UNIVERSIDADE LUTERANA DO BRASIL

PRÓ-REITORIA DE GRADUAÇÃO

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Diego Farias de Fraga

Projeto de um Equipamento para Ensaios de Vibração de Máquinas Rotativas

Canoas, Dezembro de 2009.

Diego Farias de Fraga

Projeto de um Equipamento para Ensaios de Vibração de Máquinas Rotativas

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia Elétrica da ULBRA como um dos requisitos obrigatórios para a obtenção do grau de Engenheiro Eletricista

Departamento:

Engenharia Elétrica

Área de Concentração

Instrumentação

Professor Orientador:

Prof. ME. Augusto Alexandre Durgante de Mattos– CREA-RS: 088.003-D

Canoas

2009

FOLHA DE APROVAÇÃO

Nome do Autor: Diego Farias de Fraga

Matrícula: 021002669-3

Título: Projeto de um Equipamento para Ensaios de Vibração em Máquinas Rotativas

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia Elétrica da ULBRA como um dos requisitos obrigatórios para a obtenção do grau de Engenheiro Eletricista

Professor Orientador:

Prof. ME. Augusto Alexandre Durgante de Mattos

CREA-RS: 088.003-D

Banca Avaliadora:

ME. Paulo César Cardoso Godoy

CREA-RS: 116.822-D

Conceito Atribuído (A-B-C-D):

ME. Miriam N. Cáceres Villamayor

CREA-RS: 067.231-D

Conceito Atribuído (A-B-C-D):

Assinaturas:

Autor Diego Farias de Fraga	Orientador Augusto Alexandre Durgante de Mattos
Avaliador Miriam N. Cáceres Villamayor	Avaliador Paulo César Cardoso Godoy

Relatório Aprovado em:

DEDICATÓRIA

A minha família, por todos os esforços e sacrifícios passados para que o dia de hoje se tornasse realidade.

AGRADECIMENTOS

Agradeço a três pessoas em especial que estão me acompanhando desde o momento de meu nascimento. À minha mãe, meu exemplo de vida, que me ensinou a lutar e conquistar dignamente tudo que eu sonhava. Ao meu pai, sempre preocupado em saber como estavam os estudos, nunca se importando em acordar às cinco horas da manhã para me levar a parada em dias chuvosos. Foi ele que me ensinou o verdadeiro valor do trabalho. À minha irmã, que foi acima de tudo, a minha amiga que sempre esteve do meu lado quando achei que o mundo poderia desabar.

A vocês tudo que eu faça para tentar retribuir o esforço e dedicação dispensada a mim será pouco.

Agradeço a minha namorada, Fabiane, por toda a paciência dispensada ao longo desses quatro anos em que estamos juntos. Sei que não foi fácil agüentar-me, todo o final de semestre, mas ela sempre estava ao meu lado, nunca duvidando de meus objetivos. A você só tenho a agradecer por estar do meu lado.

Ao meu orientador Professor Augusto pelo apoio, dedicação e esforço pessoal dispensado.

Gostaria de agradecer ao Engenheiro Victor Guidobono, pela oportunidade de crescimento profissional fornecida durante o estágio. Agradecer também ao Sr. Carlos Canteli pela dedicação e tempo dispensados na confecção do protótipo utilizado neste projeto.

A todos os colegas que tive o prazer de cursar alguma disciplina em comum.

A todos os professores que tive o privilégio de ser aluno, não somente pelo ensino acadêmico, mas por proporcionarem outra visão de mundo durante minha formação acadêmica.

EPÍGRAFE

"Os homens perdem a saúde para juntar dinheiro,

depois perdem o dinheiro para recuperar a saúde.
E por pensarem ansiosamente no futuro esquecem do presente

de forma que acabam por não viver nem no presente nem no futuro.

E vivem como se nunca fossem morrer...

e morrem como se nunca tivessem vivido."

Dalai Lama

RESUMO

Autor: Diego Farias de Fraga de. **Projeto de um Equipamento para Ensaios de Vibração de Máquinas Rotativas.** Trabalho de Conclusão de Curso em Engenharia Elétrica - Departamento de Engenharia Elétrica. Universidade Luterana do Brasil. Canoas, RS. 2009.

Este trabalho apresenta o desenvolvimento de um equipamento capaz de adquirir o sinal de vibração mecânico de máquinas rotativas e visualizar seu sinal no domínio tempo e no domínio frequência. Essa visualização ocorre em um software desenvolvido no MATLAB. Como placa de aquisição do sinal analógico foi utilizada a placa de som do computador PC. Uma mesa de vibração foi testada utilizando a placa de som do computador PC como geradora do sinal. Esse sinal é criado através de um software desenvolvido em ambiente MATLAB, possibilitando a escolha da forma, frequência e amplitude do sinal. Para a realização de testes que simulassem falhas reais, foi desenvolvido um protótipo de máquina rotativa. Neste protótipo são simuladas falhas como desbalanceamento e defeito em rolamentos. A rotação do protótipo pode ser variada, o que possibilita que se explorem algumas faixas de frequências distintas. Os resultados experimentais mostram que o sistema proposto para a aquisição e visualização do sinal tem um bom desempenho, já a mesa de vibração não possui uma resposta linear no que diz respeito à amplitude de vibração desejada.

Palavras chave: Vibração. MATLAB. Placa de Som. Acelerômetro. Mesa de Vibração. Gerador de Sinais. *Data Acquisition Toolbox*.

ABSTRACT

Author: Fraga, Diego Farias de. **Project of an Equipment for Assays of Vibration of Rotative Machines.** Work of Conclusion of Course in Electrical Engineering - Electrical Engineering Department. Lutheran University of Brazil. Canoas, RS. 2009.

