can be easily found in the market, for example TronicalTune and RoadieTuner, althought, they are not popular due to elevated price. In this project, an automatic tuning control system for acoustic string instrument was developed with accuracy, rapidity and low cost. Such system has a metal piece that fits on the tunning machine and by a software the user chooses the desired frequency for the chosen string. A microphone captures the acoustic signal produced by the string; after that, an analog input of the Arduino UNO reads that signal and by an autocorrelation algorithm estimates the fundamental frequency of the audio sample. Besides that, the Arduino is responsible for determining the difference between the frequency of the received signal and the frequency of the desired musical note. Two digital controls, PID and another one based on the system equation (obtained through a mathematical modeling), were implemented in the Arduino to calculate and send to the stepper motor the signal that sets the direction of rotation, stretching or loosening the string, and the amount of steps required for the string to be tuned. Experiments were performed using a ukulele, obtaining satisfactory results in time and accuracy. The system can be used by both beginner and advanced musicians and minimizes tuning time.

Keywords: Digital Signal Processing. Fundamental frequency estimation. Digital PID controller.

LISTA DE FIGURAS

Figura 1 - Notação musical na clave de Sol 21
Figura 2 - Notação musical na clave de Fá..... 21
Figura 3 - Escala natural. 22
Figura 4 - Escala temperada. 22
Figura 5 - Pautas musicais com as respectivas notas de cada corda do violino, viola, violoncelo e contrabaixo..... 24
Figura 6 - Ukulele soprano. 25
Figura 7 - Afinações do Ukulele. 26
Figura 8 – Relação entre harmônicos, comprimento de onda e comprimento da corda: a) primeiro harmônico; b) segundo harmônico; c) terceiro harmônico. 27
Figura 9 - Autocorrelação de um sinal. 29
Figura 10 - Sistema de controle em malha fechada. 31
Figura 11 - Resposta ao degrau de um sistema. 33
Figura 12 - Exemplo de resposta transitória com oscilação constante. 33
Figura 13 - Resposta ao degrau do sistema. 34
Figura 14 - Diagrama de blocos do projeto. 37
Figura 15 - Circuito de ligação do microfone..... 38
Figura 16 - Esquema de ligação do motor de passo e o EasyDriver com o Arduino. 39
Figura 17 – Sinal de saída do microfone visto pelo osciloscópio. 40
Figura 18 - Amplificador não-inversor..... 41
Figura 19 - Sinal do circuito amplificador no osciloscópio. 42
Figura 20 - Circuito ganho DC. 43
Figura 21 - Sinal de saída do circuito de pré-processamento do sinal de áudio. 43
Figura 22 – Diagrama de blocos da etapa de pré-processamento do sinal. 43
Figura 23 - Circuito montado na placa. 44
Figura 24 - Fluxograma do sistema. 45
Figura 25 - Máquina de estados do algoritmo implementado no microcontrolador. 46
Figura 26 - Demonstração da corda do sistema. 48
Figura 27 - Tela inicial da interface de usuário do sistema de afinação automática de instrumento de cordas. 53
Figura 28 - Tela principal da interface de software do protótipo de afinação automática proposto. 53

Figura 29 - Motor com suporte para encaixe na tarraxa do instrumento..... 54
Figura 30 - Motor e microfone fixos no instrumento. 54
Figura 31 - Resposta do sistema com o modelo matemático variando da nota C4 para A4 57
Figura 32- Resposta do sistema com o modelo matemático variando da nota A4 para C4. 57
Figura 33- Resposta do sistema com o controlador PID variando da nota C4 para A4. 58
Figura 34- Resposta do sistema com o controlador PID variando da nota A4 para C4. 58

LISTA DE TABELAS

Tabela 1 - Nome das notas musicais e nomenclatura. 20
Tabela 2 - Afinações padrões dos instrumentos de corda (Hz). 24
Tabela 3 - Frequência das notas das afinações low-G e high-G do ukulele. 26
Tabela 4 - Parâmetros dos controladores pelo método de Ziegler-Nichols. 33
Tabela 5 - Especificações do microcontrolador ATmega328P do Arduino. 38
Tabela 6 - Especificações motor de passo. 39
Tabela 7 - Especificações das cordas do ukulele utilizadas do projeto. 40
Tabela 8 - Parâmetros medidos da corda. 50
Tabela 9 - Gastos obtidos para o desenvolvimento do projeto. 59

LISTA DE GRÁFICOS

Gráfico 1 - Relação entre os valores de frequência mostrada pelo aplicativo Ukulele Tuner e calculada pela autocorrelação implementada. 55
Gráfico 2 - Comparação do tempo de afinação entre o controlador PID e o controle pelo modelo matemático..... 56
Gráfico 3 - Comparação do tempo gasto entre afinação automática e a afinação manual. 59

LISTA DE QUADROS

Quadro 1 - Função que monta o vetor com as amostras do sinal. 47
Quadro 2 - Função do controlador PID. 51
Quadro 3 - Função do controle matemático. 52

LISTA DE SIGLAS E ABREVIATURAS

MIDI	*Musical Instrument Digital Interface*
PID	Proporcional Integral Derivativo
SIFT	*Simplified Inverse Filter Technique*
HPS	*Harmonic Product Spectrum*
ML	*Maximum Likelihood*
SRAM	*Static Random-Access Memory*
EEPROM	*Electrically Erasable Programmable Read-Only Memory*
PWM	*Pulse Width Modulation*
UART	*Universal Asynchronous Receiver-Transmitter*
SPI	*Serial Peripheral Interface*
I2C	*Inter-Integrated Circuit*
DC	*Direct Current*
ADC	*Analogic-Digital Converter*
EQM	Erro Quadrático Médio

LISTA DE SÍMBOLOS

#	Sustenido
♭	Bemol
f	Frequência de vibração de uma corda
n	Harmônico de uma frequência fundamental
L	Comprimento de uma corda
F	Tensão submetida à uma corda
f_0	Frequência fundamental
i	Intervalo de semitons
f_i	Frequência de uma determinada nota
N	Número MIDI
λ	Comprimento de onda
V	Velocidade de propagação de uma onda
T	Tensão
μ	Densidade linear
M	Número de amostras de um sinal
m	Instante de atraso
$r[m]$	Fator de autocorrelação no instante
k	Instante de amostra de um sinal
$x[k]$	Sinal no instante k
$G(s)$	Função de transferência de um sistema
K_p	Ganho proporcional
T_i	Tempo integral
T_d	Tempo derivativo
u(t)	Saída do controlador
y(t)	Saída de um sistema
$e(k)$	Erro no instante k
T_s	Tempo de amostragem
$u[k]$	Sinal de controle
$q_{0,1,2}$	Parâmetros para o cálculo da saída do controlador PID digital
e(t)	Resposta ao degrau unitário
a	Amplitude do sobressinal da resposta ao degrau unitário

K_u	Ganho ajustado
T_u	Período de oscilação
K	Ganho de um sistema
yf	Valor em regime permanente da resposta ao degrau unitário
τ	Constante de tempo de um sistema
θ	Tempo de atraso ou tempo morto
$Y(s)$	Função de transferência da saída
$R(s)$	Função de transferência da entrada
ΔY	Variação da amplitude da saída
ΔU	Variação da amplitude da entrada
$t_{1,2,3}$	Instante de tempo
f_{ref}	Frequência de referência
f_e	Frequência estimada
v_0	Tensão de saída
v_i	Tensão de entrada
R	Resistor
C	Capacitor
F_s	Taxa de amostragem
σ	Tensão de tração
A	Área da secção transversal
l_0	Comprimento inicial
Δl	Variação de comprimento
F	Força
E	Módulo de elasticidade
ϵ	Relação de deformação
$\Delta\theta$	Ângulo
r	Raio
F_s	Taxa de amostragem
T_0	Período estimado
e_r	Erro relativo

SUMÁRIO

1 INTRODUÇÃO 16

2 OBJETIVOS 18

2.1 OBJETIVO GERAL 18

2.2 OBJETIVOS ESPECÍFICOS 18

3 JUSTIFICATIVA 19

4 REFERENCIAL TEÓRICO 20

4.1 TEORIA MUSICAL 20

4.1.1 Notas musicais e suas respectivas notações 20

4.1.2 Escalas musicais 21

4.1.3 Afinação musical 23

4.1.4 Ukulele 25

4.3 FREQUÊNCIA FUNDAMENTAL 26

4.3.1 Limiar de discriminação de frequência auditiva 27

4.3.2 Estimação da frequência fundamental 28

4.3.2.1 Método da Autocorrelação 29

4.4 CONTROLE DE SISTEMAS DINÂMICOS 30

4.4.1 Definições 30

4.4.2 Controlador PID 31

4.4.3 Controlador PID digital 32

4.4.4 Métodos para projeto de PID 32

4.4.4.1 Sintonização de Ziegler-Nichols 32

4.4.4.2 Identificação de sistema por Ziegler-Nichols 33

4.4.5 Modelagem matemática de sistemas 35

5 DESENVOLVIMENTO 37

5.1 Pré-processamento do sinal acústico 40

5.2 Algoritmo implementado no microcontrolador 44

5.2.1 Algoritmo para a estimação do *pitch*: Autocorrelação 46

5.3 Modelagem matemática 48

5.4 Algoritmo de controle 50

5.4.1 Controlador PID 50

5.4.2 Controlador matemático 51

5.5 Interface de usuário 52

6 EXPERIMENTOS E RESULTADOS 55
6.1 Precisão de afinação 55
6.2 Tempo de afinação 56
6.3 Relação de gastos para o desenvolvimento do projeto 59
6.4 Usabilidade do sistema 60
7 CONCLUSÕES 61
8 REFERÊNCIAS 62
APÊNDICE A - Frequência e número MIDI das notas na escala temperada 65
ANEXO A – Algoritmo implementado no Arduino UNO 76

1 INTRODUÇÃO

A música esteve presente nas diversas culturas espalhadas pelo mundo, pois ela desenvolve a mente, oferece bem-estar e melhora a concentração (TRIMBLE; HESDORFFER, 2017). Para a música proporcionar tais benefícios, a afinação dos instrumentos que a compõe é extremamente relevante, pois a correta afinação é um dos aspectos que diferencia um som agradável de um ruído.

Os métodos de afinação ao decorrer da história evoluíram muito: afinação através dos ouvidos, diapasão, afinadores eletrônicos e, mais recentemente, os afinadores automáticos. Afinar um instrumento não é simples; sem o auxílio de um aparelho eletrônico se requer habilidade e muito conhecimento, sendo estas algumas das dificuldades para os iniciantes no mundo da música. Além disso, as cordas de um instrumento precisam ser afinadas regularmente; uma vez que as variações da temperatura alteram o comprimento das cordas (BERNI, 2016). Por isso, um afinador automático é a solução mais prática e rápida, podendo ainda ser operado por qualquer pessoa.

Existem dois modelos de afinadores automáticos, onde cada um utiliza um fator diferente para o controle da afinação, sendo estes: 1) baseados na determinação da frequência de vibração da corda; 2) baseados na mensuração da tensão sob qual a corda está submetida. O primeiro modelo é o mais comum, e uma das primeiras patentes deste tipo foi criada por Hendrick (1978), o qual utilizava um microfone para receber o sinal da corda (HENDRICK, 1978); posteriormente, captadores magnéticos começaram a ser usados para a mesma função. Já no segundo modelo, tem-se o dispositivo criado por Scholz (1984) (SCHOLZ, 1984). Entretanto, esse método não foi muito eficiente (SERAFINA, 2015), uma vez que, considera-se que a frequência produzida pela corda depende somente da tensão mecânica, quando depende, também, do comprimento e das características intrínsecas da corda (material e espessura) (FERRARO; SOARES, 2003).

O TronicalTune e o RoadieTuner são uns dos afinadores automáticos disponíveis no mercado; os que foram desenvolvidos pela Tronical e pela Band Industries, respectivamente. A Tronical, fundada em 2005 na Alemanha, é a empresa líder de mercado e pioneira tecnológica no ramo de afinadores automáticos. Por outro lado, a Band Industries foi fundada em 2012 e só em 2014 conseguiu lançar a primeira campanha Kickstarter, uma plataforma de *crowdfunding*, para o desenvolvimento do RoadieTuner. Tais marcas possuem aparelhos que afinam a partir de um microfone e, os mais recentes, a partir de um sensor de vibração, o qual possibilita a afinação em ambientes barulhentos. O RoadieTune está crescendo muito no mercado e

recebendo críticas positivas quanto a facilidade de uso e a precisão da afinação. Já as críticas quanto aos produtos da Tronical são em relação ao preço ser muito elevado, a partir de U$200, não compensando o custo-benefício.

O protótipo proposto neste trabalho consiste de um suporte de fácil manuseio que acopla o sistema à tarraxa do instrumento, sendo possível afinar uma corda de cada vez. Ao tocar a corda, um microfone capta o som propagado e o transforma em um sinal elétrico que é recebido por uma entrada analógica do Arduino UNO, o qual está programado com um algoritmo capaz de estimar a frequência fundamental, ou *pitch*, do sinal de entrada. Em seguida, o *pitch* calculado é comparado com uma frequência conhecida (conforme a tabela MIDI (ANEXO A)), que corresponde com uma nota musical a ser escolhida através de um *software*; tal nota é a que se deseja igualar (afinar). Esta comparação gera um erro, o qual pode ser tratado de duas maneiras: ou através de um controlador PID digital ou através de um controlador feito com própria equação do sistema. O controlador torna-se responsável por calcular qual será o ajuste necessário para o som propagado pela corda se tornar equivalente ao desejado. O ajuste é formado pela direção de rotação e a quantidade de passos a serem executados por um motor de passo acoplado ao suporte encaixado à tarraxa do instrumento. Esse processo se repete até o erro ser igual a zero ou mínimo.

O método de processamento de sinais utilizado para determinar o *pitch* foi a autocorrelação, o qual atua no domínio do tempo. A validação do resultado de precisão obtido pelo sistema foi através de um aplicativo para *smartphones* chamado Ukulele Tuner, disponível na PlayStore, o qual utiliza o microfone do *smartphone* para captar um áudio e em seguida mostra na tela a frequência calculada.

Para validar os resultados de tempo, foram realizados testes comparando os dois métodos de controle analisando qual atingiu o valor de referência mais rapidamente. E outro teste de tempo foi uma comparação entre o tempo entre a afinação manual e a automática, assim mostrando que o sistema visa a ser mais rápido e prático.

Este trabalho apresentará os principais conceitos necessários para o desenvolvimento do projeto. No Capítulo 4 é apresentado conceitos sobre a teoria musical, métodos para a estimação do *pitch* e métodos para o controle do sistema, e o Capítulo 5 contém o desenvolvimento do projeto. O Capítulo 6 apresenta os experimentos e resultados e o Capítulo 7 as conclusões. As referências estão presentes no Capítulo 8.

2 OBJETIVOS

A seguir serão apresentados os objetivos geral e específicos cumpridos no projeto.

2.1 OBJETIVO GERAL

Projetar e implementar um sistema de controle digital que seja capaz de alterar a tensão mecânica proporcionada à uma corda, com o intuito de afinar automaticamente uma corda de um instrumento acústico em uma determinada nota musical.

2.2 OBJETIVOS ESPECÍFICOS

- Desenvolver um sistema que permita afinar uma corda de um instrumento de cordas dentro de um intervalo de frequência com precisão aceitável, dentro do limiar de discriminação de frequência do ouvido humano (até 10 Hz de diferença);
- Obter um processo de afinação com tempo máximo aceitável, até 30 segundos;
- Projetar um sistema de fácil manuseio;
- Desenvolver um sistema cujos elementos tenham custo baixo;
- Desenvolver uma interface de *software* de fácil uso;
- Possibilitar a adaptação do sistema para outras configurações desejadas pelo usuário, como aumentar o número de cordas e alterar a nota desejada.

3 JUSTIFICATIVA

A motivação para a realização desse trabalho foi a compreensão de que afinar um instrumento sem o auxílio de aparelhos eletrônicos é complicado, demorado e requer certo conhecimento musical e prática. Portanto, o desenvolvimento de um afinador automático ajudará músicos que desejam afinar seus instrumentos rapidamente e também pessoas que não possuem a habilidade de afinar um instrumento somente com o ouvido.

4 REFERENCIAL TEÓRICO

Neste capítulo serão apresentadas as fundamentações teóricas necessárias para o desenvolvimento e entendimento do projeto; iniciando com os conceitos sobre teoria musical, seguido pelos métodos de estimação do *pitch*, métodos para controle de sistemas dinâmicos e os métodos para a identificação de sistemas.

4.1 TEORIA MUSICAL

Nesta seção serão apresentados os conceitos, em relação à teoria musical, necessários para a compreensão das etapas de desenvolvimento do projeto, sendo estas: as notas musicais, suas respectivas notações e as escalas musicais.

4.1.1 Notas musicais e suas respectivas notações

Na música ocidental existem sete notas musicais. Tais notas são representadas ou por monossílabos, ou por letras. A representação por monossílabos foi introduzido por Guido d'Arezzo no século XI, o qual inspirou-se em um hino cantado em louvor a São João Batista. Tal hino possui sete versos; com isso, a primeira sílaba de cada verso foi adotada como uma nota musical (COTTA, 2008). A Tabela 1 mostra as nomenclaturas de cada nota.

Tabela 1 - Nome das notas musicais e nomenclatura.

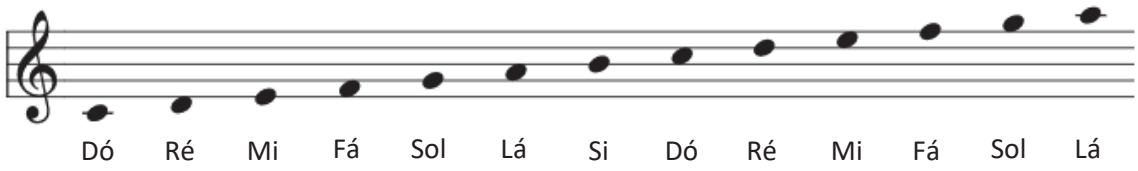
Países Latinos	Inglês
Dó	C
Ré	D
Mi	E
Fá	F
Sol	G
Lá	A
Si	B

Fonte: do autor.

A partitura é a representação escrita de uma música. As notas são representadas como um desenho no formato oval e são escritas em pautas de cinco linhas. A Figura 1 mostra uma

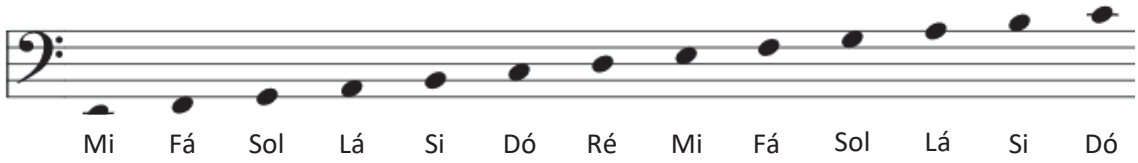
pauta na clave de Sol[1] com cada nota representada e a Figura 2 mostra uma pauta na clave de Fá[2] (claves mais utilizadas) (MED, 1996).

Figura 1 - Notação musical na clave de Sol



Fonte: do autor.

Figura 2 - Notação musical na clave de Fá



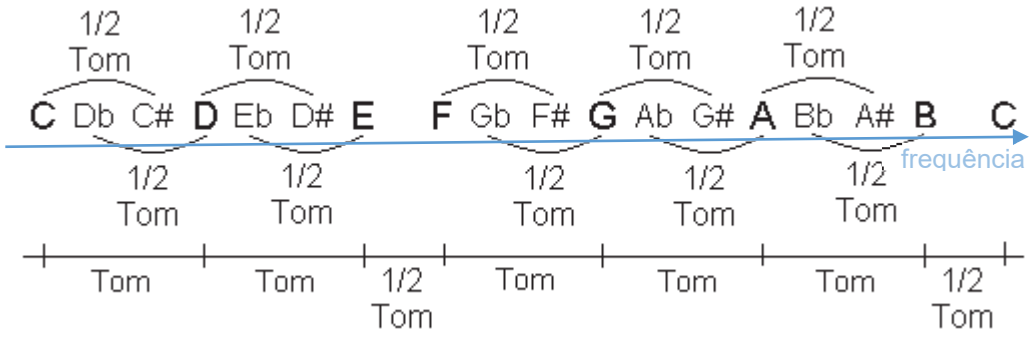
Fonte: do autor.

4.1.2 Escalas musicais

As escalas musicais são a divisão da sequência de notas que existem dentro de uma oitava[3] (SOUZA, 2009). Na música ocidental existem duas escalas musicais: a natural e a temperada. A escala natural define com precisão, através de cálculos acústicos, o número de vibrações para cada nota e a relação entre elas. A Figura 3 representa as divisões dentro de uma oitava na escala natural.

[1] Própria para grafar notas agudas (violino, flauta, oboé, canto, etc.) (MED, 1996).
[2] Indicada para grafar notas mais graves (violoncelo, contrabaixo, trombone, etc.) (MED, 1996).
[3] A frequência de um som é exatamente o dobro da frequência de outro som. Desta definição tem-se que os dois sons são a mesma nota, mas com oitavas diferentes. A nota com maior frequência está uma oitava acima da nota com menor frequência (FERRARO; SOARES, 2003).

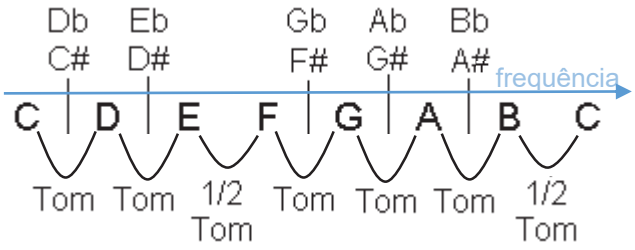
Figura 3 - Escala natural.



Fonte: do autor.

Os tons e semitons representam a diferença de altura de uma nota para outra. Um intervalo de dois semitons são equivalentes a um tom. Se o semitom está à direita de sua respectiva nota (maior altura) acrescenta-se à nomenclatura o símbolo "#" (sustenido), ou acrescenta-se o símbolo "♭" (bemol) se o semitom está à esquerda (menor altura). Na escala natural, o tom é dividido em nove partes iguais, as quais são denominadas comas. Com isso, foi estabelecido que os semitons estão à cinco comas de distância de sua respectiva nota. Isso faz com que a diferença entre um C# e um D♭, por exemplo, seja de apenas uma coma[4], o que é praticamente imperceptível. Para facilitar, foi criada a escala temperada, a qual iguala os semitons em partes perfeitamente iguais e equivalente a quatro comas e meia; isso produz uma equivalência entre as notas que na escala natural estava por apenas uma coma (MED, 1996). O padrão de afinação adotada neste projeto é a temperada, a Figura 4 mostra as divisões desta escala.

Figura 4 - Escala temperada.



Fonte: do autor.

[4] O intervalo de 1 coma corresponde à uma relação igual a 1,0129 entre duas frequências. A quantidade de ciclos por segundo contido em uma coma é variável ao longo de uma mesma oitava (NETTO, 2018).

4.1.3 Afinação musical

A afinação musical corresponde à reprodução de um som com frequência igual a outra tomada como referência. Em um violão, a frequência é ajustável através da tensão das cordas, quanto mais tensionada, mais agudo será o som reproduzido e maior será a frequência de vibração da corda. Quando a frequência da corda atingir a frequência da nota desejada, ela estará afinada.

Quando algum mecanismo altera a pressão do ar, é produzido um som; essa variação da pressão sobre a massa de ar produz diferentes sons. A vibração de determinados materiais é transmitida às moléculas de ar sob a forma de ondas sonoras, e quando essa vibração ocorre de maneira repetitiva, forma-se um tom com uma altura igual à sua frequência. Uma nota musical é um som cuja frequência de vibração encontra-se dentro do intervalo perceptível pelo ouvido humano[5] (SOUZA, 2009).

A frequência sonora que uma corda produz, quando é estimulada, é caracterizada por três fatores: o comprimento da corda, a densidade linear (relação entra a massa da corda e o comprimento) e a tensão aplicada à corda. A fórmula para determinar a frequência de vibração (f_n) de uma corda é:

$$f_n = n\frac{1}{2L}\sqrt{\frac{F}{\mu}}, \tag{1}$$

onde n é o harmônico desejado, L é o comprimento da corda, F é a tensão que a corda está submetida e μ é a densidade linear da corda. Portanto, para determinar a frequência fundamental, n deve ser igual a 1 (FERRARO; SOARES, 2003). De outro modo, a frequência (f_i) das notas musicais é calculada a partir de uma frequência de referência (f_0):

$$f_i = f_0 \cdot 2^{\frac{i}{12}}, \tag{2}$$

onde i é o intervalo de semitons contido no intervalo entre a nota de referência e a nota que deseja-se obter a frequência. Relacionando a afinação temperada com o padrão MIDI[6], foi criado um formato composto de 128 notas espalhadas em várias oitavas, adotando a nota A4[7] (440 Hz) como referência. A fórmula para determinar a frequência de cada uma das 128 notas no formato MIDI é (WOLFE, 1997):

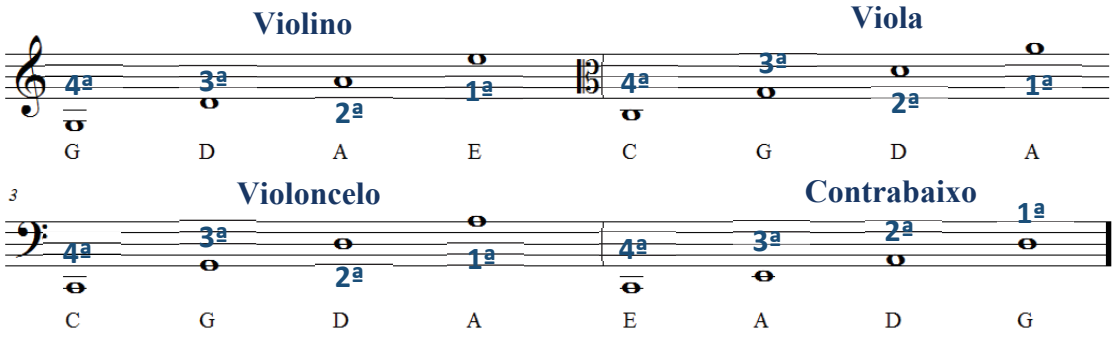
$$f_N = f_0 \cdot 2^{\frac{N-69}{12}}, \tag{3}$$

[5] O ouvido humano percebe sons cuja frequência se limita ao intervalo entre 20 e 20.000 Hz (SOUZA, 2009).
[6] MIDI (Musical Instrument Digital Interface) é um padrão de comunicação entre instrumentos, criado no início dos anos 80.
[7] A4 é a nota de número MIDI igual a 69.

onde N é o número MIDI da nota, f_N é sua respectiva frequência e f_0 é a frequência fundamental adotada como referência (A4 em 440 Hz). A Tabela A-1 contida no ANEXO A contém a frequência e o respectivo número MIDI de cada uma das notas musicais nas primeiras 9 oitavas.

Cada instrumento possui uma afinação padrão, que define a nota que cada uma das cordas emitirá quando tocadas isoladamente. Nos instrumentos de corda, geralmente são usados os intervalos de quarta justa ou quinta justa entre uma corda e outra. O intervalo de quarta justa se dá à nota que está a 5 semitons acima da respectiva nota; por exemplo, a quarta justa da nota C é a nota F. Já a quinta justa, se dá à nota que está a 7 semitons acima da respectiva nota; por exemplo, a quinta justa da nota C é G (MED, 1996). O violino, a viola, o violoncelo e o contrabaixo acústico são exemplos de instrumentos que são afinados com intervalo de quarta justa. No caso do violão, as cordas são afinadas com intervalos de quinta justa. A Figura 5 representa as pautas musicais com a nota respectiva de cada corda dos instrumentos, bem como o número de cada corda. A maioria dos instrumentos de corda seguem a afinação linear, onde a primeira corda é a mais aguda e a última corda é a mais grave. Uma exceção é o ukulele, pois possui uma afinação reentrante, onde a última corda, que deveria ser a mais grave, é afinada em uma oitava acima, tornando assim, a terceira corda a mais grave; isso é responsável pela característica única do som deste instrumento. A seguir, são mostradas as frequências das afinações padrões do violino, viola, violoncelo e contrabaixo acústico (Tabela 2).

Figura 5 - Pautas musicais com as respectivas notas de cada corda do violino, viola, violoncelo e contrabaixo.



Fonte: do autor.

Tabela 2 - Afinações padrões dos instrumentos de corda (Hz).

Corda	Violino (Hz)	Viola (Hz)	Violoncelo (Hz)	Contrabaixo acústico (Hz)
1	659,26	440	220	98
2	440	293,66	146,83	73,42
3	293,66	1196	98	55
4	196	130,81	65,41	41,2

Fonte: (BENNETT, 1986).

4.1.4 Ukulele

O ukulele é um instrumento que se originou no fim do século XIX no Havaí. Foi inspirado em um instrumento português chamado Braguinha, o qual chegou à ilha junto ao navio Português Ravenscrag. Marceneiros que estavam a bordo do navio observaram que uma madeira da região, a *koa*, seria ótima para a produção de um novo instrumento. O instrumento foi produzido e adotado pela ilha incorporando-o à sua cultura definitivamente (KING; TRINQUADA, 2003).

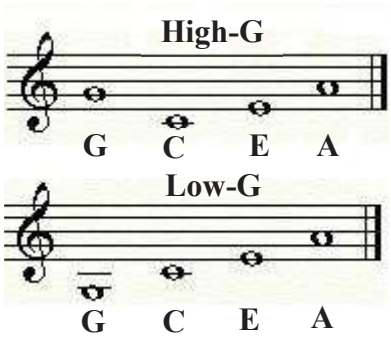
Existem quatro tipos de ukulele, o soprano (Figura 6), o concerto, o tenor e o barítono. A diferença entre eles é o tamanho, portanto, diferenciam-se a tessitura[8] e o registro[9] de cada instrumento. O soprano é o menor (mais agudo) e o barítono o maior (mais grave). Todos possuem quatro cordas, porém a afinação do barítono é diferente. A afinação padrão dos três menores modelos é reentrante, com a quarta corda afinada uma oitava acima do padrão de uma afinação linear. As notas das cordas, da 4ª para a 1ª são: G4, C4, E4 e A4; essa afinação também é chamada *high-G*. Todavia, há uma outra configuração de afinação muito usada, onde a 4ª corda é afinada em G3, tornando assim, a afinação linear, chamada *low-G*; é mais comumente encontrada em ukuleles tenores, e geralmente é utilizada para dar mais força ao som quando não há outros instrumentos acompanhando o ukulele. A Figura 7 apresenta a pauta das duas afinações mais comuns do ukulele e a Tabela 3 apresenta a frequência das notas nessas afinações.

Figura 6 - Ukulele soprano.

Fonte: do autor.

[8] A tessitura de um instrumento musical é a extensão de notas que ele pode tocar. Por padronização identifica-se a tessitura através do nome e da oitava da nota mais grave e da mais aguda que um instrumento pode executar.
[9] Chamam-se registros as três regiões em que a tessitura de um instrumento pode ser dividida. Divide-se em registro grave, médio e agudo.

Figura 7 - Afinações do Ukulele.



Fonte: do autor.

Tabela 3 - Frequência das notas das afinações *high*-G e *low*-G do ukulele.

Afinação	Corda 4	Corda 3	Corda 2	Corda 1
***High*-G (Hz)**	392	262	330	440
***Low*-G (Hz)**	196	262	330	440

Fonte: do autor.

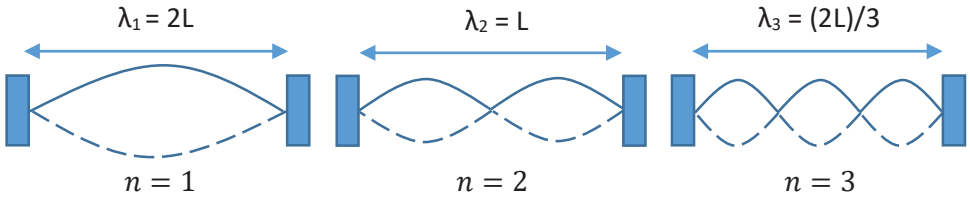
4.3 FREQUÊNCIA FUNDAMENTAL

Instrumentos diferentes reproduzindo uma mesma nota não soarão iguais; isso se deve ao fato de que um som, na verdade, é uma mistura de sons (SOUZA, 2009). Produzindo-se uma perturbação em qualquer ponto entre os extremos fixos de uma corda de comprimento L, é provocada uma onda que propaga-se até cada uma das extremidades, se refletindo em sentido contrário. Assim, formam-se ondas estacionárias com nós (pontos que não vibram) e ventres (distância entre dois nós, onde todos os pontos estão em movimento vibratório). A onda estacionária de frequência mais baixa é chamada frequência fundamental; ela corresponde a uma onda estacionária com um único ventre, o harmônico fundamental ou primeiro harmônico. As demais frequências naturais são chamadas sobretons ou harmônicos superiores, pois as frequências subsequentes são múltiplos inteiros da frequência fundamental (SERWAY; JEWETT, 2004). Os comprimentos de onda λ, para os quais a onda estacionária acontece, correspondem às frequências de ressonância da corda. Na Figura 8 é mostrada a relação entre os harmônicos n, o comprimento de onda λ_n e o comprimento L da corda. A partir disso, buscando generalizar uma fórmula entre os harmônicos e o comprimento de onda, tem-se a Eq. (4) (SERWAY; JEWETT, 2004):

$$\lambda_n = \frac{2L}{n}, \tag{4}$$

(4)

Figura 8 – Relação entre harmônicos, comprimento de onda e comprimento da corda: a) primeiro harmônico; b) segundo harmônico; c) terceiro harmônico.



Fonte: do autor.

A equação fundamental da ondulatória, Eq. (5), afirma que a velocidade (V) com que a onda é propagada na corda é igual ao produto entre o comprimento de onda λ e a frequência de oscilação (f) da onda na corda (HALLIDAY; RESNICK; WALKER, 2012):

$$V = \lambda \cdot f. \tag{5}$$

Outra fórmula relevante à determinação da frequência dos harmônicos em uma corda vibrante é a fórmula de Taylor (Eq. (6)), a qual descreve que para a propagação de uma onda periódica em uma corda, a velocidade V com que a onda se propaga depende da densidade linear de massa μ da corda e da intensidade da força tensora T a que ela está sujeita (SERWAY; JEWETT, 2004):

$$V = \sqrt{\frac{T}{\mu}}. \tag{6}$$

Substituindo a Eq. (4) em Eq. (5) e posteriormente em Eq. (6), tem-se a Eq. (1).

A vantagem de produzir ondas estacionárias é que, nessas condições, a corda passa a oscilar com grande amplitude, movimentando periodicamente o ar ao redor e produzindo assim uma onda sonora audível com a mesma frequência que as oscilações da corda. Quando diferentes instrumentos tocam a mesma nota, produzem a mesma frequência fundamental, mas os harmônicos superiores têm intensidades diferentes (HALLIDAY; RESNICK; WALKER, 2012).

4.3.1 Limiar de discriminação de frequência auditiva

O ouvido humano é bastante sensível a diferenças de frequências entre dois sons. Em sons graves mudanças de frequência de 1 Hz podem ser detectadas. Aos 1000 Hz a maior parte das pessoas é capaz de distinguir mudanças na frequência com o valor de 3 Hz. Aos 100 Hz mudanças na frequência podem ser notadas a partir dos 0,3 Hz. Ou seja, o ouvido é sensível não propriamente a mudanças absolutas da frequência, mas sim a uma razão entre a zona que

frequências do som que se está ouvindo e da mudança efetuada, e quanto maior a frequência, maior será o limiar de discriminação no ouvido humano (CARNEIRO, 2003).

A membrana basilar é responsável pelo processo de percepção do som, perfazendo uma análise das frequências componentes de um som. Diferentes partes da membrana basilar são sensíveis a diferentes frequências puras. Quando dois sons puros, com frequências e amplitudes diferentes, são tocados juntos e chegam ao ouvido, é percebido uma variação na intensidade do som resultante; ela aumenta e diminui alternadamente, produzindo um fenômeno chamado batimento. Tal fenômeno é resultante da interferência construtiva e destrutiva das duas ondas quando ficam em fase ou em oposição de fase. Se as duas frequências ficarem próximas, o batimento ficará gradualmente mais lento (a frequência do batimento é a subtração das frequências das ondas) e desaparecerá quando elas forem idênticas (uníssono). Os batimentos entre dois tons podem ser percebidos pelo ouvido humano até uma frequência de 15 Hz. Quando as frequências são superiores a 15 Hz os batimentos individuais não podem ser distinguidos (MONTEIRO, 2006). Até o ponto de aproximadamente 10 Hz, o ouvido percebe somente sinais separados, pulsações rítmicas, sendo realmente interpretado como uma melodia a partir de 15 Hz; portanto, cria-se um limiar de erro em uma mesma nota de até 10 Hz (MARÇOLLA, 2015).

4.3.2 Estimação da frequência fundamental

Uma das tarefas mais difíceis na análise de sinais é a determinação do *pitch* de um sinal periódico; uma vez que, certamente apresentará erros na detecção. Por exemplo, um erro de 6%, consideravelmente pequeno, em um semitom, muda completamente uma frase musical[10]. Uma precisão aceitável de um detector de frequência é de 0,001%, ou 1 Hz; porém, para isso ser possível, a forma de onda não deve conter ruídos, já que o detector é sensível aos pontos que o sinal cruza o zero (inversão de sinal) (CHAMBERLIN, 1987).