This work presents the development of an equipment capable to acquire the mechanical signal of vibration of rotating machines and to visualize its signal in the domain time and the domain frequency. This visualization occurs in a software developed in the MATLAB. As card of acquisition of the analogical signal the sound card of computer PC was used. A vibration table was tested using the sound card of computer PC as generator of the signal. This signal is created through a software developed at MATLAB environment, having made possible the choice of the form, frequency and amplitude of the signal. For the accomplishment of tests that simulated real imperfections, a prototype of rotating machine was developed. In this prototype imperfections can be simulated as unbalancing and defect in bearings. The rotation of the prototype can be varied, which makes it possible to explore some different frequency bands. The experimental results show that the considered system for the acquisition and visualization of the signal has a good performance, but the vibration table does not have a linear reply regarding the desired vibration amplitude.

Keywords: Vibration. Sound Card. Accelerometer. Vibration Table. Signs Generator, Data Acquisition Toolbox.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

FFT: Fast Fourier Transform

MATLAB: Matrix Laboratory

DSP: Digital Signal Processor

MEMS: Micro Electro Mechanical Systems

g: gravidade

mm/s: milímetros por segundo

Vdc: Tensão continua

Hz: Hertz

Vdc: Tensão em corrente continua

rpm: Rotações por minuto

SUMÁRIO

1. INTRODUÇÃO

A globalização estende-se hoje no mundo todo, em todos os seus setores e mais do que nunca na indústria mundial, seja do ramo madeireiro, metalúrgico ou qualquer outra atividade industrial. A indústria que for competitiva sobrevive nessa selva da globalização e acaba se agigantando a frente daquelas que não conseguiram por algum motivo se adaptar a esse processo mundial. Então o papel da manutenção chega a ser em muitos casos fator decisivo para o fortalecimento de uma indústria. Chegou-se então a era da manutenção moderna, onde não se tolera a parada de um processo de forma não programada e custos de manutenção devem ser sempre revisados buscando uma redução.

A manutenção, como os demais setores da indústria, tem exigido constante aprimoramento na qualidade de seus serviços, em face de disputa do mercado. Nas últimas décadas, esta prática tem obrigado as indústrias cada vez mais a reduzirem seus custos operacionais. Estes custos são afetados pela eficiência de um sistema de manutenção. Com isso as Empresas, através do departamento de manutenção, vêm dando cada vez mais apoio ao desenvolvimento, aprimoramento e implantação de tecnologias aplicadas à manutenção.

A idéia de parar uma máquina para abri-la, inspecioná-la ou esperar que venha a falhar pode ser um grande risco para a produção. Baseado neste conceito e considerando-se os aspectos de segurança, confiabilidade, desempenho e disponibilidade, conclui-se que o ideal é que a manutenção intervenha na máquina apenas quando e onde se fizer necessário. No entanto a manutenção preditiva se propõe, através de técnicas de diagnóstico, estabelecer parâmetros confiáveis para avaliação do estado real dos componentes e, com isto, prever uma falha, ao invés de apenas presumi-la.

Um dos mais importantes métodos de predição utilizado na indústria principalmente quando se trata de equipamentos rotativos (bombas, turbinas, redutores, ventiladores, compressores) é a chamada técnica de análise vibracional.

Várias empresas possuem apenas o monitoramento da aceleração de seus equipamentos em nível global, não realizando a análise do espectro de vibração. Isso se deve ao alto custo de aquisição de equipamentos e *softwares* específicos para a realização deste tipo de análise.

Com a evolução dos computadores *desktops* e a criação de *softwares* de programação voltados para o cálculo matemático (como o MATLAB), criou-se a possibilidade de se desenvolver um algoritmo que realize a função de analisar os dados recebidos do sensor de vibração.

Outro fator importante é o desenvolvimento de novas tecnologias na confecção dos sensores de aceleração. Os sensores utilizados no passado possuíam um alto custo, como no caso dos acelerômetros piezelétricos (os mais utilizados na indústria atualmente). Atualmente já existem acelerômetros de tecnologia MEMS (*Micro Electro Mechanical Systems*) concorrendo com um custo menor. Utilizando-se um acelerômetro do tipo MEMS, foi desenvolvido um equipamento de baixo custo que possibilite realizar a aquisição do sinal de vibração e também a geração de um sinal de vibração mecânica através de uma mesa de vibração controlada pelo computador.

Esse equipamento é composto por uma etapa de condicionamento do sinal proveniente do acelerômetro e um *software* implementado em MATLAB, através do qual é possível tanto controlar a mesa de vibração como realizar a visualização do sinal de vibração mecânica a ser medido.

O presente trabalho está divido da seguinte forma. No capítulo dois foi descrito uma breve análise sobre o que é a vibração mecânica, quais são suas principais características e quais são as grandezas mais importantes. Também veremos quais são os principais tipos de transdutores utilizados para a medida de vibração. O capitulo três apresenta o descritivo do que foi realizado durante o projeto, constando as principais características de cada etapa do projeto. No capitulo quatro consta os resultados dos testes propostos, com o objetivo de validação do projeto.