Existem duas categorias de métodos para determinar o *pitch* de um sinal, métodos através do domínio do tempo e métodos através do domínio da frequência. Dentro dessas categorias existem diversos métodos para estimação do *pitch*, como a Autocorrelação, *Simplified Inverse Filter Technique* (SIFT), *Cepstrum*, *Harmonic Product Spectrum* (HPS), *Maximum Likelihood* (ML), entre outros (COCA, 2004). A seguir será apresentado o método da Autocorrelação (domínio do tempo).

[10] Trecho de um solo musical.

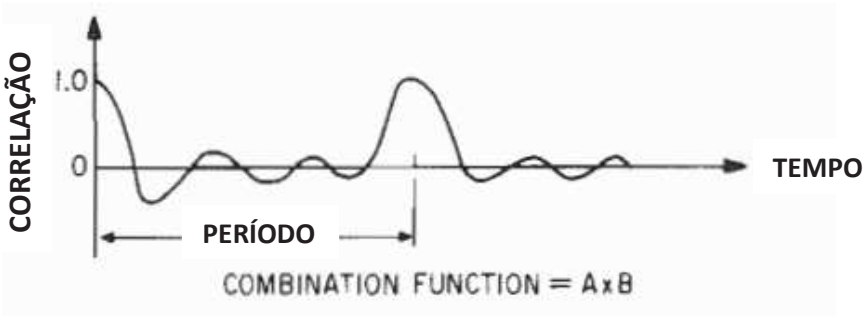
4.3.2.1 Método da Autocorrelação

A Autocorrelação é uma medida de similaridade de um sinal acústico com uma cópia do mesmo sinal atrasado no tempo. Na prática, o atraso começa em zero e é aumentado até a correlação atingir um valor máximo, o que indica um atraso do tamanho do período do sinal (CHAMBERLIN, 1987). A função para obter a o fator de correlação $r[m]$ é (SERAFINA, 2015):

$$r[m] = \frac{1}{M}\sum_{k=0}^{M-m-1} x[k]x[k+m], \quad (7)$$

onde m é o instante de atraso, M é o número de amostras do sinal e $x[k]$ é o sinal no instante k. O valor máximo da correlação é quando m é igual a zero. O instante do próximo valor máximo da correlação será uma estimativa do período fundamental. A frequência fundamental será o inverso do período estimado (SZWOCH, 2000). A Figura 9 apresenta um exemplo da autocorrelação de um sinal.

Figura 9 - Autocorrelação de um sinal.



Fonte: (CHAMBERLIN, 1987)

O tamanho da janela do sinal a ser usado no método da autocorrelação varia de acordo com o sinal, o ideal é conter de 2 a 3 períodos completos. Deste modo, para sinais com períodos curtos, uma janela de 5 a 20 ms é suficiente; e para sinais com períodos longos, janelas de 20 a 50 ms são indicadas (RABINER, 1977).

Em teoria, a autocorrelação produz picos de amplitude máxima apenas nos instantes múltiplos do período do sinal. Por isso, qualquer sinal perfeitamente periódico será corretamente analisado pelo método da autocorrelação. Porém, um sinal que possui uma forma de onda variável, este método não será efetivo. Quando a forma de onda muda, o pico correspondente ao período é menor do que deveria, devido a repetição inexata do sinal. Há, também, picos adicionais que podem ser proveniente de harmônicos, formantes ou ruído. Se esses picos forem grandes o suficiente para confundir com o pico máximo no instante zero do sinal, poderá causar erros na detecção do *pitch*. (CHAMBERLIN, 1987). Logo, o método da

autocorrelação é mais indicado para a análise em sinais de áudio de uma nota musical do que em um sinal de áudio de voz.

4.4 CONTROLE DE SISTEMAS DINÂMICOS

A palavra controle refere-se ao ato de produzir um resultado desejado (TEWARI, 2002). Existem diversos exemplos de sistemas de controle: controle de velocidade, posição, temperatura, luminosidade, etc. A seguir, serão explicadas as técnicas do controlador que será utilizado.

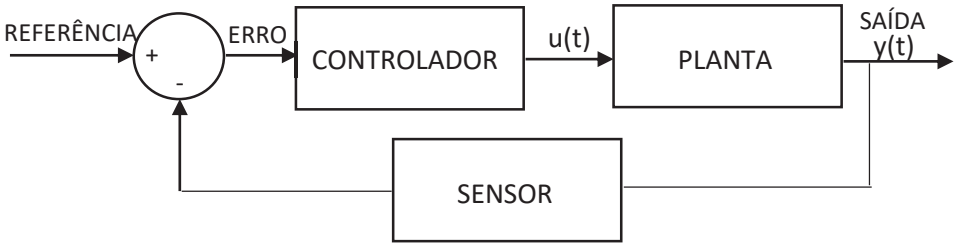
4.4.1 Definições

Para entender sistemas de controle, é importante definir algumas terminologias básicas (OGATA, 2002):

- Sistemas: em controle, um sistema é a combinação de componentes que atuam juntos e realizam um objetivo específico.
- Variável de saída e variável de entrada: a variável de saída é a condição que deve ser medida e controlada pelo sistema de controle. A variável de entrada é a condição que é variada pelo controlador, de modo que manipule o valor da variável de saída.
- Valor de referência: é uma entrada do controlador que determina qual o valor desejado na saída do sistema de controle. O projeto do controlador é conseguir com que a saída do sistema de controle atinja este valor de referência.
- Planta: função de transferência do sistema a ser controlado. É onde o controlador atuará, modificando a resposta do sistema.

Um sistema de controle pode ser ou de malha aberta ou malha fechada. Neste projeto se trata de um controle em malha fechada. Tal conceito implica que haverá uma realimentação no sistema de controle; ou seja, a saída do sistema é mensurada por um sensor adequado e será, também, uma entrada do sistema, onde será feita uma comparação da saída do sistema com a outra entrada do controlador, o valor de referência (*setpoint*). Desta comparação será gerado o erro, o qual é utilizado no cálculo do sinal de controle realizado pelo controlador e enviado ao atuador da planta (OGATA, 2002). Quando o valor do erro for alto, implica que o valor da saída do controlador não está perto do valor desejado. Com isso, o controlador atua novamente, até o erro estar dentro de um valor aceitável. A Figura 10 traz um diagrama de blocos de um sistema de controle em malha fechada.

Figura 10 - Sistema de controle em malha fechada.



Fonte: do autor.

A função de transferência da planta é obtida através de uma modelagem matemática; segundo Ogata (2002), modelagem matemática de sistemas é o desenvolvimento de um conjunto de equações que descrevem a dinâmica deste sistema. Estas equações vêm da própria teoria dos elementos presentes nele; por exemplo em um sistema elétrico serão usadas as leis de Kirchhoff para correntes e tensões elétricas, que são as leis básicas que regem um circuito. Porém, existem os casos onde não existe um conhecimento satisfatório sobre a física da planta para chegar a um modelo suficientemente bom para representá-lo matematicamente, nestes casos é usada a chamada modelagem em caixa-preta *(black-box modeling*) onde o modelo é inteiramente obtido através de experimentos (OGATA, 2002).

4.4.2 Controlador PID

O controlador PID (Proporcional Integral Derivativo) é a junção de três controladores: o proporcional, o integral e o derivativo. Tal controlador possui a seguinte função de transferência (OGATA, 2002):

$$G(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right), \tag{8}$$

onde K_p é o ganho proporcional, T_i é chamado de tempo integral e T_d é o tempo derivativo. Cada uma dessas constantes são responsáveis por manipular uma característica na resposta de um sistema. Quanto maior o ganho proporcional K_p, maior a velocidade de resposta do sistema, porém, mais oscilatório (provoca uma instabilidade no sistema). O tempo integral T_i é responsável por eliminar o erro[11] em regime permanente. Já o tempo derivativo T_d é responsável por tornar o sistema mais estável (CHEN, 1993).

[11] Erro de estabilização, a resposta do sistema estabiliza em um ponto diferente do *setpoint*.

4.4.3 Controlador PID digital

O PID apresentado anteriormente é um controlador analógico, mas para poder trabalhar com sinais amostrados é necessário adaptar tal controlador. A transformação do controlador PID analógico para digital é obtida pela discretização da Eq. (8) no domínio do tempo. A componente integral é substituída por uma somatória, e a derivada por uma diferença de primeira ordem. Desta forma o cálculo do sinal de controle do PID digital, na sua forma recursiva é dada pela Eq. (9), onde $e(k)$ é o erro no instante de tempo k, T_s é o tempo de amostragem e K_p, T_d e T_i são o ganho proporcional, tempo derivativo e tempo integral, respectivamente (CHEN, 1993).

$$u(k) = u(k-1) + q_0 e(k) + q_1 e(k-1) + q_2 e(k-2) \tag{9}$$

Onde,

$$q_0 = K_p \left[1 + \frac{T_d}{T_s} + \frac{T_s}{T_i}\right] \tag{10}$$

$$q_1 = -K_p \left[1 + 2\frac{T_d}{T_s}\right] \tag{11}$$

$$q_2 = K_p \frac{T_d}{T_s} \tag{12}$$

4.4.4 Métodos para projeto de PID

Várias regras foram desenvolvidas para ajustar os três parâmetros do controle PID (K_p, T_i e T_d). Nesta subseção serão mostrados os métodos de Ziegler-Nichols para a sintonização do PID e da identificação de sistemas.

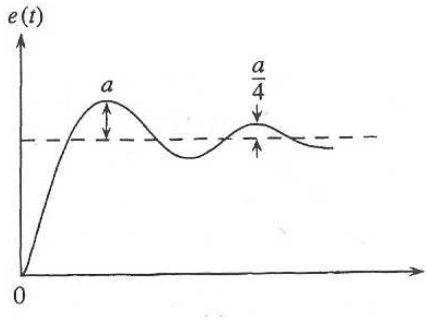
4.4.4.1 Sintonização de Ziegler-Nichols

Esse método é empregado em sistemas que tem uma resposta ao degrau unitário semelhante à Figura 11, na qual a amplitude do segundo sobressinal é cerca de 25% da amplitude do primeiro sobressinal (CHEN, 1993).

Tal método caracteriza-se pelos seguintes passos:

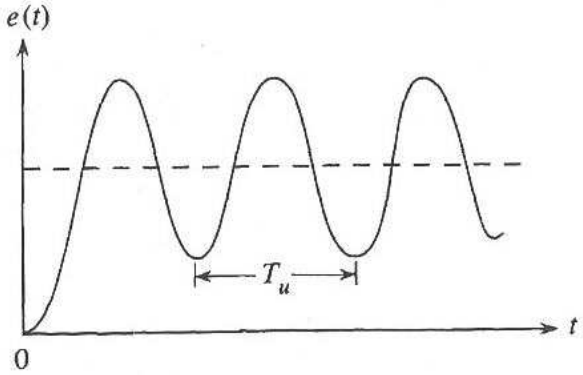
1. Reduzir as ações integral e derivativa ao seu efeito mínimo;
2. Iniciar o processo com ganho reduzido;
3. Aumentar o ganho até que a variável controlada (saída do sistema) entre em oscilações com amplitude constante (o ganho ajustado é denominado K_u). Anotar o ganho K_u e o período de oscilação T_u, como mostrado na Figura 12. Por fim, os parâmetros são calculados de acordo com a Tabela 4.

Figura 11 - Resposta ao degrau de um sistema.



Fonte: CHEN, 1993.

Figura 12 - Exemplo de resposta transitória com oscilação constante.



Fonte: CHEN, 1993.

Tabela 4 - Parâmetros dos controladores pelo método de Ziegler-Nichols.

Controlador	K_p	T_i	T_d
P	$0{,}5K_u$		
PI	$0{,}4K_u$	$0{,}8T_u$	
PID	$0{,}6K_u$	$0{,}5T_u$	$0{,}12T_u$

Fonte: CHEN, 1993.

4.4.4.2 Identificação de sistema por Ziegler-Nichols

A identificação da planta pelo método experimental de identificação de sistemas de Ziegler-Nichols é aplicada em sistemas de primeira ordem. Primeiramente, é analisado a resposta ao degrau do sistema, contendo somente a entrada, a planta e a saída (COELHO; COELHO, 2004).

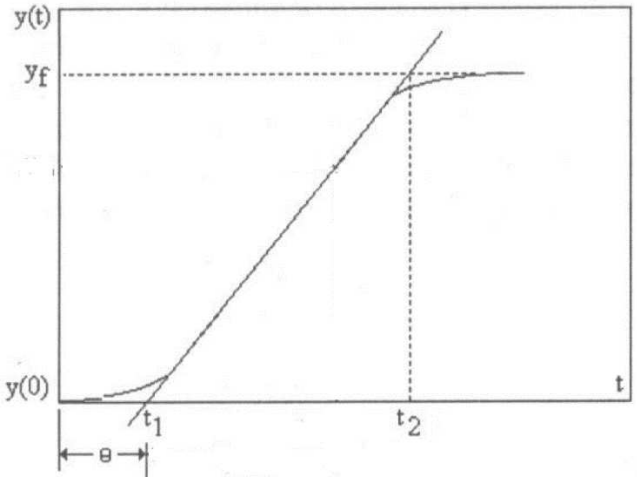
O gráfico da resposta ao degrau do sistema será analisada para estimar os seguintes parâmetros que compõem a função de transferência da planta de um sistema de primeira ordem:

o ganho K, a constante de tempo τ e o atraso de tempo θ. A função de transferência da planta G(s) é a razão da saída Y(s) pela entrada R(s) do sistema (COELHO; COELHO, 2004):

$$G(s) = \frac{Y(s)}{R(s)} = \frac{Ke^{-\theta s}}{\tau s+1} \tag{13}$$

O gráfico de resposta ao degrau terá o aspecto como mostra a Figura 13.

Figura 13 - Resposta ao degrau do sistema.



Fonte: (COELHO; COELHO, 2004).

O ganho K é obtido pela seguinte equação:

$$K = \frac{\Delta Y}{\Delta U}, \tag{14}$$

onde ΔY é a variação da amplitude da resposta de zero até o ponto de estabilização ($yf - y(0)$), e ΔU é a variação do degrau (entrada do sistema), no caso de um degrau unitário a amplitude varia de zero a um. Nota-se pelo gráfico que o atraso de tempo (θ) é o tempo que o sistema demora para responder ($t_1 - 0$). Para determinar a constante de tempo (τ), é desenhado a tangente no plano de máxima inclinação. O ponto onde a tangente atingir $y(0)$ é chamado de t_1 e o ponto onde a tangente atingir o valor máximo da resposta é chamada de t_2. Com isso, a diferença de t_2 e t_1 é o valor de τ (COELHO; COELHO, 2004):

$$\tau = t_2 - t_1. \tag{15}$$

Uma vez obtido todos os parâmetros, substituindo-os na Eq. 28 encontra-se a função de transferência da planta.

4.4.5 Modelagem matemática de sistemas

A modelagem matemática é um processo que envolve a obtenção de um modelo, é uma representação simplificada, porém tendo como característica o uso de um conjunto de símbolos e relações matemáticas. Dessa forma, representa um objeto ou um fenômeno estudado, ou ainda, um problema proveniente de uma situação real. O modelo produz informações importantes para a criação de alternativas de solução.

De acordo com a proposta de Bassanezi (2002), as etapas para a modelagem são:

- Experimentação: esta etapa é o primeiro contato com a situação problema envolvendo a coleta de dados.
- Abstração: nesta fase tem-se a formulação do modelo matemático e para tal é necessário estabelecer as seguintes ações básicas:
    - Seleção das variáveis: ao fazer as escolhas das variáveis, é necessário ter a clareza de quais são as variáveis de estado que descrevem a evolução de um sistema e as variáveis de controle que agem sobre o sistema.
    - Problematização: nesta etapa é fundamental gerar problemas ou questionamentos em uma linguagem própria, que em geral é diferente da linguagem da realidade. Etapa para identificar a relação existente entre as variáveis.
    - Formulação de hipóteses: as hipóteses dirigem o processo e são formulações gerais que permitem ao pesquisador deduzir detalhes específicos.
    - Simplificações: considerando a complexidade do contexto real, pode acontecer de um problema matemático não poder ser resolvido. Dessa forma, é necessário voltar ao problema original para fazer simplificações que automaticamente gerarão novas variáveis ou redução de variáveis. As hipóteses são revistas e o modelo refeito.
- Resolução: nesta etapa, encontrar a solução pode ser algo simples ou bastante complicado. Neste momento, os recursos computacionais são valiosos, pois podem auxiliar com soluções numéricas aproximadas.
- Validação: é o momento de verificar se o modelo proposto pode ser aceito ou não. Usando as hipóteses, o modelo deve ser testado com variáveis empíricas, comparando-se a solução apresentada pelo modelo com os valores obtidos na situação real. Em geral usa-se ferramentas gráficas para facilitar as previsões e para dar os devidos encaminhamentos de aperfeiçoamentos do modelo.

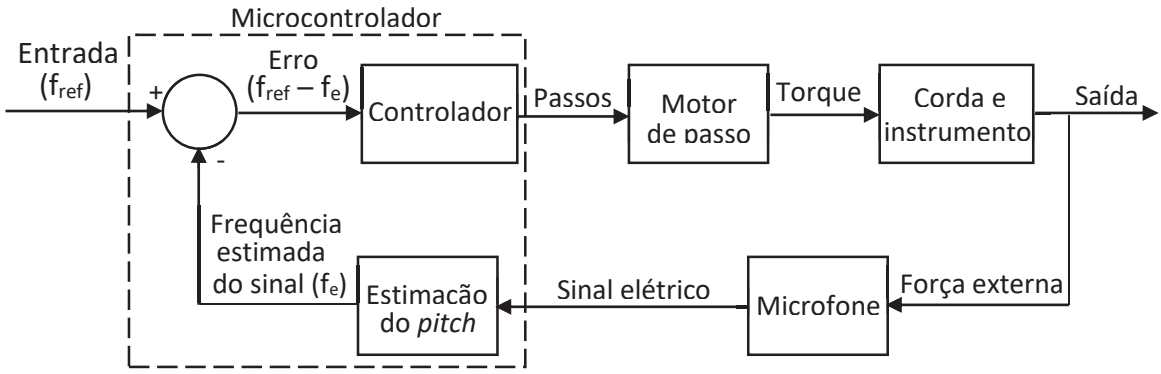
- Modificação: alguns fatores ligados ao problema inicial podem provocar a geração de modelos inadequados e que podem não ser aceitos. Dessa forma, alguma hipótese utilizada poderá ser falsa ou não adequada, ou alguns dados foram obtidos incorretamente, ou os dados e as hipóteses são verdadeiros, porém insuficientes para a solução.