2. VIBRAÇÃO MECÂNICA

2.1. Breve Histórico Sobre a Análise de Vibração

O uso da análise de vibração é uma ferramenta fundamental para o monitoramento da condição de operação de equipamentos e vem sendo desenvolvida ao longo de aproximadamente 35 anos. Com o desenvolvimento paralelo de equipamentos eletrônicos, transdutores, computadores e *softwares a* supervisão dos equipamentos hoje é praticamente toda automatizada. De 1960 a meados de 1970 o uso de simples métodos de análise era utilizado, juntamente com uma cuidadosa observação sobre o comportamento do equipamento, muitas vezes reforçados por manutenções freqüentes. Instrumentos simples por muitas vezes eram utilizados para medir e registrar os níveis de falha em que a manutenção dos equipamentos era baseada. Isso exigiu profissionais de manutenção altamente qualificados e experientes para assegurar a eficiência da operação e assim evitando falhas catastróficas.

Durante a década de 70 houve a evolução da instrumentação analógica e computadores. Diversos acelerômetros, transdutores de velocidade e deslocamento foram desenvolvidos e adaptados às necessidades das aplicações industriais. A instrumentação analógica se tornou popular através de medidores portáteis de vibração e equipamentos de gravação do sinal através de "fitas" FM, que posteriormente eram estudadas em analisadores de frequência. Muitos desses equipamentos eram pesados e complexos, mas provaram com maior precisão o registro e análise da vibração. Se uma empresa tinha acesso a um computador mainframe, os dados poderiam ser analisados, e assim uma estratégia para a manutenção poderia ser desenvolvida.

Embora alguns instrumentos digitais estavam disponíveis durante o inicio dos anos 1970, uma evolução significativa ocorreu durante o final dos anos 70 início dos anos 80, devido à disponibilidade de novos microprocessadores. Circuitos puderam ser miniaturizados, reduzindo assim as dimensões e peso dos instrumentos e permitiu que os dados fossem processados em altas velocidades. Microprocessadores *onboard* deram a capacidade a instrumentos de capturar os dados, analisá-los através de algoritmos adequados e, em seguida, armazenar e exibir as informações. A análise da frequência através do eficiente cálculo da FFT

(Fast Fourier Transform) de múltiplos canais dentro de segundos e a capacidade de armazenar dados para futuras decisões foram outras características que se somaram a análise de vibração. Em longo prazo o armazenamento de dados se tornou uma pratica aceita.

Desde meados dos anos 80 a evolução tem sido associada com o computador *desktop* e sua interface de *software*. Muitos fabricantes têm produzido instrumentos portáteis para medição instantânea, gravação e análise de variáveis. A informação é, muitas vezes, transferida do instrumento portátil para um computador *desktop* possibilitando uma análise estatística dos dados. Isto torna mais fácil a decisão sobre a estratégia de manutenção, o profissional que analisa os dados não necessita mais de um alto nível de habilidade.

A década de 90 se caracteriza por minimizar a instrumentação e as aquisições de dados de processo. A instrumentação é simples de operar, e os problemas podem ser analisados com o auxilio de sistemas baseados em conhecimento. Em meados da década de 90 foi iniciado o monitoramento remoto com uma unidade de controle central para interligar os conhecimentos e experiências dos clientes, fabricantes de equipamentos e os seguros de responsabilidades das empresas.

Embora a análise de vibração tenha sido utilizada principalmente para determinar falhas e pontos críticos de operação, atualmente a exigência do acompanhamento da condição de vibração não está mais limitada a tentar minimizar as consequências de equipamentos problemáticos, mas sim de utilizar esses recursos existentes de forma mais eficaz.

O aumento exponencial das capacidades de computação e cálculos matemáticos, o aumento da velocidade da telecomunicação permite que novas análises técnicas sejam implementadas. No processo moderno de sistemas de vigilância como a lógica *FUZZY* e redes neurais surgem como ferramentas para um sistema automático de controle, classificação e sinalização. A forma final desta abordagem é um sistema especializado, que eventualmente, pode levar a uma condição totalmente automatizada.

2.1. Vibração

Vibrações mecânicas são, em geral, movimentos oscilatórios de uma massa (ou massas) em torno de uma posição de referência. Em uma máquina, vibração é o resultado de forças dinâmicas internas não compensadas, criadas por elementos rotativos, oscilantes ou de deslocamento aleatório [1]. Os movimentos podem ser caracterizados em termos de:

- **Deslocamento:** é à distância do afastamento da massa de sua posição natural em metros (m).
- **Velocidade:** é a derivada do deslocamento (taxa de variação da distância) com que a massa se movimenta em metros por segundo (m/s).
- **Aceleração:** a taxa de mudança de velocidade da massa, em metros por segundo ao quadrado (m/s^2).

2.1.1. Movimento Harmônico (deslocamento)

Movimento harmônico, expresso em termos de deslocamento, é a forma de vibração mais simples. Quando analisado em função do tempo, é representada por uma função senoidal como apresentada na Figura 1. O movimento instantâneo de um corpo vibrando em torno de uma posição de referência pode ser descrito matematicamente pela equação:

$$x(t) = X \sin \omega t$$

eq. (2.1-1)

Onde: ***x(t)*** é o valor instantâneo do deslocamento;

X é a amplitude máxima do movimento, também representada por $\boldsymbol{x}_{pico}$ e

ω é a velocidade angular.

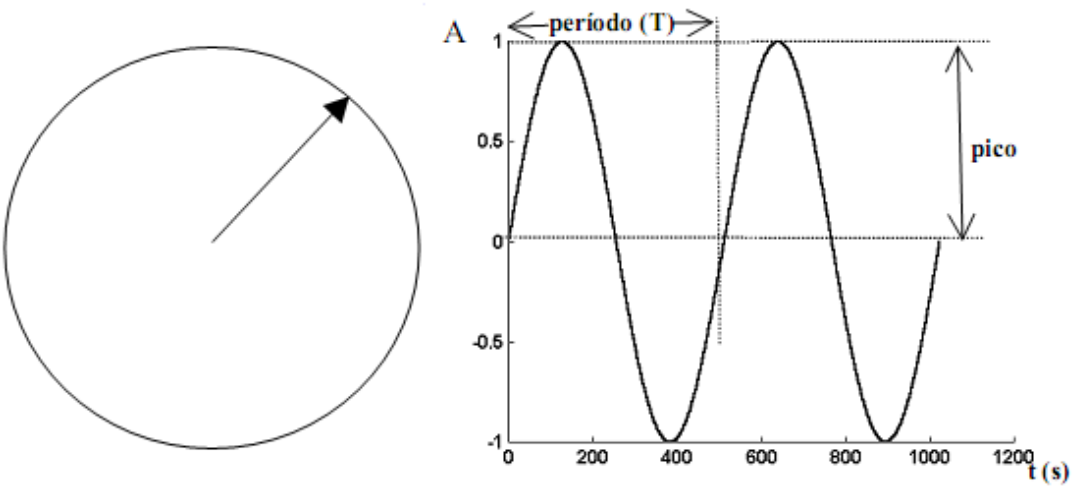


Figura 1 - Movimento Harmônico Simples
Fonte: Marçal (2000)

2.1.2. Velocidade e Aceleração

Na Figura 2, o movimento harmônico foi descrito em termos de movimento ou deslocamento. De fato, o movimento harmônico tem outras duas propriedades usadas na análise vibracional de máquinas: *velocidade* e *aceleração*, conforme mostra a Figura 2.

Velocidade é a taxa de variação do deslocamento em relação ao tempo.

$$v(t) = \frac{dx(t)}{dt}$$

eq. (2.1-2)

Aceleração é a taxa de variação da velocidade em relação ao tempo ou a segunda derivada no tempo do deslocamento.

$$a(t) = \frac{dv(t)}{dt} = \frac{d^2x(t)}{dt^2}$$

eq. (2.1-3)

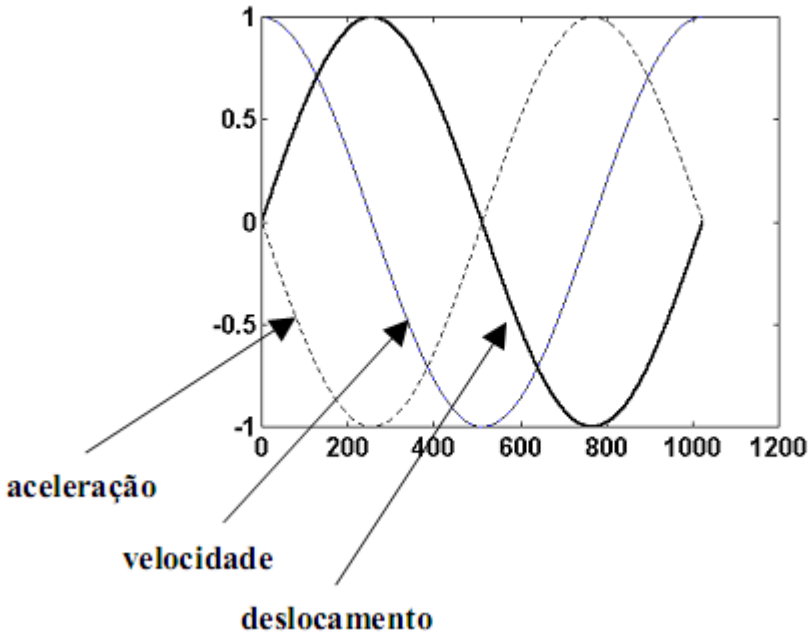


Figura 2 – Relação entre deslocamento, velocidade e aceleração
Fonte: MARÇAL (2000)

As equações 2.1-1, 2.1-2 e 2.1-3 representam a definição geral de velocidade e aceleração. No caso específico do movimento harmônico, e somente neste caso, a relação entre deslocamento, velocidade e aceleração é simplificada conforme as equações abaixo:

$$x(t) = x_{pico} \sin \omega t$$

eq. (2.1-4)

$$v(t) = \left(\omega x_{pico}\right) \cos \omega t = v_{pico} \sin(\omega t + \pi/2)$$

eq. (2.1-5)

$$a(t) = -\left(\omega^2 x_{pico}\right) \sin \omega t = a_{pico} \sin(\omega t + \pi)$$

eq. (2.1-6)

Como indicado na Figura 2, a velocidade conduz ao deslocamento por um ângulo de fase de 90°. A aceleração conduz à velocidade por um ângulo de fase de 90° e ao deslocamento por um ângulo de 180°.

Esta variação nos valores de velocidade e aceleração com a frequência é extremamente importante para formar as bases para critérios mais rigorosos de análise vibracional. Serve, também, para balizar a seleção da variável que será mais representativa para detecção e análise de uma determinada falha. Entretanto,

cabe ressaltar que falhas em equipamentos podem ocorrer sem que nenhum sinal de vibração anormal tenha sido detectado.

2.1.3.Amplitude de Vibração

Diferentes valores numéricos podem ser usados para caracterizar aamplitude do movimento, conforme ilustra a Figura 3.