5 DESENVOLVIMENTO

Nesta seção serão apresentados os procedimentos que foram necessários para o desenvolvimento do projeto; desde a captação do som do instrumento, a preparação deste sinal para ser interpretado pelo microcontrolador, os algoritmos que estimam a frequência fundamental do sinal, a identificação do sistema até o controle digital aplicado no sistema.

A Figura 14 representa um diagrama de blocos com cada etapa do projeto. Será mostrado nessa seção, também, a forma de utilizar o sistema; foi criado um *software* que serve como uma interface, ele se comunica com o Arduino e através dele é escolhida a nota de referência que se deseja afinar.

Figura 14 - Diagrama de blocos do projeto.



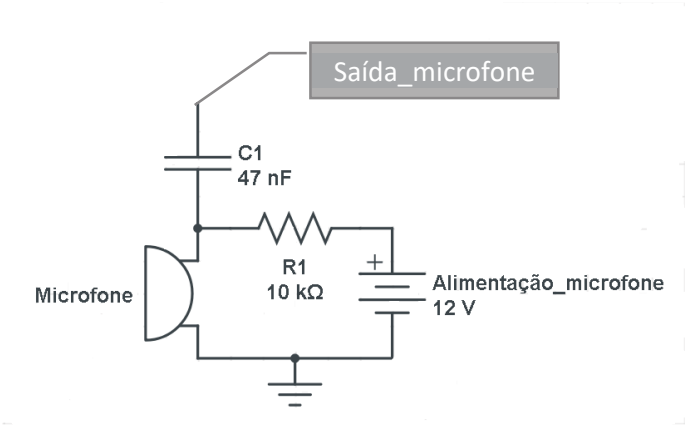
Fonte: do autor.

Os componentes utilizados neste projeto foram:

a. Microfone: O microfone de eletreto é o microfone utilizado no projeto; foi escolhido por ter excelente sensibilidade, pequeno tamanho e baixo custo. O funcionamento deste microfone tem como base a capacitância e a eletrostática (PEREIRA, [s.d.]). Para a ligação, requer uma alimentação externa e um circuito de polarização. O circuito de ligação do microfone de eletreto no projeto é mostrado na Figura 15.

b. Microcontrolador: Para facilitar a implementação do projeto, foi escolhido o *kit* de desenvolvimento Arduino UNO. O microcontrolador usado neste *kit* é o ATmega328P, as principais especificações deste componente estão presentes na Tabela 5.

Figura 15 - Circuito de ligação do microfone.



Fonte: do autor.

Tabela 5 - Especificações do microcontrolador ATmega328P do Arduino.

Memória programável (kB)	32
SRAM (Bytes)	2048
Data EEPROM (Bytes)	1024
Periféricos Comunicação Digital	1-UART, 2-SPI, 1-I2C
Periféricos *Capture*/*Compare*/PWM	1 Input Capture, 1 CCP, 6 PWM
Timers	2 x 8-bit, 1 x 16-bit
Tensão de operação (V)	1,8 a 5,5
Número de pinos	32

Fonte: Microchip, ATmega328P Datasheet.

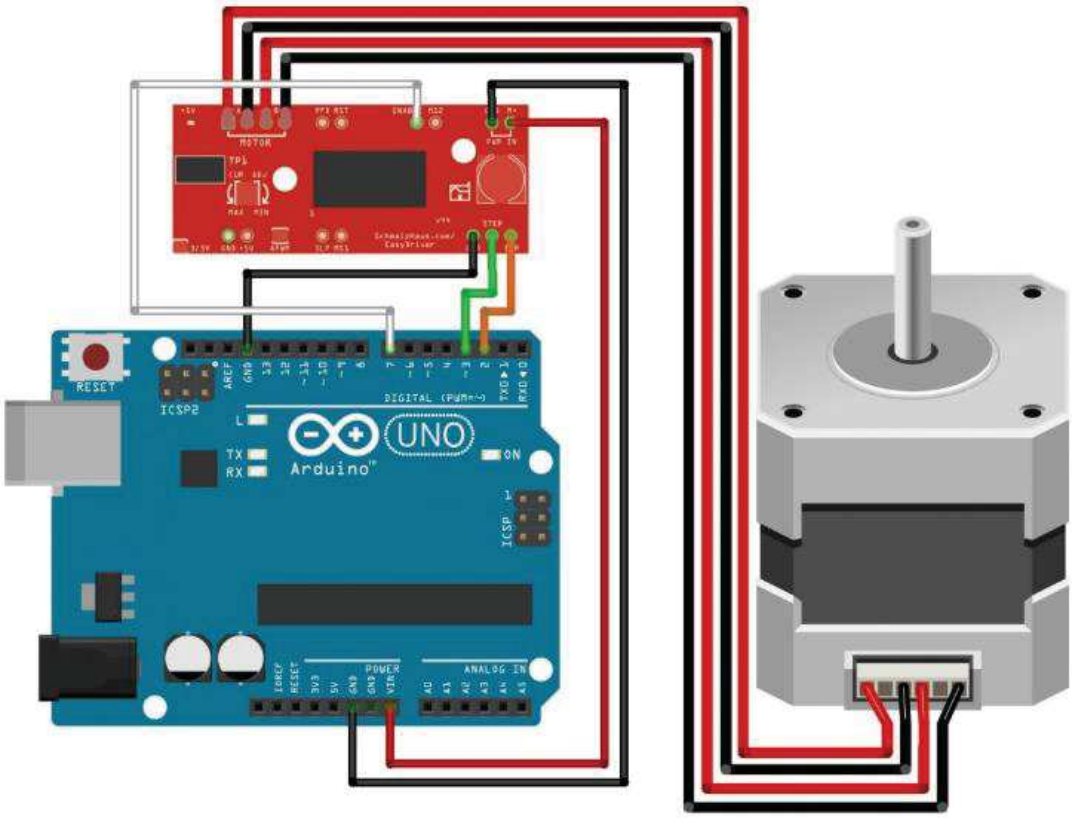
c. Motor de passo: O motor de passo empregado neste projeto é o modelo AK17 da marca Akiyama Motors. As especificações técnicas do motor são mostradas na Tabela 6. O *driver* utilizado para fazer a comunicação entre o motor de passo e o Arduino foi o EasyDriver. A saída deste *driver* é um pulso digital de 0 a 5 V e a tensão de alimentação é de 7 a 20 V. É compatível com motores de 4, 6 e 8 fios. Além disso, o EasyDriver oferece a opção resolução de passos: *full* (1/1), *half* (1/2), *quarter* (1/4) e *eight steps* (1/8). A resolução 1/8 é a configuração padrão do *driver*; portanto, o motor passa de 200 passos para 1600 passos necessários para dar uma volta completa (360º). O esquema de ligação entre o motor de passo, o EasyDriver e o Arduino é apresentado na Figura 16.

Tabela 6 - Especificações motor de passo.

Conexão	Bipolar-Série
Torque (kgf.cm)	1,10
Corrente (mA/fase)	70
Tensão (V/fase)	16,8
Ângulo do passo	1,8º
Número de passos	200
Número de fios	6

Fonte: Neoyama, AK17 Datasheet.

Figura 16 - Esquema de ligação do motor de passo e o EasyDriver com o Arduino.



Fonte: (GRAN-JANSEN, 2014).

d. Corda e instrumento: O projeto foi feito para um ukulele, o qual possui quatro cordas. As cordas escolhidas são da marca Andaluz e as especificações estão na Tabela 7. A corda utilizada para testes foi a corda 1, porém, pode ser substituída pelas outras cordas. Quanto à tarraxa do instrumento, há uma engrenagem entre a tarraxa e o pino que a corda é enrolada. Tal engrenagem provoca uma redução de 1/15 na rotação da tarraxa.

Tabela 7 - Especificações das cordas do ukulele utilizadas do projeto.

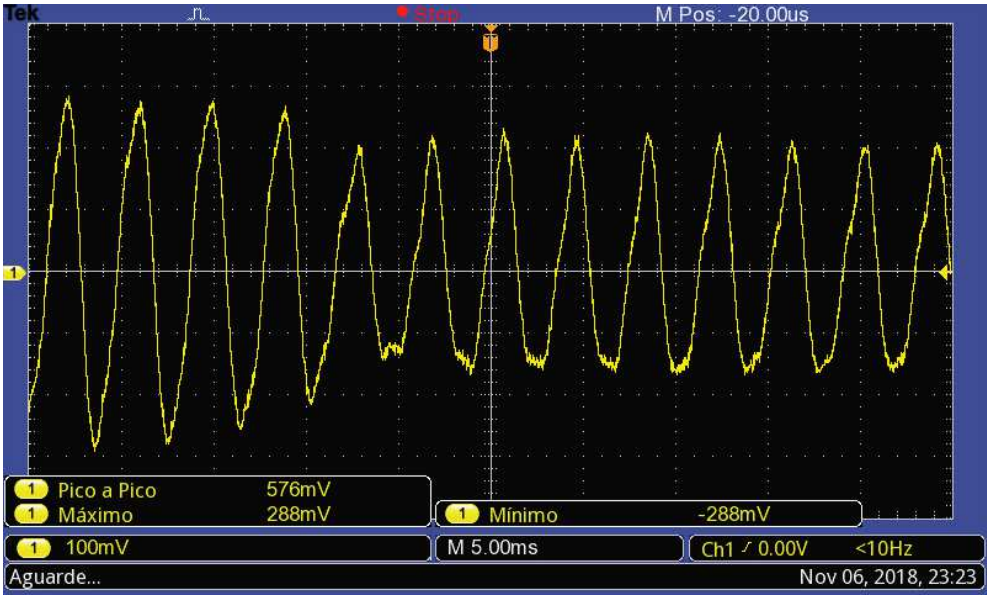
Corda	1	2	3	4
Material	10% Nylon + 20% poliéster + fibra de carbono			
Nota	A4 (440 Hz)	E4 (329 Hz)	C4 (262 Hz)	G4 (392 Hz)
Comprimento (cm)	64,9	65,1	64,9	63,9
Diâmetro (mm)	0,58	0,78	0,94	0,66
Peso (g)	0,2531	0,4206	0,6223	0,2826

Fonte: do autor.

5.1 Pré-processamento do sinal acústico

O som propagado pela corda do instrumento percorre por vários processos antes de ser lido pelo microcontrolador. Analisando primeiramente o sinal elétrico que o microfone produz, de acordo com o som captado, foi observado por um osciloscópio (Figura 17) que a tensão varia de -0,288 a 0,288 mV. De acordo com as especificações[12] do Arduino UNO, a tensão que o pino analógico pode receber é de 0 a 5 V. Deste modo, para facilitar o processamento do sinal pelo microcontrolador, o sinal elétrico do microfone precisou ser ampliado a uma faixa de -2,5 a 2,5 V, e posteriormente deslocado para cima, de forma que o sinal varie entre 0 a 5 V.

Figura 17 – Sinal de saída do microfone visto pelo osciloscópio.



Fonte: do autor.

[12] *Datasheet* do Arduino UNO.

O circuito amplificador escolhido possui característica não-inversora (Figura 18) utilizando um amplificador operacional UA741CN. Os valores de R_1 e R_2 foram estabelecidos conforme a Eq. (16) (SEDRA; SMITH, 2007):

$$\frac{v_o}{v_i} = 1 + \frac{R_1}{R_2} \tag{16}$$

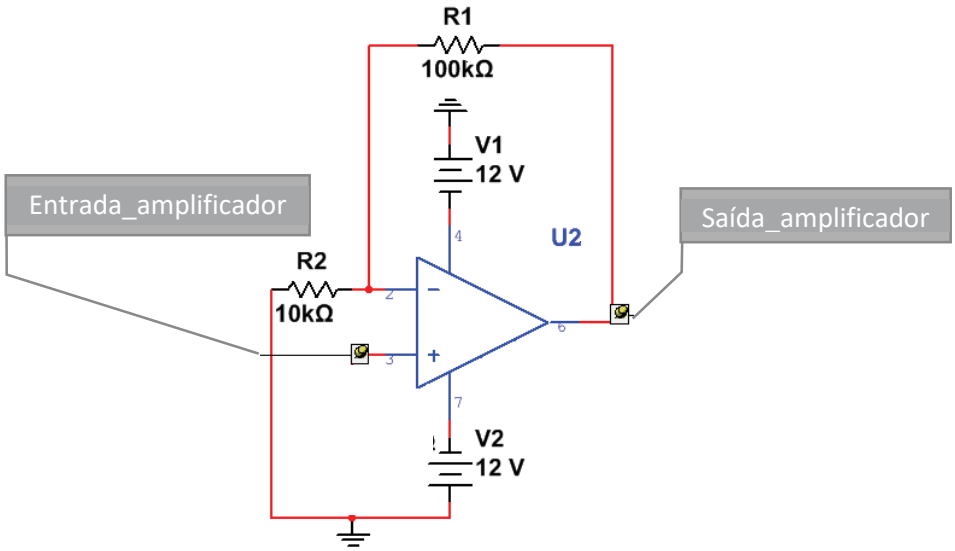
onde v_o é a tensão de saída e v_i é a tensão de entrada. O valor do sinal de entrada foi arredondado para 0,3 mV para facilitar os cálculos. Assumindo R_1 como 100 kΩ, tem-se:

$$\frac{2{,}5V}{0{,}3V} = 1 + \frac{100 \times 10^3}{R_2}, \tag{17}$$

$$R_2 = 13{,}637\ k\Omega. \tag{18}$$

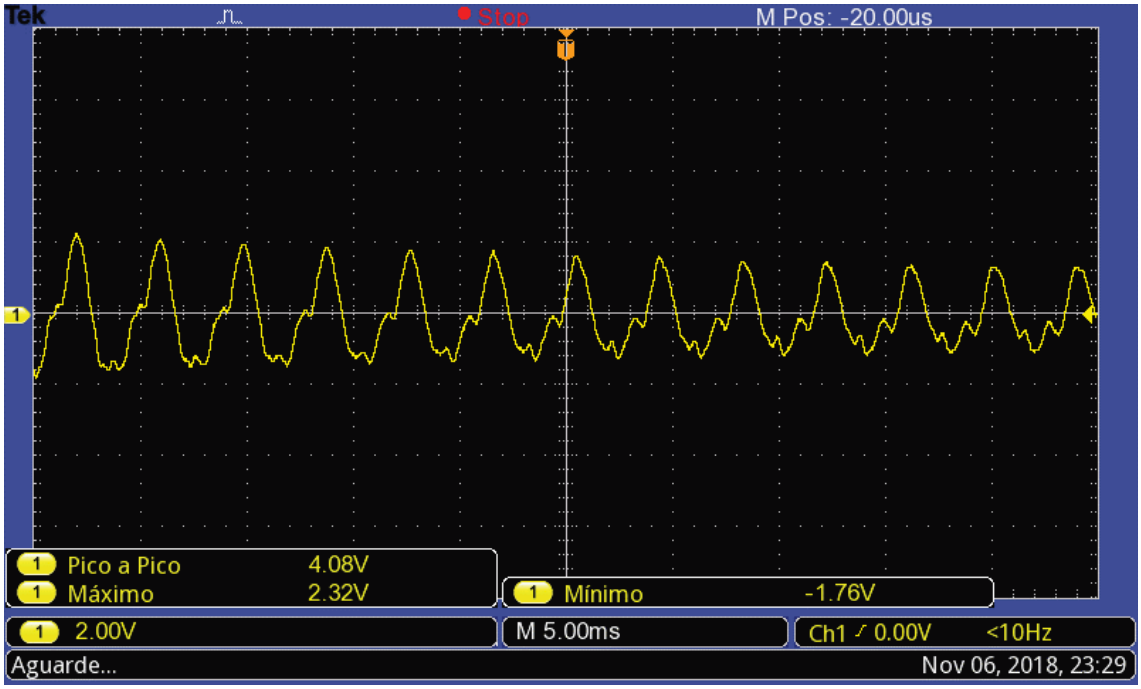
Para facilitar a montagem do circuito, R_2 foi substituído por um resistor com valor de 10 kΩ. O sinal de saída do amplificador pode ser visto na Figura 19.

Figura 18 - Amplificador não-inversor.



Fonte: do autor.

Figura 19 - Sinal do circuito amplificador no osciloscópio.



Fonte: do autor.

Para o sinal ficar entre 0 e 5 V é necessário acrescentar um nível DC de 2,5 V, conforme o circuito da Figura 20. O ganho DC de 2,5 V é obtido por uma divisão de tensão (ALEXANDER; SADIKU, 2008):

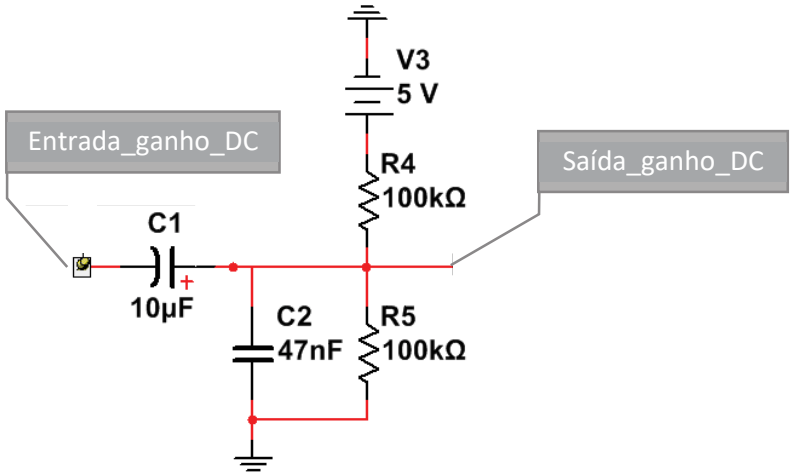
$$x = \frac{R_4}{R_4 + R_5} V_3, \tag{19}$$

$$x = 2{,}5\ V, \tag{20}$$

onde x é a tensão do nó correspondente à entrada do pino analógico do Arduino. O capacitor C_1 elimina qualquer sinal DC que pode existir na parte anterior do circuito (microfone e amplificador). O capacitor C_2 é um filtro passa-baixa. O sinal de entrada do Arduino, portanto, será o sinal do microfone amplificado entre 0 e 5 V. O sinal de saída do circuito é apresentado na Figura 21.