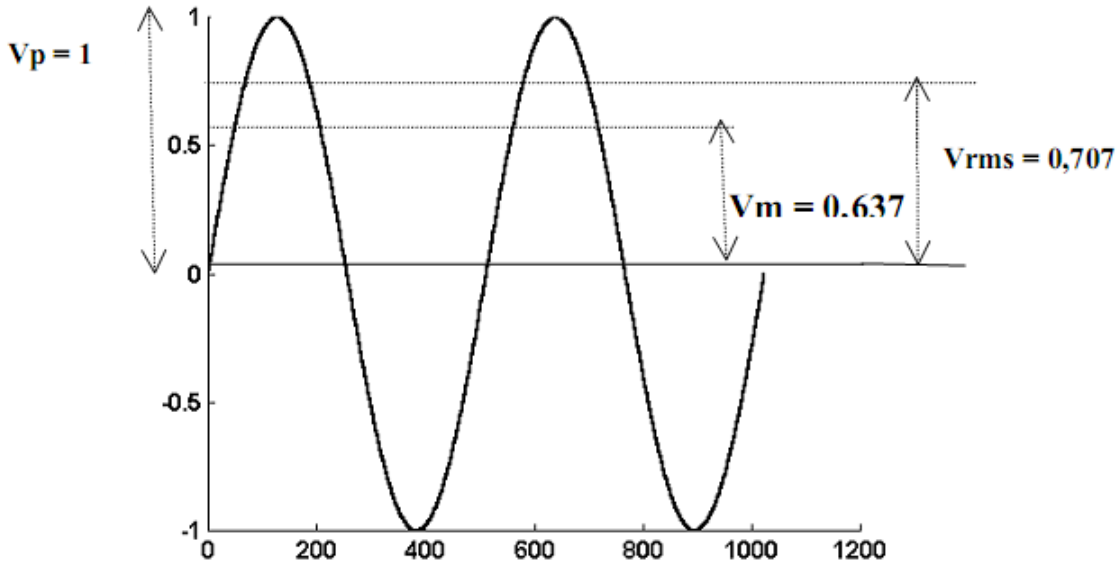


Figura 3 - Formas de expressar a amplitude de um movimento harmônico
Fonte: MARÇAL (2000)

Sendo:

$\boldsymbol{V_{pico}}$: o valor da amplitude do movimento;

$\boldsymbol{V_{pico-a-pico}}$: a diferença entre os valores máximos positivo e negativo do movimento;

$\boldsymbol{V_{médio}}$: o valor médio que é definido por:

$$V_m = 1/T \int_0^T x(t)dt$$

eq. (2.1-7)

$\boldsymbol{V_{rms}}$ **:** o valor rms ("*root mean square*") ou eficaz o qual é definido por:

$$V_{rms} = \sqrt{1/T \int_0^T x^2(t)dt}$$

eq. (2.1-8)

Para movimento harmônico puro todos estes valores estão relacionados pela amplitude e podem ser determinado um a partir do outro.

Vibrações reais encontradas na análise de máquinas raramente são puramente harmônicas. A vibração é normalmente uma combinação de vários movimentos harmônicos, de diferentes amplitudes e frequências, e movimentos aleatórios (aqueles cujos valores instantâneos não podem ser descritos por uma equação matemática e onde somente suas propriedades estatísticas são conhecidas).

O valor eficaz (ou valor rms) do sinal oferece uma estimativa da energia contida na vibração, por isso, é preferido em relação ao valor médio. É um parâmetro largamente utilizado para a estimativa da gravidade da vibração em carcaças de máquinas ou medidas externas. O valor eficaz exige um detector de valor eficaz. A simples conversão partindo-se do resultado obtido através de um detector de valor médio (ou seja, multiplicando-se por 1,11), conduz a resultados errôneos, quando o sinal de entrada não for uma onda senoidal.

O valor de pico tem vantagens quando for necessário determinar se a vibração é de natureza impulsiva, tal como, a originada de engrenagens ou mancais de elementos rolantes. O valor de pico deve ser medido com um detector de valor pico, também se evitando, através do escalonamento do resultado obtido por um detector de valor eficaz (multiplicando-se por $\sqrt{2}$), que se subestime ou se obtenha uma análise errada do verdadeiro valor.

O valor pico-a-pico pode ser utilizado quando for medido o deslocamento relativo de um eixo dentro de um mancal. Neste caso, o valor pico-a-pico representa o percurso do eixo e pode ser diretamente comparado com a folga do mancal. Mais uma vez, deve-se ter cuidado na obtenção do valor pico-a-pico, não bastando multiplicar um dos valores de pico por 2. Uma boa regra é sempre graficar um valor de amplitude em função do seu tipo de detector, seja este rms, pico ou pico-a-pico.

2.2. *Transdutores de Vibração*

A escolha do parâmetro para uma medição particular depende da natureza da vibração e do propósito da medida. Considerando que, **deslocamento** é usualmente preferido para medições de vibrações em baixa frequência, tais como, desbalanceamento de máquinas e vibrações de suportes estruturais, **aceleração** é escolhida para medição de choques e vibrações em alta frequência onde os primeiros sinais de desgaste e fadiga da máquina geralmente aparecem. **Velocidade**, por outro lado, oferece uma indicação da vibração, sendo o parâmetro indicado em monitoração da condição de máquinas e em programas de manutenção preventiva [4].

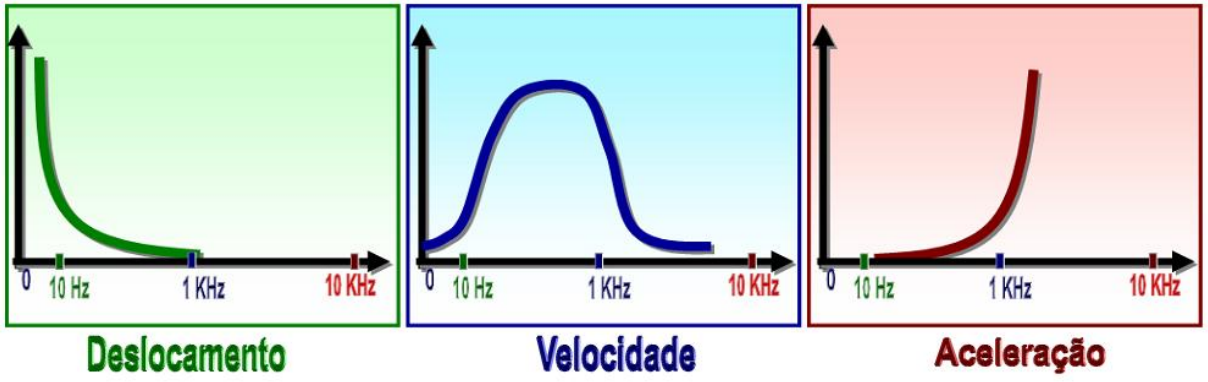


Figura 4 – Faixa de frequência para cada parâmetro de medição.
Fonte: SEMEQ (2006)

Transdutores de vibração são dispositivos que convertem movimento mecânico em um sinal elétrico (AC) dinâmico. Geralmente, necessitam de um condicionamento eletrônico que adapta o sinal elétrico para transmissão ou uso por instrumentos de monitoração, analisadores, monitores e aparelhos de gravação. Os transdutores tradicionalmente usados para investigação de vibração são: os Medidores de deslocamento sem contato (como o "*eddy current probe*"); os Sensores de velocidade (eletromecânicos ou piezoeléctricos) e os Acelerômetros.

2.2.1. Medidores de Deslocamento

O medidor de deslocamento é um sistema composto de um transdutor de deslocamento (*probe*), um cabo de extensão e um oscilador/demodulador. O

transdutor de deslocamento consiste de uma bobina de cobre montada em um material não condutor, plástico ou cerâmico, acoplado a um corpo cilíndrico rosqueado, para que possa ser fixado a um suporte. Em operação, o transdutor é excitado com uma frequência em torno de 1500 Hz, gerada pelo oscilador e transmitida pelo cabo de extensão. Quando a extremidade do transdutor é posicionada próximo de um material condutor (superfície da máquina a ser monitorada), correntes de *eddy* são induzidas na superfície do material, extraindo energia da excitação do *probe* e modificando a amplitude do sinal, isto é, modulando em amplitude o sinal de 1500 Hz. Como a distância entre a extremidade do transdutor e a superfície observada, conhecida como *gap,* é variável (por estar vibrando) o sinal modulante representa a vibração, que é expressa na saída do demodulador como uma tensão DC proporcional.

Os transdutores de deslocamento sem contato (*noncontact displacement transducer*) têm alcançado grande aceitação no monitoramento e proteção de máquinas industriais. São efetivos para o monitoramento de máquinas com mancais de rolamento ou deslizamento. Indicam o movimento do eixo e a posição relativa do mesmo ao mancal. O deslocamento radial do eixo (valor pico-a-pico) pode ser relacionado diretamente com a folga radial no mancal. A componente estática (DC) de um sistema de medição de vibração de proximidade representa a posição média do eixo em relação ao transdutor de deslocamento. Normalmente, utilizada para monitorar a posição axial o eixo, a componente estática, também é importante, para o monitoramento radial. Neste caso, esta pode localizar a posição média da linha de centro longitudinal do eixo dentro do mancal. O monitoramento da posição axial serve, ainda, para acompanhar a posição relativa do rotor em relação aos componentes estacionários da máquina.

Pode-se apontar como uma limitação do uso do transdutor de movimento a dificuldade que este tem em monitorar um eixo que apresente irregularidade na superfície do material, como a periferia de um eixo. Este sistema não consegue distinguir entre o movimento do eixo (a vibração) e as imperfeições na superfície deste (*runout*). As imperfeições podem ser crateras, dentes, excentricidade do eixo, variações de propriedades eletromagnéticas ou variações na condutividade e permeabilidade do material. Como consequência, o sinal de saída é o vetor da soma da vibração e de todas as imperfeições da superfície observada pelo sensor. Considerando que o campo magnético produzido pelo sensor penetra a superfície do material monitorado qualquer heterogeneidade produzirá certa distorção no sinal de saída. Isto pode alterar a configuração da forma de onda como pode aumentar a

amplitude da vibração das harmônicas da frequência (velocidade) de giro da máquina.

2.2.2.Sensores de Velocidade

Existem basicamente dois tipos de sensores de velocidade, o eletromecânico e o piezoeléctrico. O sensor de velocidade piezoeléctrico é basicamente um acelerômetro piezoeléctrico com um circuito integrador internamente. O seu funcionamento será relatado no item 2.3.3 que tratará sobre os tipos de acelerômetros.

O sensor de velocidade eletromecânico consiste de um imã permanente e uma bobina, conforme ilustrado na Figura 5. No interior do sensor uma bobina cilíndrica, fixada ao seu corpo, envolve um imã permanente suspenso por molas. O sistema de sustentação das molas é projetado para uma frequência natural baixa, de forma que o imã permaneça estacionário (massa inercial) no espaço para frequências abaixo de 10 Hz, aproximadamente. Geralmente é colocado um óleo sintético para amortecer a frequência natural do sistema massa mola e para baixar sua resposta para valores inferiores a 10Hz.