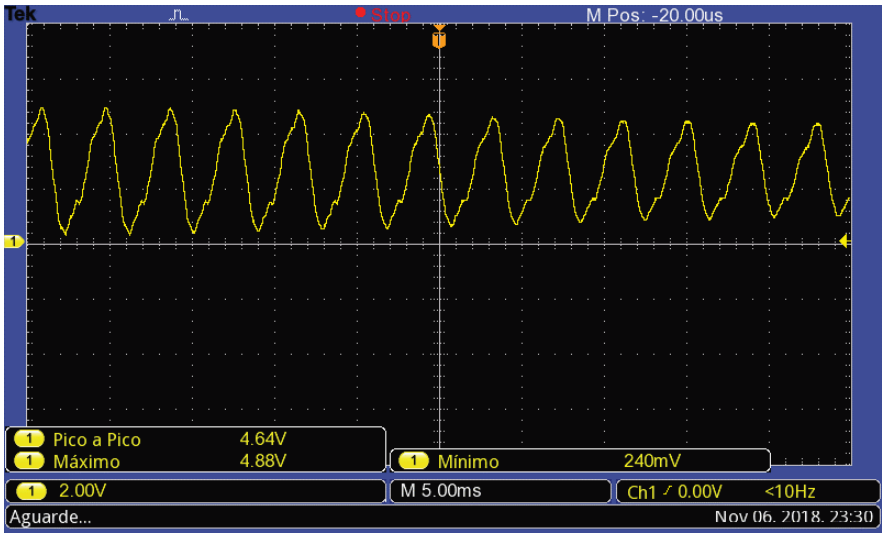
Por fim, o esquemático dos circuitos de ligação do microfone, amplificador do sinal e ganho DC é mostrado na Figura 22 e a placa real montada no projeto está na Figura 23.

Figura 20 - Circuito ganho DC.



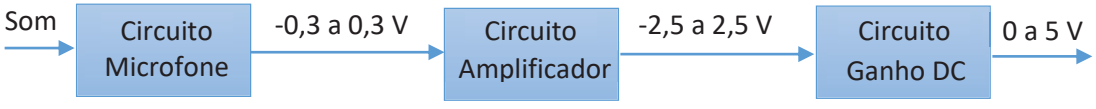
Fonte: do autor.

Figura 21 - Sinal de saída do circuito de pré-processamento do sinal de áudio.



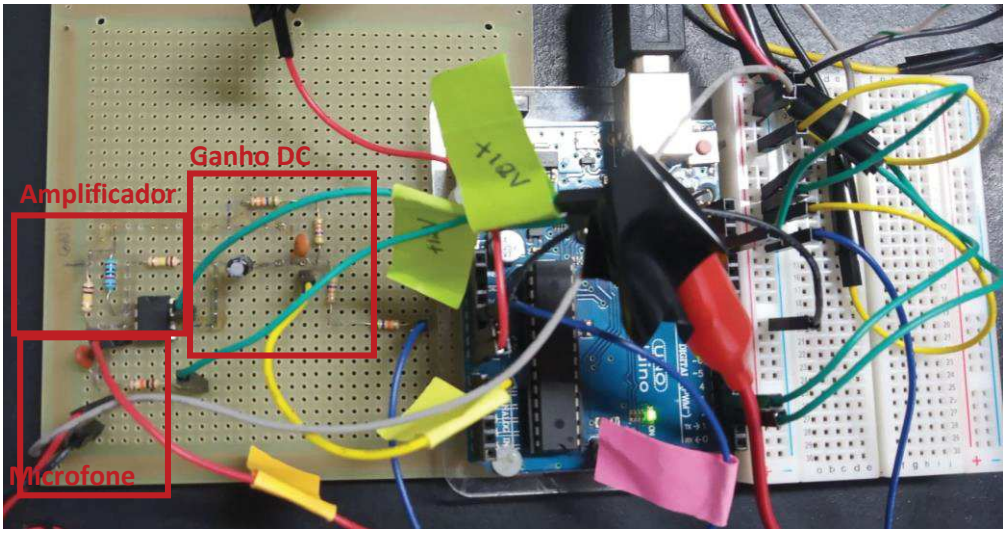
Fonte: do autor.

Figura 22 – Diagrama de blocos da etapa de pré-processamento do sinal.



Fonte: do autor.

Figura 23 - Circuito montado na placa.

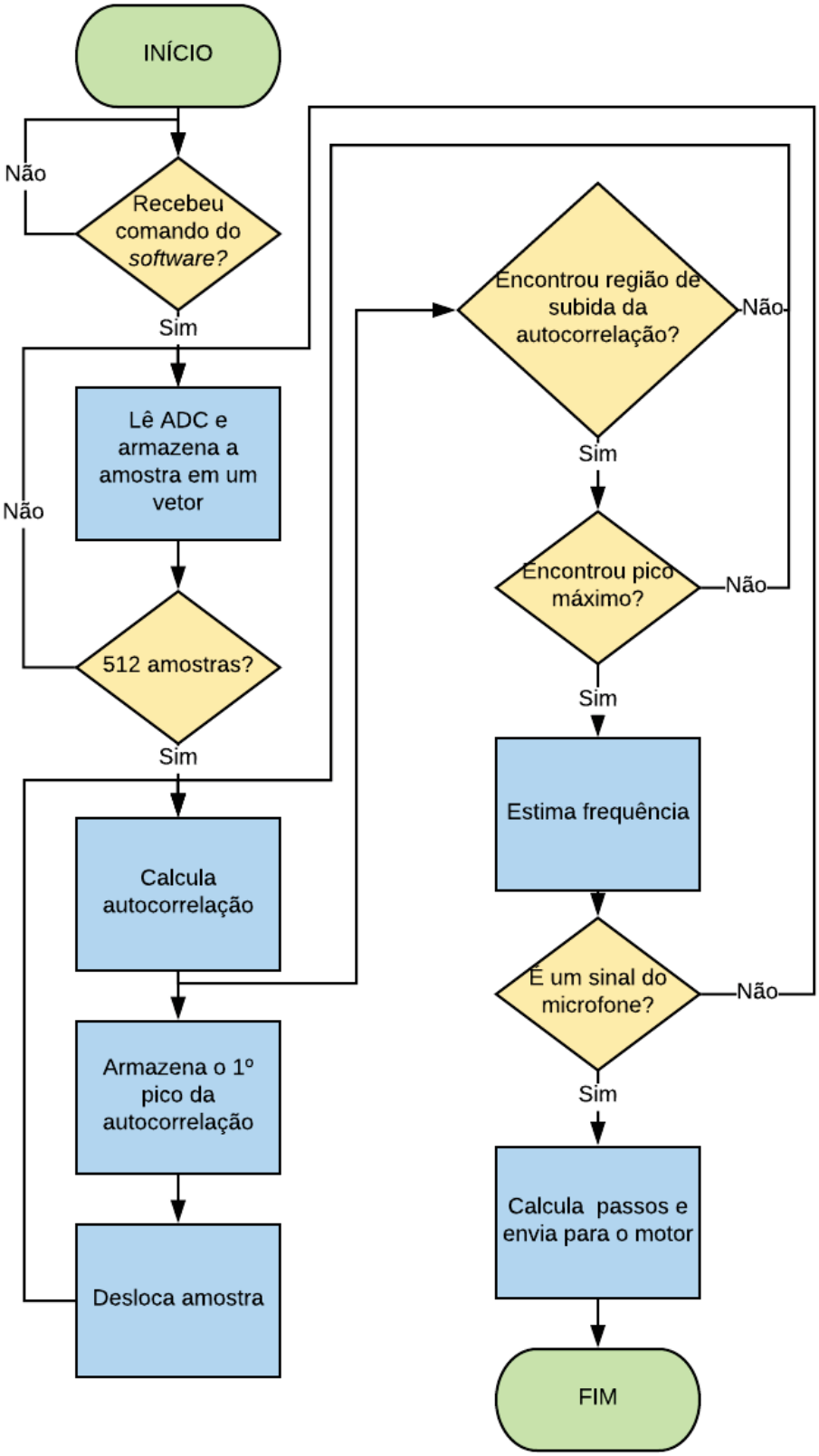


Fonte: do autor.

5.2 Algoritmo implementado no microcontrolador

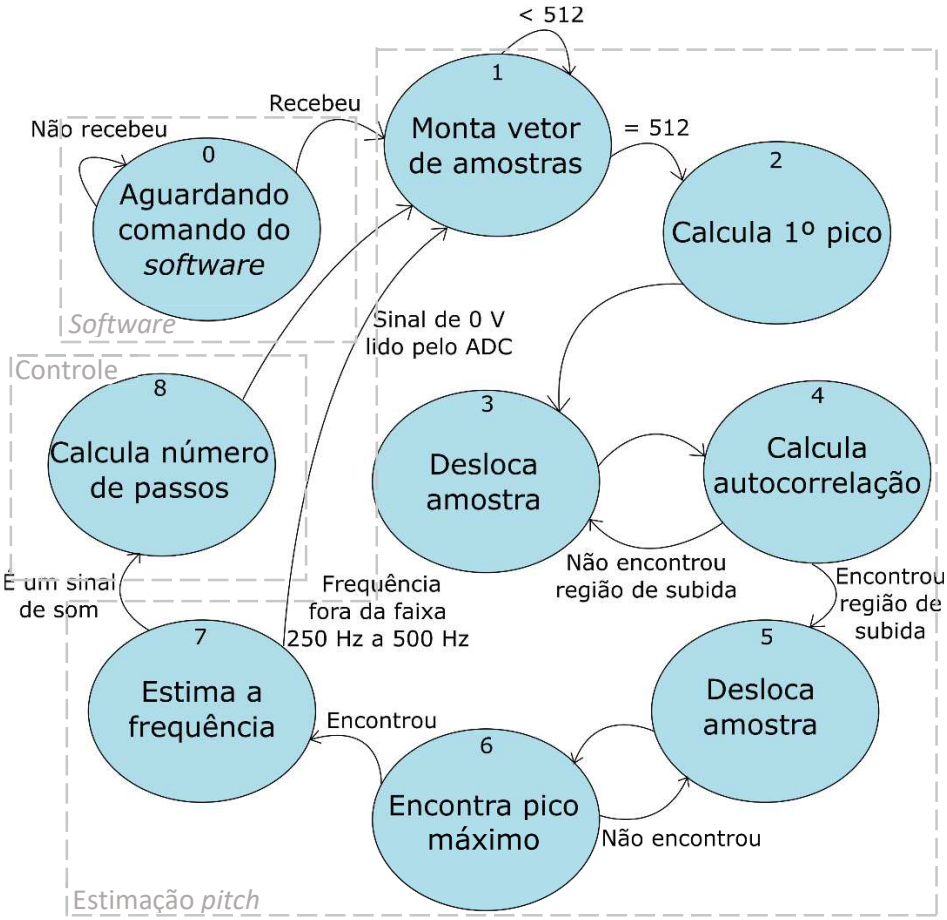
O algoritmo implementado no projeto (APÊNDICE A) é responsável por efetuar todos os cálculos e controlar o sistema. Ele pode ser dividido em três etapas: a comunicação com o *software*, a estimação do *pitch* e o controlador. Na Figura 24, o fluxograma apresenta a ordem de execução do programa, começando pelo aguardo do comando enviado pelo *software*, a leitura do canal ADC, a estimação da frequência pela autocorrelação e o controlador. A Figura 25 contém a máquina de estados elaborada para guiar a construção do código.

Figura 24 - Fluxograma do sistema.



Fonte: do autor.

Figura 25 - Máquina de estados do algoritmo implementado no microcontrolador.



Fonte: do autor.

5.2.1 Algoritmo para a estimação do *pitch*: Autocorrelação

Conforme a Figura 25, depois que o Arduino recebe o comando do *software*, via Serial, inicia-se a estimação do *pitch*. Para isso, primeiramente o sinal analógico precisa ser discretizado; esse processo é feito através de um conversor Analógico-Digital (ADC) do Arduino. O primeiro estado “MONTA_VETOR_ADC” preenche um vetor com as amostras do sinal discretizado. São armazenadas 512 amostras no vetor “EstimarPitch.sinalADC” (Quadro 1). A taxa de amostragem do ADC do Arduino é de aproximadamente 10.000 Hz; logo, são armazenados aproximadamente 50 ms do sinal.

Após o vetor ser completado, é calculado a autocorrelação com um atraso igual a zero, determinando o valor do primeiro pico da autocorrelação no estado “CALCULA_PICO_1”. Nesse estado também é estabelecido um valor mínimo para o segundo pico da autocorrelação, o qual é igual à metade do valor do primeiro pico. Esse limite é estabelecido para encontrar a região em que começará a ter um aumento nos valores da autocorrelação, próximo ao segundo pico máximo.

Quadro 1 - Função que monta o vetor com as amostras do sinal.

```
//função que lê o pino analógico e monta o vetor de amostras
void lerADC() {

    //Lê o valor e armazena no vetor                //Valor ADC de 10 bits para 8 bits
    EstimarPitch.sinalADC[EstimarPitch.contAmostras] = analogRead(PINO_ANALOGICO) >> 2;

    //incrementa a posição do vetor
    EstimarPitch.contAmostras++;
}
```

Fonte: do autor.

A próxima etapa é deslocar o sinal e calcular a autocorrelação através da função "autocorrelacao()". Esse processo ocorre até ser encontrado um aumento nos valores da autocorrelação, uma região de subida. Quando encontrado, o sinal será novamente deslocado uma amostra e é calculada novamente a autocorrelação, repetindo as operações até encontrar o pico máximo da região de subida, ou seja, até encontrar uma região de descida. Uma vez encontrado o segundo pico, o estado "ESTIMA_FREQ" verifica se o valor do segundo pico é de um som captado pelo microfone. Para estabelecer este valor, foi analisado o valor do pico quando a corda era tocada (em diferentes frequências) e o valor quando nenhuma corda era tocada (canal ADC recebendo 0 V). O valor para um sinal de 0 V é igual a 32 e o menor valor testado com as notas era de aproximadamente 120. Portanto, se o valor do segundo pico for aceitável, o instante que este pico foi encontrado é o período fundamental T_0 do sinal. Finalmente, para estimar a frequência fundamental f_0, como o sinal é discretizado, é feito o seguinte cálculo:

$$f_0 = \frac{F_s}{T_0}, \tag{21}$$

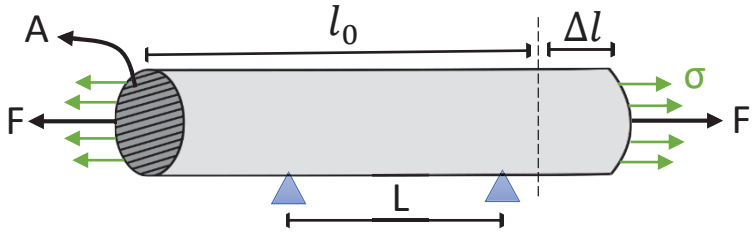
onde F_s é a taxa de amostragem do ADC. Para determinar o valor exato da taxa de amostragem, foi utilizada uma função chamada "micros()" que retorna o número de microssegundos desde que o programa começou a executar. Com isso, recebendo o valor da função quando o canal ADC começa a ler o sinal e o valor de quando termina de receber o sinal é possível calcular a frequência para a leitura de cada amostra. O valor obtido foi 9.217 Hz. A última etapa, após calcular a frequência fundamental, é uma verificação que ignora o valor calculado se este valor estiver fora de uma faixa de frequência (250 Hz a 500 Hz); isso foi necessário, pois, dependendo de como a corda é tocada, se for muito forte, é calculada uma frequência errada, muito alta ou muito baixa, atrapalhando o processo de afinação. Assim se encerra o processo de estimação do *pitch* e avança para a etapa do controlador, que será mostrado na seção 5.4.

5.3 Modelagem matemática

A identificação do sistema foi realizada através da modelagem matemática, mediante a relação entre o comprimento linear que a corda deverá esticar ou afrouxar de acordo com a frequência atual da corda e a frequência que deseja-se obter. Posteriormente, o comprimento é convertido em ângulo que a tarraxa deve girar.

Para a modelagem, analisa-se primeiro a força que atua na corda do sistema (Figura 26).

Figura 26 - Demonstração da corda do sistema.



Fonte: do autor.

Em resistência dos materiais, a tensão de tração σ [N/m²] na corda é dada pela seguinte fórmula (BOTELHO, 2013):

$$\sigma = \frac{F}{A}, \tag{22}$$

onde F é a força [N] submetida à corda e A é a área [m²] da secção transversal da corda.
Outra interpretação da tensão de tração é a lei de Hooke Eq. (23), a qual representa a equação da região linear do diagrama tensão-deformação (HIBBELER, 2010):

$$\sigma = E \cdot \epsilon, \tag{23}$$

o módulo de Young, ou módulo de elasticidade E [N/m²] representa a constante de proporcionalidade entre a tensão e a deformação ϵ. Tal deformação é obtida pela relação entre a variação do comprimento Δl [m] (o quanto a corda esticou) e o comprimento inicial l_0 [m] da corda Eq. (24) (BOTELHO, 2013):

$$\epsilon = \frac{\Delta l}{l_0}. \tag{24}$$

Substituindo a Eq. (22) e a Eq. (24) na Eq. (23), obtém-se a seguinte relação:

$$F = \frac{E \cdot A \cdot \Delta l}{l_0}. \tag{25}$$

Esta definição Eq. (25) é substituída na Eq. (1). A partir disso, tem-se:

$$f_n = \frac{n}{2L} \cdot \sqrt{\frac{E \cdot A \cdot \Delta l}{\mu \cdot l_0}}. \tag{26}$$

Através de manipulação da Eq. (26) é possível obter que:

$$f_n{}^2 = \frac{E \cdot A \cdot \Delta l \cdot n^2}{l_0 \cdot 4 \cdot L^2}. \tag{27}$$

Como o objetivo é encontrar somente a frequência fundamental, n é igualado a 1:

$$f^2 = \frac{E \cdot A \cdot \Delta l}{l_0 \cdot 4 \cdot L^2}. \tag{28}$$

Nota-se que o módulo de Young E, a área da seção transversal A da corda, o comprimento inicial l_o, o comprimento fixo L e a divisão por 4 são constantes, logo:

$$f^2 = c \cdot \Delta l, \qquad c = \frac{E \cdot A}{l_0 \cdot \mu \cdot 4 \cdot L^2}. \tag{29}$$

Fazendo uma análise onde, de uma determinada frequência f_1 (atual), queira atingir uma outra frequência f_2, tem-se:

$$\Delta f = f_2 - f_1. \tag{30}$$

Dessa variação de frequência terá uma variação do comprimento da corda:

$$\Delta l = \Delta l_2 - \Delta l_1. \tag{31}$$

Assim, substituindo as Eqs. (30) e (31) na Eq. (28), tem-se:

$$\Delta l = \frac{{f_2}^2 - {f_1}^2}{c}. \tag{32}$$

Esta é a equação final do sistema, onde é possível saber qual o valor, em metros, que a corda deve ser esticada para ser alterada da frequência f_1 para f_2.

Convertendo essa medida em metros para o ângulo em graus que a tarraxa deve girar, tem-se:

$$\Delta\theta = \frac{360 \cdot \Delta l}{2\pi \cdot (r_c + r_t)}, \tag{33}$$

onde r_c é o raio da corda e r_t é o raio do pino que a corda se enrola ligado à tarraxa.