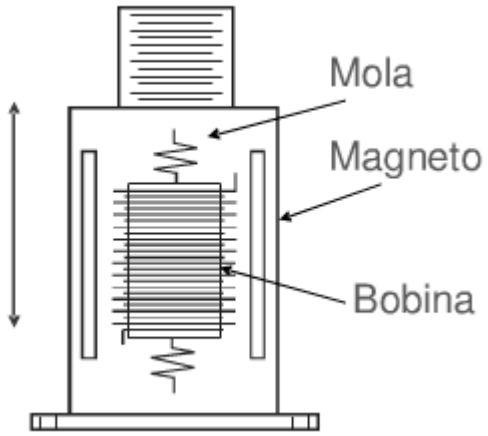


Figura 5 - Sensor de Velocidade Eletromecânico
Fonte: ProvibTech (2009)

O sensor é fixado à superfície que se deseja medir vibração. Quando esta vibra o movimento relativo entre o imã estacionário e a bobina gera um campo magnético variável através da bobina, induzindo uma tensão proporcional à velocidade absoluta da superfície. Um sensor de velocidade é um mecanismo autogerador que produz um sinal de baixa impedância. Este sinal pode ser usado

diretamente em um analisador ou equipamento de monitoração sem necessitar de fonte externa de energia.

O sensor de velocidade eletromecânico possui uma curva de sensibilidade x frequência de resposta limitada em baixas frequências pela frequência natural amortecida do sistema. A faixa de operação deste sistema, geralmente, está na ordem de 10 Hz a 1500 Hz.

2.2.3.Acelerômetros

O mais importante captador (*pickup*) para vibração, choques e medição de movimento é o acelerômetro. Este instrumento está disponível comercialmente em uma ampla variedade de tipos e modelos atendendo as mais variadas particularidades no âmbito de medição [5].

Sua resposta em frequência pode abranger desde alguns Hz até kHz, medições de movimentos transientes são melhores obtidas através de acelerômetros do que sensores de velocidade ou de deslocamento. Os sinais de velocidade ou deslocamento podem ser obtidos mais facilmente através do acelerômetro, bastando apenas se realizar uma integração elétrica que é mais estável que a diferenciação.

Existem vários tipos de acelerômetros. Cada um possui características únicas, vantagens e desvantagens. Dos acelerômetros mecânicos, os mais comuns são os capacitivos, os piezoeléctricos e os piezoresistivos. Recentemente, os acelerômetros mecânicos começam a ser substituídos por um novo tipo de acelerômetros, os eletromecânicos. Um exemplo é o acelerômetro MEMS.

Abaixo segue uma descrição do funcionamento de cada tipo de acelerômetro.

Acelerômetros capacitivos

Capacitores são componentes elétricos que armazenam carga. Um capacitor é geralmente formado dispondo duas placas paralelamente uma à outra. Os acelerômetros capacitivos contêm um capacitor entre a massa e a estrutura de suporte e são sensíveis às mudanças na capacidade entre estes. Uma aceleração da massa provoca variações no espaço entre a placa fixa e móvel do condensador, o qual é inversamente proporcional à carga do condensador. Como resposta à aceleração, a capacidade elétrica varia, o que faz variar o sinal de saída do circuito. [2]

Acelerômetros Piezoeléctricos

Os cristais piezoelétricos são cristais sintéticos ou naturais que produzem carga quando são comprimidos ou deflexionados. Nos acelerômetros piezoelétricos, a massa é unida a um cristal piezoelétrico. Quando o corpo do acelerômetro é sujeito à vibração, a massa obedece às leis da inércia e o cristal piezoelétrico fica submetido a forças de tração e compressão, gerando cargas. Estas forças são proporcionais á aceleração, de acordo com a lei de Newton. [2]

Acelerômetros Piezoresistivos

Neste tipo de acelerômetros, comparando com os piezoelétricos, surge um componente piezoresistivo em substituição ao cristal piezoeléctrico. A força exercida pela massa faz variar a resistência, que é detectada por uma ponte *Wheatstone*. Estes acelerômetros têm a vantagem de conseguir medir acelerações desde os 0 Hz. [2]

Microacelerômetros

A tecnologia dos sistemas mecânicos microeletrônicos, MEMS, é uma das mais excitantes no campo da tecnologia analógica. Explora as propriedades mecânicas do silício para criar estruturas móveis que, no caso dos sensores MEMS, detectam movimento (aceleração e vibração) em direções distintas. Estes sensores são sensíveis, compactos e baratos, e permitem acrescentar novas capacidades aos produtos, tornando-os mais funcionais e seguros [2].

A família dos sensores MEMS inclui acelerômetros que podem medir a aceleração e a vibração em um, dois ou três eixos. O sensor MEMS de um acelerômetro linear é baseado numa estrutura em silício, com interligações e em forma de pente composta por dedos fixos e móveis. A aceleração é obtida da medição dos deslocamentos de elementos móveis que estão associados aos eixos. O movimento medido pelo sensor é então convertido num sinal analógico ou digital [2].

3. CONDICIONAMENTO E AQUISIÇÃO DO SINAL

3.1. Descrição Geral do Projeto

O projeto realizado pode ser dividido em duas partes. A primeira consiste na aquisição do sinal de vibração mecânico, através de um acelerômetro. O acelerômetro converte o sinal mecânico em um sinal elétrico analógico. Uma placa de som de computador foi utilizada como placa de aquisição de sinal. O sinal foi digitalizado sendo tratado matematicamente através de um programa escrito para o software MATLAB.

A segunda parte do projeto consiste na geração de um sinal para a mesa de vibração com frequência e amplitude variáveis para posterior aquisição do sinal. O sinal é gerado pela placa de som é conectado diretamente à mesa de vibração.

A Figura 6 mostra um diagrama de blocos simplificado do projeto.