Os valores dos parâmetros (Tabela 8) que formam a constante c da Eq. (29) e os raios da Eq. (33) foram medidos com o auxílio de balança de precisão, paquímetro e fita métrica. A massa da corda foi necessária para determinar a densidade linear μ. O único parâmetro que precisou de um experimento para ser determinado foi o módulo de elasticidade E. Foi preso à corda um peso de 1 kg e medido o quanto a corda esticou e posteriormente substituído os valores na Eq. (25). Com isso, foi obtido uma variação de 2,5 mm resultando em um módulo de elasticidade igual a 9,5186 GPa.

Tabela 8 - Parâmetros medidos da corda.

Massa, m (kg)	$(0{,}2531 \pm 0{,}00005) \times 10^{-3}$
Comprimento, l_0 (m)	$0{,}649 \pm 0{,}0005$
Comprimento, L (m)	$0{,}356 \pm 0{,}0005$
Raio, r_c (m)	$(0{,}29 \pm 0{,}005) \times 10^{-3}$
Raio, r_t (m)	$(2 \pm 0{,}005) \times 10^{-3}$

Fonte: do autor.

5.4 Algoritmo de controle

A intenção inicial do projeto era realizar um controlador PID. Para a sintonização correta do controlador, o ganho proporcional, o tempo integral e o tempo derivativo seriam obtidos pelo método de Ziegler-Nichols; para isso, foi feita a modelagem do sistema para posteriormente aplicar o método. Ao realizar a modelagem, percebeu-se que o sistema é estático, não varia no tempo, resultando em um empecilho para a o método de Ziegler-Nichols; uma vez que, uma das etapas é determinar o período de oscilação da resposta do sistema. Como o sistema é estático, não há como mensurar esse período, já que o tempo entra as oscilações é puramente o tempo que o usuário demora para tocar novamente a corda e o motor atuar. Visto isso, o controlador PID foi implementado e sintonizado através da tentativa e erro, até encontrar valores que geram a resposta desejada no sistema. Todavia, analisando a equação do sistema, notou-se que é possível implementar um controlador onde a própria equação do sistema realiza o controle, equivalente a um controlador Proporcional. Por isso, foi implementado duas opções de controlador no projeto. A seguir será explicado como cada uma delas.

5.4.1 Controlador PID

A função que é responsável pelo controlador PID (Quadro 2) possui uma verificação de tempo, onde limita a atuação do PID em um intervalo de 200 ms. Para o cálculo da saída do PID, a qual é o número de passos necessários que o motor de passo deverá girar, o valor do ganho proporcional (Kp), tempo integral (Ti) e tempo derivativo (Td) são muito importantes. A escolha desses valores foi estabelecida através de tentativa e erro até a resposta do sistema ser satisfatória, onde o sistema atinge a frequência desejada sem ocorrer sobressinais e sem oscilar. Com isso, os valores que cumpriram com o comportamento do sistema foram: Kp igual a 30, Ti igual a 0,2 e Td igual a 7.

Quadro 2 - Função do controlador PID.

```
//função que calcula os passos que o motor deve dar
void controlePID(){
    //variáveis para armazenar o erro
    static double erro = 0, erroAnterior = 0, dErro = 0;
    //variável para armazenar o instante de atuação do PID
    static unsigned long instanteAtuacao = 0;

    if(instanteAtuacao == 0)
      instanteAtuacao = millis(); //inicia a contagem de tempo;
    else{
      unsigned long t = millis(); //armazena quanto tempo passou;

      while(t < (instanteAtuacao + 200)) //espera passar 200 ms para atuar;
        t = millis();

      instanteAtuacao = t; //atualiza instante de tempo da atuação
    }

    //Cálculo do erro entre a frequência atual e a desejada
    erro = CalcularPassos.setpoint - EstimarPitch.pitch;

    //Cálculo do termo integral
    CalcularPassos.Integral += (CalcularPassos.Ki * erro);

    dErro = (erro - erroAnterior); //Cálculo da variação do erro atual e o anterior

    //Saída do PID é o número de passos que o motor dará
    CalcularPassos.passos = (CalcularPassos.Kp * erro) + CalcularPassos.Integral + (CalcularPassos.Kd * dErro);

    erroAnterior = erro;  //Salvando o valor atual do erro
}
```

Fonte: do autor.

5.4.2 Controlador matemático

A equação do sistema, Eq. (32), fornece, em metros, o quanto a corda deve esticar para alterar de uma frequência atual para uma frequência desejada. Analisando a equação, nota-se que se trata de um ganho multiplicado por um erro. Onde, o ganho é composto pelas propriedades da corda e o erro é a diferença dos quadrados das frequências. O Quadro 3 mostra a função implementada que calcula o número de passos que o motor deve dar a partir da equação do sistema.

A cada leitura da frequência da corda, o valor atual da frequência “EstimarPitch.pitch” é alterado, até o *setpoint* ser atingido. Pela teoria, em apenas uma atuação deveria atingir a frequência desejada; porém, devido a erros de aproximação no cálculo da autocorrelação, são necessárias mais atuações.

Os vetores “vCorda[]” e “vRaio[]” armazenam os valores das constantes e dos raios das cordas, respectivamente. As posições vão de 0 a 3 e armazenam os dados das cordas 1 a 4. O *software* envia qual é a corda que o usuário escolheu e o cálculo é feito com a posição correta do vetor.

A função, primeiramente, calcula o erro entre a frequência estimada atual e a frequência do *setpoint*. Esse valor é multiplicado pela constante da corda escolhida e assim é determinada a variação necessária do comprimento da corda. Tendo esse valor como base, é determinado o ângulo que a tarraxa deve girar para atingir tal variação. Finalmente, através de uma regra de três, é calculado o número de passos equivalente ao ângulo. Vale ressaltar que o valor da constante "NUMERO_PASSOS" é 24.000, pois é o número de passos que o motor deve dar para girar a engrenagem ligada à tarraxa em 360º.

Quadro 3 - Função do controle matemático.

```
void controleMatematico() { //função que calcula os passos que o motor deve dar
    static float erro = 0;

    //erro entre a frequência atual e a desejada
    erro = (CalcularPassos.setpoint*CalcularPassos.setpoint)-(EstimarPitch.pitch*EstimarPitch.pitch);

    //calculo do quanto a corda deve esticar
    CalcularPassos.deltaLCorda = CalcularPassos.vCorda[ComunicacaoSoftware.corda]*erro;

    CalcularPassos.angulo = CalcularPassos.deltaLCorda/vRaio[ComunicacaoSoftware.corda];

    //calculo do ângulo que a tarraxa deve girar
    CalcularPassos.anguloGraus = (GRAUS_VOLTA_COMPLETA*CalcularPassos.angulo)/(2*VALOR_PI);

    //quantos passos para atingir o ângulo
    CalcularPassos.passos = (CalcularPassos.anguloGraus*NUMERO_PASSOS)/GRAUS_VOLTA_COMPLETA;
}
```

Fonte: do autor.

5.5 Interface de usuário

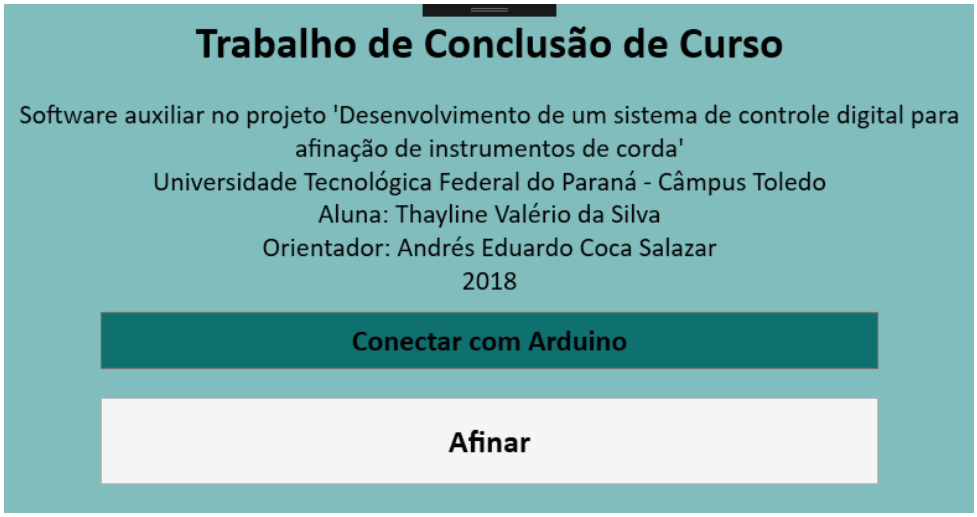
Para auxiliar o uso do dispositivo, foi criado um *software* com uma interface de uso, onde o usuário seleciona a corda do ukulele que deseja afinar e a nota de referência que deseja afinar a corda. Ao selecionar a corda, a frequência de referência é ajustada ao padrão de afinação da respectiva corda; porém, também é possível alterar a frequência para uma outra. Outra opção que o usuário seleciona antes de iniciar a afinação é o método de controle desejado: o PID ou o Proporcional a partir da equação do sistema.

Existem duas telas na interface, na tela inicial (Figura 27) a primeira ação a ser tomada é clicar no botão que faz o *software* se comunicar com o Arduino. Enquanto essa comunicação não for feita, não é possível clicar no botão "Afinar", pois estará inativo. Uma vez feita a comunicação, a tela inicial mudará o estado dos botões possibilitando abrir a tela de afinação.

Na tela de afinação (Figura 28), há botões para selecionar a corda do ukulele que o dispositivo está encaixado. Na parte superior da tela, há uma barra ajustável onde, quanto mais

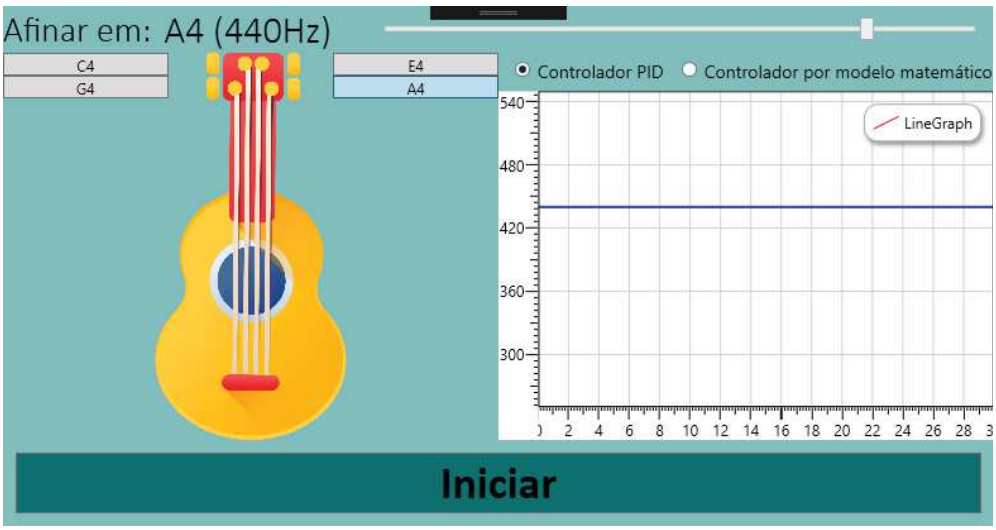
à esquerda, menor é a frequência (nota) e quanto mais à direita, maior é a frequência (nota). Embaixo da barra ajustável, estão as opções de controle do sistema. O botão “Iniciar” permite o envio da nota de referência para o Arduino e o início do processo de afinação (está é a condição para mudar o estado da máquina de estados para a etapa de estimação do *pitch*, como é mostrado na Figura 25). Na mesma tela, à direita, há um gráfico que apresenta em tempo real a frequência atual da corda, conforme o Arduino realiza a leitura. A linha azul marca a frequência da nota de referência, que facilita a visualização de quando a corda atingir a nota escolhida.

Figura 27 - Tela inicial da interface de usuário do sistema de afinação automática de instrumento de cordas.



Fonte: do autor.

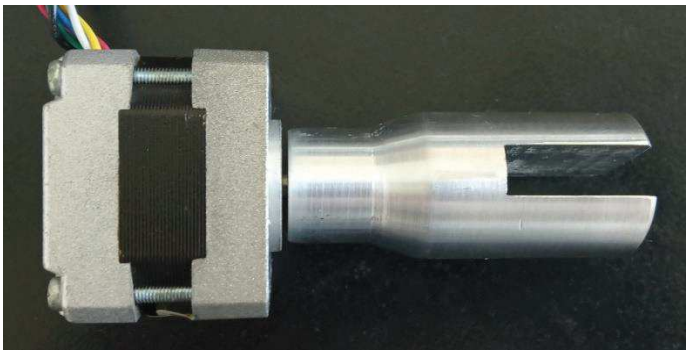
Figura 28 - Tela principal da interface de software do protótipo de afinação automática proposto.



Fonte: do autor.

A operação do dispositivo constitui de encaixar o motor na tarraxa do ukulele, prender o microfone na abertura do corpo do instrumento e tocar a corda no início e após cada atuação do motor. O motor é encaixado na tarraxa através de uma peça feita sob as medidas da tarraxa do ukulele (Figura 29). Uma visão geral do dispositivo pronto para o uso é mostrada na Figura 30.

Figura 29 - Motor com suporte para encaixe na tarraxa do instrumento.



Fonte: do autor.

Figura 30 - Motor e microfone fixos no instrumento.

Fonte: do autor.

6 EXPERIMENTOS E RESULTADOS

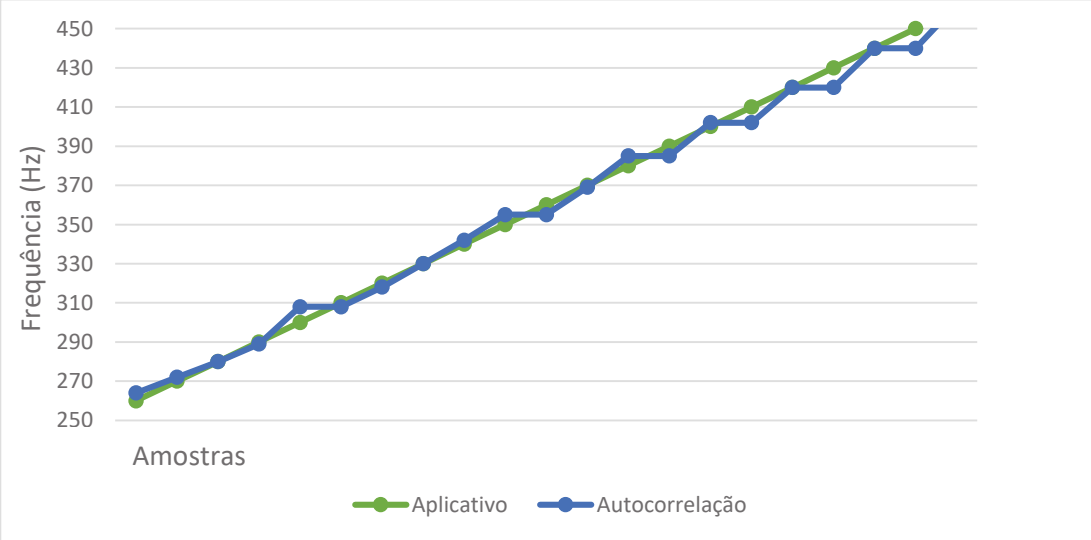
Os experimentos realizados neste projeto foram direcionadas a testar se o tempo e a precisão de afinação estão dentro do intervalo aceitável proposto no projeto. Além disso, será mostrada uma relação dos gastos que foram necessários para o desenvolvimento do projeto e os resultados quanto a usabilidade do sistema.

6.1 Precisão de afinação

Para os testes de precisão, foram comparados os valores das frequências calculadas pela autocorrelação com o valor indicado pelo aplicativo Ukulele Tuner. Foram feitos testes com frequências variando a cada 10 Hz a partir da nota C4 (261,62 Hz) até a nota A4 (440 Hz). O Gráfico 1 apresenta uma relação entre o valor da frequência mostrada pelo aplicativo (cor verde) e o valor calculado pela autocorrelação (cor azul). Nota-se que, em determinadas faixas de frequência, a autocorrelação não detecta uma diferença nos cálculos, ocorrendo uma linearização em algumas faixas de valores. As maiores diferenças ocorre em: 300 Hz, quando a autocorrelação calcula 308 Hz; 410 Hz, calculando 402 Hz; 430 Hz, obtendo-se 420 Hz; e em 450 Hz, quando é estimado 440 Hz. Dessa análise, o erro relativo máximo é obtido na frequência de 300 Hz:

$$e_r = \left(\frac{308 - 300}{300}\right) \times 100\% = \pm 2{,}67\%. \tag{34}$$

Gráfico 1 - Relação entre os valores de frequência mostrada pelo aplicativo Ukulele Tuner e calculada pela autocorrelação implementada.



Fonte: do autor.

De acordo com as medidas do Gráfico 1, o Erro Quadrático Médio (EQM) entre o valor real das frequências e os valores estimados pela autocorrelação é:

$$EQM = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} = \sqrt{\frac{471}{21}} = 4{,}731. \quad (35)$$

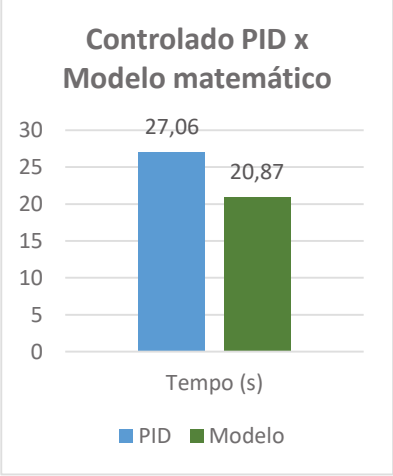
Conforme a frequência aumenta, nota-se uma maior variação na autocorrelação, isso acontece devido à frequência fundamental ser calculada a partir da Eq. (21); para períodos menores, uma pequena diferença no cálculo do período provoca uma diferença maior na frequência.

6.2 Tempo de afinação

Os experimentos para validar o tempo de afinação foram separados em: testes de comparação entre os métodos de controle (PID e modelo matemático) e testes de comparação entre a afinação automática e a afinação manual.

Para a comparação entre os métodos de controle foram feitos dez testes para cada método, divididos em cinco testes variando de 262 Hz (C4) a 440 Hz (A4) e cinco testes com as mesmas frequências, porém indo da maior frequência para a menor, afrouxando a corda. O tempo de afinação foi cronometrado para cada teste e posteriormente feita uma média entre o resultado dos dez testes de cada método. O Gráfico 2 apresenta o resultado obtido.

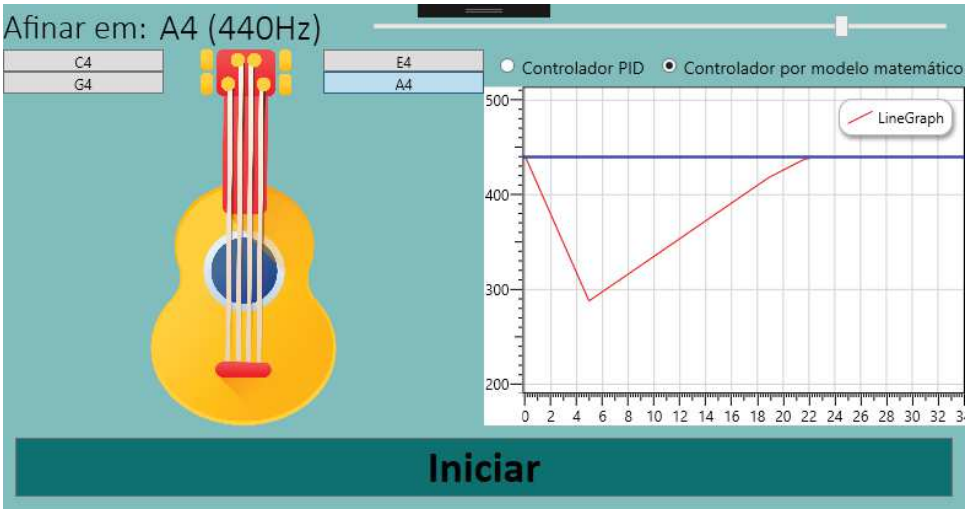
Gráfico 2 - Comparação do tempo de afinação entre o controlador PID e o controle pelo modelo matemático.



Fonte: do autor.

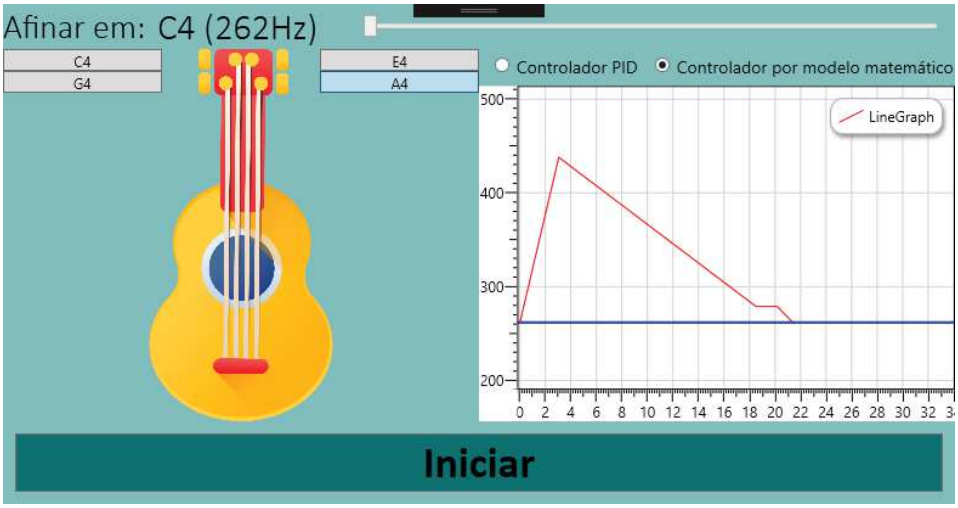
A partir dos resultados, tem-se uma economia de 22,87% no tempo utilizando o controle pelo método matemático. Isso se deve ao fato de que com o controlador PID foram necessárias mais atuações até atingir a frequência desejada, enquanto que com o modelo matemático duas atuações eram suficientes, também porque o processo de cálculo é menor. As Figuras 31 e 32 mostram a resposta do sistema para o modelo matemático e as Figuras 33 e 34 mostram a resposta do sistema para o controlador PID. Os quatro testes foram realizados na corda A4, variando da frequência 262 Hz para 440 Hz e posteriormente variando da frequência 440 Hz para 262 Hz.

Figura 31 - Resposta do sistema com o modelo matemático variando da nota C4 para A4.



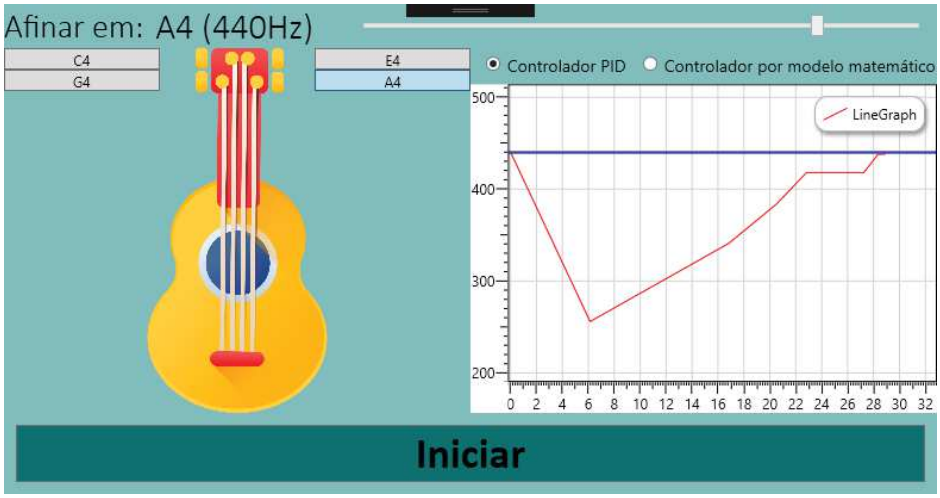
Fonte: do autor.

Figura 32- Resposta do sistema com o modelo matemático variando da nota A4 para C4.



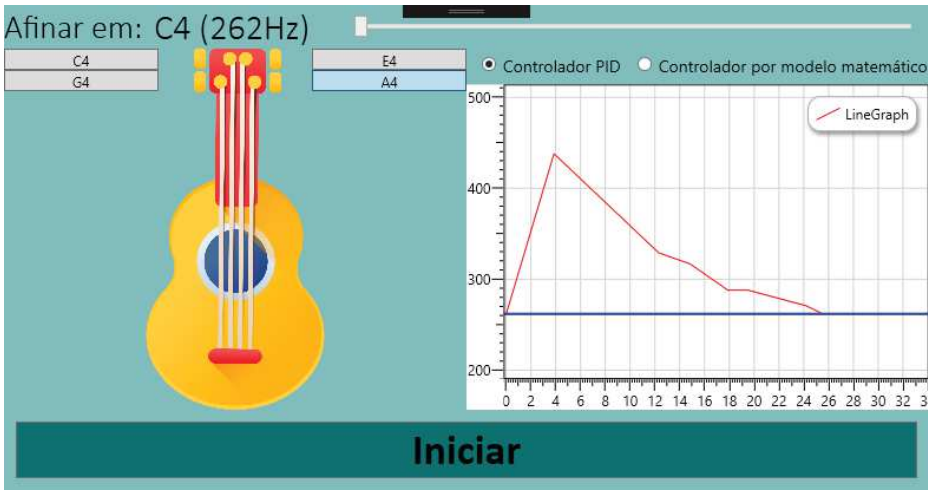
Fonte: do autor.

Figura 33- Resposta do sistema com o controlador PID variando da nota C4 para A4.



Fonte: do autor.

Figura 34- Resposta do sistema com o controlador PID variando da nota A4 para C4.

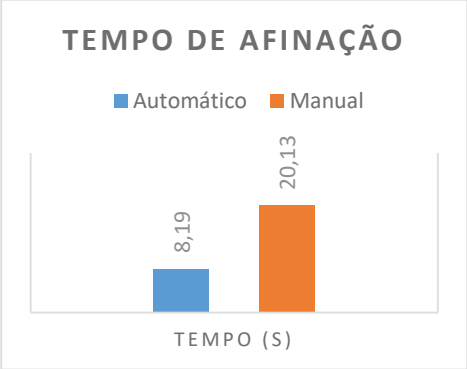


Fonte: do autor.

Os testes para comparação entre a afinação manual e automática utilizou somente o controle através do modelo matemático. Foram dez testes para cada método, sendo dividido em 5 testes variando da nota A4 para G4 e 5 testes da nota G4 para A4. O Gráfico 3 apresenta a média do tempo gasto, em segundos, para cada método.

Os resultados de economia de tempo quanto à afinação manual e à afinação automática foi de 59,32%.

Gráfico 3 - Comparação do tempo gasto entre afinação automática e a afinação manual.



Fonte: do autor.

6.3 Relação de gastos para o desenvolvimento do projeto

Para a construção do projeto, foi proposto o menor custo possível. A Tabela 9 contém uma relação de todos os gastos para o desenvolvimento do projeto; com base nela tem-se que no total foram gastos R$269,00. Quanto aos afinadores automáticos disponíveis no mercado, o preço do RoadieTuner varia de U$99 a U$149, segundo o próprio site da empresa, e o TronicalTune varia entre U$200 a U$434, de acordo com o site de compras Amazon; convertendo esses valores para Real, segundo o site de informações econômicas da UOL, a cotação do dólar é de R$3,75 (17/11/2018 às 20:30), isso totaliza um valor de R$371,25 a R$558,75 para o RodieTuner e de R$750,00 a R$1627,50 para o Tronical Tune. Dessa forma, a economia de custo entre o protótipo construído e os afinadores existentes é de 27,54% a 83,47%, respectivamente.

Tabela 9 - Gastos obtidos para o desenvolvimento do projeto.

Elemento	Preço
Componentes eletrônicos	R$22,00
Placa para soldar	R$10,00
Corda	R$20,00
Motor de passo	R$85,00
Microfone	R$2,00
EasyDrive	R$20,00
Arduino Uno	R$30,00
Suporte motor	R$80,00
TOTAL	R$269,00

Fonte: do autor.

6.4 Usabilidade do sistema

A interface e o processo de afinação, realizam juntos uma usabilidade intuitiva e prática do sistema. Há apenas uma proposta de melhoria quanto a posição que o instrumento deve permanecer durante o processo de afinação, deitado em uma mesa; uma forma de melhorar é realizar a afinação segurando o instrumento. Para isso, é necessário a construção de um suporte para fixar o motor ao braço do instrumento e, ainda assim, ficar alinhado ao eixo da tarraxa.

Outro objetivo do projeto era a adaptação do sistema para outras configurações desejadas pelo usuário, como trocar a corda a ser afinada e a nota de referência; isso foi cumprido: para alterar a corda, basta selecionar no *software* a corda desejada e encaixar o motor na respectiva tarraxa, e para alterar a nota de referência é somente mudar a nota desejada através da barra ajustável na tela de afinação da interface.

7 CONCLUSÕES

Ao final do projeto, obteve-se um afinador que pôde automaticamente afinar a corda do instrumento ukulele a uma frequência desejada. O processo de afinação teve um bom desempenho no tempo, afinando em menos de 30 segundos notas com 9 semitons de distância. Obteve-se, também, uma afinação com precisão aceitável, isto é, com baixa porcentagem de erro, ±2,67%, obtendo os valores abaixo do limiar de discriminação de frequência do ouvido humano. Quanto à facilidade de manuseio do dispositivo os resultados foram positivos, tendo objeções apenas quanto a posição do instrumento durante a afinação; isso é possível alterar fazendo um suporte de fixação do motor no braço do instrumento.

O desenvolvimento do *software* de interface contribuíram para um uso simples e intuitivo do sistema. Os testes foram realizados com apenas uma corda, porém, a adaptação para o restante das cordas do ukulele foi de fácil implementação.

O custo total do projeto, comparado com os preços dos afinadores atualmente no mercado, foi baixo: R$269,00. Tal resultado permite que este produto possa ter um maior custo-benefício.

Uma forma de melhorar ainda mais os resultados obtidos neste projeto é realizar testes com pessoas de diferentes níveis musicais e comparar o tempo de afinação automática e manual e a usabilidade do sistema. Outra sugestão é acrescentar diferentes métodos de estimação do *pitch*, podendo obter valores mais precisos e possibilitar a comparação entre os resultados dos métodos.

8 REFERÊNCIAS

ALEXANDER, C.; SADIKU, M., **Fundamentos de Circuitos Elétricos**. 3. ed. São Paulo: Mcgraw-hill, p. 30-51, 2008.

ARDUINO, **ARDUINO UNO REV 3**, [s.d.]. Disponível em: <https://store.arduino.cc/usa/arduino-uno-rev3>. Acesso em: 15 jan. 2018.

BASSANEZI, R., **Ensino - aprendizagem com Modelagem matemática**. 3. Ed. São Paulo: Editora Contexto, p. 15-32, 2002.

BENNETT, R., **Instrumentos da Orquestra**. 2. ed. Rio de Janeiro: Jorge Zahar Editor, Cap. 2, p. 14-25, 1986.

BERNI, A., **Temperatura X Afinação de Instrumentos Musicais**. SantoAngelo. 2016. Disponível em: <http://blog.santoangelo.com.br/temperatura-x-afinacao-de-instrumentos-musicais/>. Acesso em: 20 jan. 2018.

BOTELHO, M., **Resistência dos Materiais:** Para entender e gostar. 2. ed. São Paulo: Blucher, Cap. 5, p. 31-40, 2013.

CARNEIRO, F., **PERCEPÇÃO DO PITCH FANTASMA UTILIZANDO A SIRENE DE SEEBECK**. 12 f. Curso de Física, Instituto de Física Gleb Wataghin, Universidade Estadual de Campinas, Campinas, 2003.

CHAMBERLIN, H., **Musical Applications of Microprocessors**. 2. ed. Indianapolis: Hayden Books, p. 578-588, 1987.

CHEN, C., **Analog and Digital Control System Design**: Transfer-Function, State-Space, and Algebraic Methods. Nova Iorque: Oxford University Press, p. 551-566, 1993.

COCA, A., **ESTIMACIÓN DEL PITCH EN SEÑALES MONOFÓNICAS DE VOZ CANTADA**. 2004. 76 f. TCC (Graduação) - Curso de Engenharia Eletrônica, Departamento de Eletricidade, Eletrônica e Computação, Universidade Nacional da Colômbia, Manizales, 2004. Cap. 2.

COELHO, A.; COELHO, L. **IDENTIFICAÇÃO DE SISTEMAS DINÂMICOS LINEARES**. Florianópolis: UFSC, p. 65-81, 2004.

COTTA, A., Notas preliminares sobre o cantochão acompanhado na prática musical luso-brasileira dos séculos XVIII e XIX: o Hino a São João Batista de José Maurício Nunes Garcia. **Per Musi: Revista acadêmica de música**, Belo Horizonte, v. 1, n. 18, p.28-34, maio 2008.

FERRARO, N.; SOARES, P., **Aulas de Física 2:** Termologia - Óptica - Ondas. 7. ed. São Paulo: Saraiva S.a. Livreiros Editores, p. 392-412, 2003.

GRAN-JANSEN, S., **Arduino Tutorial: Stepper Motor with EasyDriver**. 2014. Disponível em: <https://www.norwegiancreations.com/2014/12/arduino-tutorial-stepper-motor-with-easydriver/>. Acesso em: 20 fev. 2018.

HALLIDAY, D.; RESNICK, R.; WALKER, J., **Fundamentos de Física:** Gravitação, Ondas e Termodinâmica. 9. ed. Rio de Janeiro: Ltc, p. 151- 164, 2012.

HENDRICK, D., **String instrument tuning apparatus**. US4088052 A, 9 maio 1978.

HIBBELER, R., **Resistência dos materiais**. 7. ed. São Paulo: Pearson Prentice Hall, p. 14 - 64, 2010.

KING, J.; TRINQUADA, J., A New History of the Origins and Development of the Ukulele, 1838-1915. **Hawaiian Journal Of History**, Honolulu, v. 37, 2003.

MARÇOLLA, B., **Saga de beija-flor**. Rio de Janeiro: Kbr, 2015.

MONTEIRO, L., **Sistemas dinâmicos**. 2. ed. São Paulo: Editora Livraria da Física, p. 260-263, 2006.

MED, B., **TEORIA da MÚSICA**. 4. ed. Brasília: Musimed, p. 13-63, 1996.

NETTO, L., **O Intervalo Musical "Coma" na Escala Temperada**. 2018. Disponível em: <https://musicaeadoracao.com.br/25365/o-intervalo-musical-coma-na-escala-temperada/>. Acesso em: 10 nov. 2018.

OGATA, K., **Modern Control Engineering**. 4. ed. Upper Saddle River: Prentice Hall, p. 1-103, p. 681-691, 2002.

PEREIRA, C., **Microfone de Eletreto**. [s.d.]. Disponível em: <http://blog.novaeletronica.com.br/microfone-de-eletreto/>. Acesso em: 12 abr. 2018.

RABINER, L., **On the Use of Autocorrelation Analysis for Pitch Detection**, IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASSP-25, fevereiro 1977.

SCHOLZ, D., **Automatic tuning device**. US4426907 A, 24 janeiro 1984.

SEDRA, A.; SMITH, K., **Microeletrônica**. 5. ed. São Paulo: Pearson Pratice Hall, p. 39-50, 2007.

SERAFINA, G., **Afinador automático digital para guitarras baseado no método do espectro do produto harmônico**. 2015. 57 f. Monografia (Especialização) - Curso de Engenharia de Computação, Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2015. Cap. 2.

SERWAY, R.; JEWETT, J., **Princípios de Física: Movimento Ondulatório e Termodinâmica**. 3. ed. São Paulo: Cengage Learning, p. 458-488, 2004.

SOUZA, C., **A Física da Música**. 2009. Disponível em: <http://www.das.inpe.br/~alex/FisicadaMusica/fismus_escalas.htm>. Acesso em: 02 set. 2017.

SZWOCH, G., **Pitch Detection Methods**. 2000. Disponível em: <http://sound.eti.pg.gda.pl/student/eim/synteza/leszczyna/index_ang.htm>. Acesso em: 10 set. 2017.

TEWARI, A., **Modern Control Design**: With MATLAB and SIMULINK. Wiley: John Wiley & Sons, p. 1-51, p. 373-378, 2002.

TRIMBLE, M.; HESDORFFER, D., **Music and the brain**: the neuroscience of music and musical appreciation. Bjpsych. International, [s.l.], v. 14, n. 02, Royal College of Psychiatrists, p.28-31, maio 2017.

WOLFE, J., **Note names, MIDI numbers and frequencies**. 1997. Disponível em: <http://newt.phys.unsw.edu.au/jw/notes.html>. Acesso em: 10 out. 2017.

APÊNDICE A – Algoritmo implementado no Arduino UNO

```
#define TAM_AMOSTRA 512         //número de amostras do sinal que serão
analisadas
#define FREQ_AMOSTRAGEM 9217    //frequência de amostragem [Hz] ADC
#define VALOR_MINIMO_NOTA 120   //valor estabelecido para eliminar o envio de
sinais que não forem provenientes da corda
#define LEN_BUFFER_SERIAL 64    //tamanho do buffer para a comunicação serial
entre o software e o arduino
#define PINO_ANALOGICO A0       //pino analógico utilizado para fazer a
conversão analógico-digital
#define SM_DIRECTION_PIN 2      //pino de direção de rotação do motor de passo
#define SM_STEP_PIN 3           //pino de envio dos pulsos para os passos serem
dados
#define NUMERO_PASSOS 24000     //número de passos para o motor dar uma volta
completa na engrenagem
#define VALOR_PI 3.141592       //valor do pi
#define GRAUS_VOLTA_COMPLETA 360 //graus para uma volta completa

//Enumeração dos estados da máquina de estados
typedef enum
{
 AGUARDA_SOFTWARE,
 MONTA_VETOR_ADC,
 CALCULA_PICO1,
 DESLOCA_AMOSTRA_1,
 CALCULA_AUTOC,
 DESLOCA_AMOSTRA_2,
 ENCONTRA_PICO_MAXIMO,
 ESTIMA_FREQ,
 CALCULA_PASSOS,
}ESTADOS;

//Struct para as variáveis da parte de estimar o pitch
typedef struct{
  float pitch;               //variável que armazena a frequência fundamental
estimada
  byte sinalADC[TAM_AMOSTRA]; //vetor que armazena o sinal discretizado
proveniente do microfone
  long autoc;                //variável que armazena a autocorrelação calculada
  long autocAnterior = 0;    //variável que armazena a autocorrelação calculada
anterior
  long primeiroPico;         //variável que armazena o valor da autocorrelação
quando o atraso = 0
  long valorMinimoPico2;
  int periodo = 0;
  int contAmostras = 0;      //contador do número de amostradas do sinal
discretizado
  int atrasoAmostra;         //variável que armazena o atraso do sinal na
autocorrelação
  int amostra;               //variável que controla a amostra do sinal que está
```

```
sendo calculada autocorrelação
}ESTRUTURA_ESTIMAR_PITCH;

//Struct para as variáveis da parte do controle
typedef struct{
  double setpoint;           //valor enviado pelo software com a frequência que
quer atingir
  float deltaLCorda;         //variável que armazena o valor, em metros, do
quanto a corda deve esticar ou afrouxar
  float angulo;               //
  float anguloGraus;         //variável que armazena o valor em graus que a
tarraxa deve girar para atingir a frequência
  int passos = 0;            //variável que armazena o número de passos que o
motor deve dar para atingir a frequência
  //vetor com o valor das contantes para cada corda
 double vCorda[4] = {0.0000000510191329, 0.0000000468788531, 0.0000000477574366,
0.0000000439927591};
  //vetor com o valor do raio da corda + raio do pino para cada corda
  float vRaio[4] = {0.00229, 0.00239, 0.00247, 0.00233};
  double Kp = 30;            //ganhos do controlador PID
  double Ki = 0.2;
  double Kd = 7;
  double Integral;
  float pitchAnterior;
}ESTRUTURA_CONTROLE;

//Struct para as variáveis da parte de comunicação com o software
typedef struct{
  char bufferSerial[LEN_BUFFER_SERIAL]; //vetor para armazenar o buffer que está
sendo enviado e recebido do serial
  boolean comandoRecebido = false;     //flag para indicar que um romando foi
recebido ou não
  boolean iniciarAfinacao = false;     //flag para indicar que o software foi
mandado iniciar a afinação (usuário)
  int corda = 0;
  int controle = 0;
}ESTRUTURA_COMUNICACAO;

//criação de variáveis globais
ESTADOS EstadoME;
ESTRUTURA_ESTIMAR_PITCH EstimarPitch;
ESTRUTURA_CONTROLE CalcularPassos;
ESTRUTURA_COMUNICACAO ComunicaSoftware;

void comunicacao(void);
void lerADC(void);
void autocorrelacao(void);
void controlePID(void);
void controleMatematico(void);
```

```
void atuaMotor(void);

void setup() {
  Serial.begin(115200);                    //declarando a taxa de amostragem de
dados na transmissão serial (bits/segundo)
  analogReference(DEFAULT);                //tensão de referência para a conversão
analógico-digital (5 [V])
  analogRead(PINO_ANALOGICO);              //declarando qual pino analógico será
feita a conversão analógico-digital
  pinMode(SM_DIRECTION_PIN, OUTPUT);      //declarando os pinos do motor de passo
como saída
  pinMode(SM_STEP_PIN, OUTPUT);            //"
  EstadoME = AGUARDA_SOFTWARE;             //declarando o estado inicial da
máquina de estados
  EstimarPitch.contAmostras = 0;           //iniciando o contador como 0
}

void loop() {
  //chama a função responsável pela comunicação com o software
  comunicacao();

  //Núcleo da máquina de estados
  switch(EstadoME){
    //estado que aguarda o software mandar começar a afinação
    case AGUARDA_SOFTWARE:
    {
     if(ComunicaSoftware.iniciarAfinacao)  //recebeu o comando?
        EstadoME = MONTA_VETOR_ADC;         //muda o estado da máquina

    }break;
    //estado que lê o pino analógico e armazena os valores em um vetor
    case MONTA_VETOR_ADC:
    {
       //chama a função que faz a leitura do pino analógico
       lerADC();

       //completou o vetor?
      if(EstimarPitch.contAmostras == TAM_AMOSTRA)
       {
        EstimarPitch.contAmostras = 0;    //zera o contador
        EstadoME = CALCULA_PICO1;         //muda o estado da máquina
       }

    }break;
    //estado que calcula o valor da autocorrelação para o atraso = 0
    case CALCULA_PICO1:
    {
     EstimarPitch.atrasoAmostra = 0; //atraso = 0
     EstimarPitch.periodo = 0;
```

```
     //chama a função que calcula a autocorrelação
     autocorrelacao();

     //estabele um valor minimo para o segundo pico (verifica quando a
autocorrelação começa a subir)
     EstimarPitch.valorMinimoPico2 = EstimarPitch.autoc*0.5;

     //Se é uma nota, avança o estado da máquina de estados
     EstadoME = DESLOCA_AMOSTRA_1;

    }break;
    //estado que desloca o vetor para a autocorrelação
    case DESLOCA_AMOSTRA_1:
    {
      //incrementa o atraso
     EstimarPitch.atrasoAmostra++;

     //foi deslocado todo o sinal?
     if(EstimarPitch.atrasoAmostra < 512){
       //se não, muda o estado que calcula a autocorrelação com o novo atraso
       EstadoME = CALCULA_AUTOC;
      }else{
       //se foi deslocado todo o sinal, é porque não encontrou o segundo pico.
Volta para o começo.
       EstadoME = MONTA_VETOR_ADC;
      }

    }break;
    //estado que calcula a autocorrelação do sinal
    case CALCULA_AUTOC:
    {
     //armazena o valor da autocorrelação calculada anteriormente (com um atraso
menor)
     EstimarPitch.autocAnterior = EstimarPitch.autoc;
     EstimarPitch.periodo = 0;
     EstimarPitch.autoc = 0;

     //chama a função que calcula a autocorrelação
     autocorrelacao();

     //Verifica se é o valor calculado é maior que o valor minimo do pico, e se
o valor atual menos o valor anterior é maior que 0 (subida)
     if((EstimarPitch.autoc >= EstimarPitch.valorMinimoPico2)&&((EstimarPitch.
autoc - EstimarPitch.autocAnterior) > 0)){
       //muda o estado para o que vai deslocar novamente o sinal mas agora para
determinar o pico máximo
       EstadoME = DESLOCA_AMOSTRA_2;
      }
```

```
     //se não é um pico, volta o estado para deslocar o sinal e calcular
novamente.
      else
        EstadoME = DESLOCA_AMOSTRA_1;

    }break;
    //estado que desloca a amostra após encontrar uma região que a autocorrelação
começa a aumentar (picos)
    case DESLOCA_AMOSTRA_2:
    {
     EstimarPitch.atrasoAmostra++; //atrasa mais uma amostra

      //não acabaou o sinal ainda?
     if(EstimarPitch.atrasoAmostra <= TAM_AMOSTRA)
        EstadoME = ENCONTRA_PICO_MAXIMO; //muda o estado para o que verifica o
pico máximo
      else
        EstadoME = MONTA_VETOR_ADC; //volta para o início e monta um novo vetor

    }break;
    //estado responsável por determinar o instante do segundo pico máximo
    case ENCONTRA_PICO_MAXIMO:
    {
      //salva valor da autocorrelação
     EstimarPitch.autocAnterior = EstimarPitch.autoc;
      EstimarPitch.autoc = 0;
      autocorrelacao();

      //valor calculado menos o anterior é menor que zero? (Encontrou o pico
máximo, porque agora começará a diminuir os valores)
     if((EstimarPitch.autoc - EstimarPitch.autocAnterior) <= 0)
      {
       EstimarPitch.periodo = EstimarPitch.atrasoAmostra; //salva o instante do
pico máximo
        EstadoME = ESTIMA_FREQ; //muda o estado para o que calculará a frequência
      }else
        EstadoME = DESLOCA_AMOSTRA_2; //Não encontrou o pico máximo, desloca
novamente uma amostra

    }break;
    //estado que estima a frequência do sinal
    case ESTIMA_FREQ:
    {
      //o valor minimo para o pico 2 é de uma nota musical?
     if((EstimarPitch.valorMinimoPico2 > VALOR_MINIMO_NOTA))
      {
       //inverso do período, multiplicado pela frequência de amostragem do ADC
       EstimarPitch.pitch = FREQ_AMOSTRAGEM/EstimarPitch.periodo;
```

```
      //Filtro de erros no calculo da autocrrelação. Se o pitch calculado for
menor ou maior
      //que a frequência das notas limites (C4 e B4), não atua o motor e volta
para o início.
      if(EstimarPitch.pitch >= 250 && EstimarPitch.pitch <= 500)
       {
        Serial.println(EstimarPitch.pitch);

        //muda o estado para calcular os passos do motor de passos
        EstadoME = CALCULA_PASSOS;
       }else
        EstadoME = MONTA_VETOR_ADC;
     }else
      EstadoME = MONTA_VETOR_ADC;

   }break;
   //estado que calcula o número de passos necessários e manda para o motor
   case CALCULA_PASSOS:
   {
     //Qual opção o usuário escolheu?
     if(ComunicaSoftware.controle == 0)
       controlePID();        //controle PID
     else
      controleMatematico(); //controle pelo modelo matemático

     atuaMotor();           //função que envia os passos para o motor

     //muda o estado para a montagem de um novo vetor de amostras
     EstadoME = MONTA_VETOR_ADC;

   }break;
   default:    //estado default
   {
     EstadoME = MONTA_VETOR_ADC;
   }break;
  }
}

//função para comunicar com o software
void comunicacao(){

  lerComando(); // Tentando ler qualquer comando enviado.

  // Verificando se terminour de ler um comando.
  if(ComunicaSoftware.comandoRecebido == true)
  {
    // Comando para identificar se é o arduino.
   if(strcmp(ComunicaSoftware.bufferSerial, "ARDUINO") == 0){
     Serial.println("EUMESMO");
```

```
   // Define a nota de referência para a afinação.
   } else if(strncmp(ComunicaSoftware.bufferSerial, "DEFNOTA", 7) == 0){
     ComunicaSoftware.iniciarAfinacao = true;
     CalcularPassos.setpoint = atoi(&ComunicaSoftware.bufferSerial[7]);
     Serial.println(CalcularPassos.setpoint);
    }
    // Define a corda para a afinação.
   else if(strncmp(ComunicaSoftware.bufferSerial, "DEFCORDA", 8) == 0)
    {
     ComunicaSoftware.corda = atoi(&ComunicaSoftware.bufferSerial[9]);
    }
    // Define o método de controle que será aplicado.
   else if(strncmp(ComunicaSoftware.bufferSerial, "DEFCONTROLE", 11) == 0)
    {
     ComunicaSoftware.controle = atoi(&ComunicaSoftware.bufferSerial[12]);
    }

   ComunicaSoftware.comandoRecebido = false;
  }
}

//função que lê o comando recebido pela serial
void lerComando() {
  // Não lê mais dados enquando o último comando não foi tratado.
  if(ComunicaSoftware.comandoRecebido == true)
    return;

  static byte numBytesRecebidos = 0;
  char rc;

  while (Serial.available() > 0 && ComunicaSoftware.comandoRecebido == false) {
    rc = Serial.read();

    if (rc != '\r' && rc != '\n') {
     ComunicaSoftware.bufferSerial[numBytesRecebidos] = rc;
      numBytesRecebidos++;
      if (numBytesRecebidos >= LEN_BUFFER_SERIAL) {
          numBytesRecebidos = LEN_BUFFER_SERIAL - 1;
       }
    } else if(numBytesRecebidos > 0) { // Ignorando comando vazios.
      ComunicaSoftware.bufferSerial[numBytesRecebidos] = '\0'; // fim da string
      numBytesRecebidos = 0;
      ComunicaSoftware.comandoRecebido = true;
    }
  }
}

//função que lê o pino analógico e monta o vetor de amostras
```

```
void lerADC() {
    //Lê o valor e armazena no vetor                //Valor ADC de 10 bits para
8 bits
   EstimarPitch.sinalADC[EstimarPitch.contAmostras] = analogRead(PINO_ANALOGICO)
>> 2;
    //incrementa a posição do vetor
   EstimarPitch.contAmostras++;
}

//função que calcula a autocorrelação
void autocorrelacao(){

  //loop que percorre todas as amostras do sinal
  for(EstimarPitch.amostra = 0; EstimarPitch.amostra < (TAM_AMOSTRA -
EstimarPitch.atrasoAmostra); EstimarPitch.amostra++){

    //eliminar ganho DC do sinal e obter valores negativos e positivos (0 [V] ->
128)
   EstimarPitch.autoc += ((EstimarPitch.sinalADC[EstimarPitch.amostra] -
128)*(EstimarPitch.sinalADC[EstimarPitch.amostra+EstimarPitch.atrasoAmostra] -
128));
  }

 EstimarPitch.autoc /= (TAM_AMOSTRA - EstimarPitch.atrasoAmostra);
}

//função que calcula os passos que o motor deve dar
void controlePID(){
    //variáveis para armazenar o erro
    static double erro = 0, erroAnterior = 0, dErro = 0;
    //variável para armazenar o instante de atuação do PID
    static unsigned long instanteAtuacao = 0;

    if(instanteAtuacao == 0)
      instanteAtuacao = millis(); //inicia a contagem de tempo;
    else{
      unsigned long t = millis(); //armazena quanto tempo passou;

      while(t < (instanteAtuacao + 200)) //espera passar 200 ms para atuar;
        t = millis();

      instanteAtuacao = t; //atualiza instante de tempo da atuação
    }

    //Cálculo do erro entre a frequência atual e a desejada
   erro = CalcularPassos.setpoint - EstimarPitch.pitch;

    //Cálculo do termo integral
   CalcularPassos.Integral += (CalcularPassos.Ki * erro);
```

```
    dErro = (erro - erroAnterior); //Cálculo da variação do erro atual e o
anterior

    //Saída do PID é o número de passos que o motor dará
    CalcularPassos.passos = (CalcularPassos.Kp * erro) + CalcularPassos.Integral
+ (CalcularPassos.Kd * dErro);

    erroAnterior = erro;  //Salvando o valor atual do erro
}

void controleMatematico() { //função que calcula os passos que o motor deve dar

    //cálculo do quanto a corda deve esticar
    CalcularPassos.deltaLCorda =
CalcularPassos.vCorda[ComunicaSoftware.corda]*((CalcularPassos.setpoint*CalcularPa
ssos.setpoint)-(EstimarPitch.pitch*EstimarPitch.pitch));

    CalcularPassos.angulo =
CalcularPassos.deltaLCorda/CalcularPassos.vRaio[ComunicaSoftware.corda];

    //cálculo do ângulo que a tarraxa deve girar
    CalcularPassos.anguloGraus = (GRAUS_VOLTA_COMPLETA*CalcularPassos.
angulo)/(2*VALOR_PI);

    //quantos passos para atingir o ângulo
    CalcularPassos.passos = (CalcularPassos.
anguloGraus*NUMERO_PASSOS)/GRAUS_VOLTA_COMPLETA;
}

void atuaMotor() {

    if(CalcularPassos.passos > 0)
    {
     digitalWrite(SM_DIRECTION_PIN, LOW); //sentido anti horario, apertar a corda
     //Trem de impulsos para o motor atuar os passos necessários
     for (int p = 0; p < CalcularPassos.passos; p++){
       digitalWrite(SM_STEP_PIN, HIGH);
       delayMicroseconds(700);
       digitalWrite(SM_STEP_PIN, LOW);
       delayMicroseconds(700);
      }
     }
     else
     {
     digitalWrite(SM_DIRECTION_PIN, HIGH); //sentido horario, soltar a corda
      for (int p = 0; p > CalcularPassos.passos; p--){
       digitalWrite(SM_STEP_PIN, HIGH);
       delayMicroseconds(700);
```

```
      digitalWrite(SM_STEP_PIN, LOW);
      delayMicroseconds(700);
     }
    }
}
```

ANEXO A - Frequência e número MIDI das notas na escala temperada

Tabela A-1 - Frequência e número MIDI das notas na escala temperada.

B	*A#*	*A*	*G#*	*G*	*F#*	*F*	*E*	*D#*	*D*	*C#*	*C*	*Notas*
61,74	58,27	55	51,91	49	46,25	43,65	41,20	38,89	36,71	34,65	32,70	**1**
35	34	33	32	31	30	29	28	27	26	25	24	
123,47	116,54	110	103,83	98	92,50	87,31	82,41	77,78	73,42	69,30	65,41	**2**
47	46	45	44	43	42	41	40	39	38	37	36	
246,94	233,08	220	207,65	196	185	174,61	164,81	155,56	146,83	138,59	130,81	**3**
59	58	57	56	55	54	53	52	51	50	49	48	
493,88	**466,16**	**440**	**415,30**	**392**	**369,99**	**349,23**	**329,63**	**311,13**	**293,66**	**277,18**	**261,62**	**4**
71	70	69	68	67	66	65	64	63	62	61	60	
987,77	932,33	880	830,61	783,99	739,99	698,46	659,26	622,25	587,33	554,36	523,25	**5**
83	82	81	80	79	78	77	76	75	74	73	72	
1975,53	1864,65	1760	1661,22	1567,98	1479,98	1396,91	1318,51	1244,51	1174,66	1108,73	1046,50	**6**
95	94	93	92	91	90	89	88	87	86	85	84	
3951,06	3729,31	3520	3322,44	3135,96	2959,95	2793,82	2637,02	2489,01	2349,32	2217,46	2093	**7**
107	106	105	104	103	102	101	100	99	98	97	96	
7902,43	7458,62	7040	6644,87	6270,93	5919,90	5587,65	5274,04	4978,06	4698,64	4434,92	4186	**8**
119	118	117	116	115	114	113	112	111	110	109	108	
				12543,85	11839,82	11175,30	10548,08	9956,06	9397,27	8869,84	8372,02	**9**
				127	126	125	1234	123	122	121	120	

Oitavas / Número da nota

Fonte: COCA, 2004.