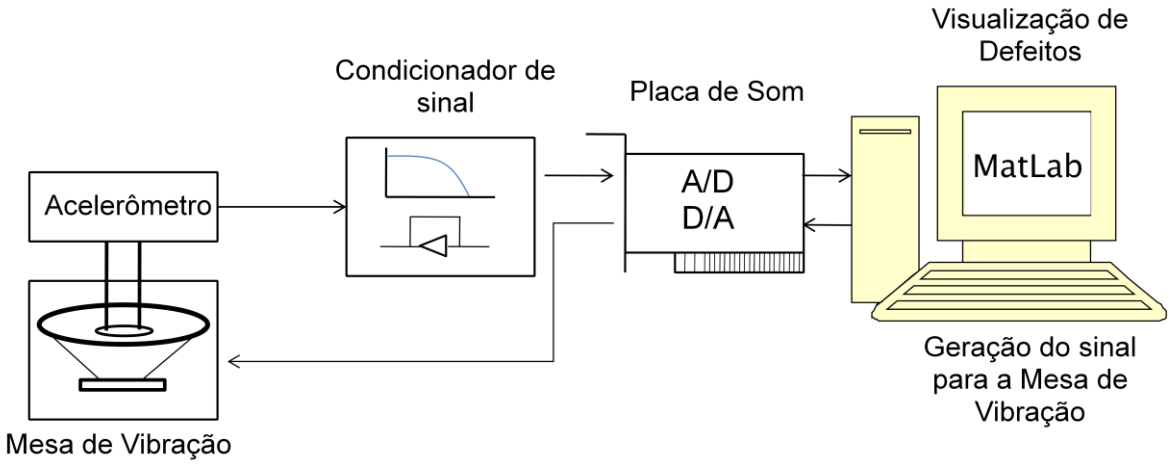


Figura 6 - Diagrama em blocos do projeto

O sinal de aceleração foi adquirido e após foi calculada FFT do sinal através do MATLAB. Assim o sinal é visualizado tanto no domínio tempo como no domínio frequência.

Sistemas comerciais de análise de vibração utilizam a mesma estrutura mencionada anteriormente. Em geral utilizam sistemas de aquisição próprios, e o *software* é o grande custo do sistema ainda mais se tratando de um sistema de monitoramento *online*.

3.2. Acelerômetro – Descrição do funcionamento

O acelerômetro utilizado neste projeto é o ADXL 321 da *Analog Device.* Ele utiliza a tecnologia MEMS e é capaz de adquirir sinais de até ±18g.

Na Figura 7 (a) podemos observar o diagrama em blocos do acelerômetro.

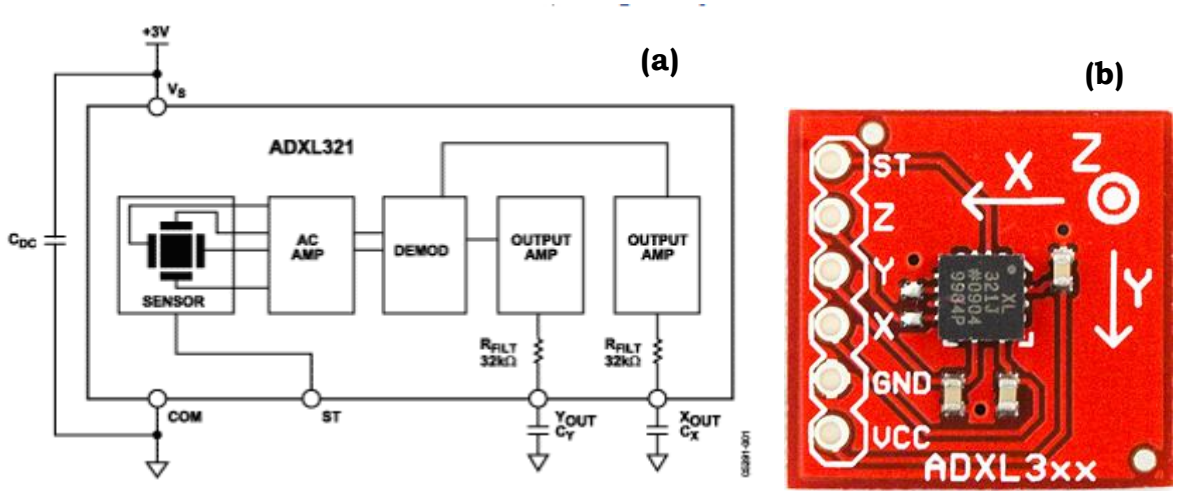


Figura 7 – (a) Diagrama em blocos do acelerômetro ADXL321 (b) Foto da placa montada pela SparkFun

Fonte: (a) Analog Device (2009) (b) SparkFun (2009)

A faixa de frequência máxima em que o acelerômetro responde é de 2.5kHz. Essa frequência foi obtida através da alteração do projeto original da placa adquirida na *Sparkfun Electronics. A*través da alteração do capacitor utilizado para desacoplamento do sinal que originalmente era usado para uma faixa de resposta em frequência de até 500Hz.

A equação 3.2-1 é fornecida no *datasheet* do acelerômetro e tem como finalidade a seleção dos capacitores para a implementação do filtro *antialiasing* e passa-baixas com o objetivo de redução de ruído e limitação da banda de frequência na saída do acelerômetro.

$$F_{-3\,dB} = 1/\left(2\pi(32k\Omega) \times C_{(X,Y)}\right)$$

eq. (3.2-1)

O esquema elétrico da placa é apresentado na Figura 8. O capacitor C2 foi alterado de 0,1µF para 2nF, assim foi obtida a máxima faixa de frequência em que o acelerômetro opera.

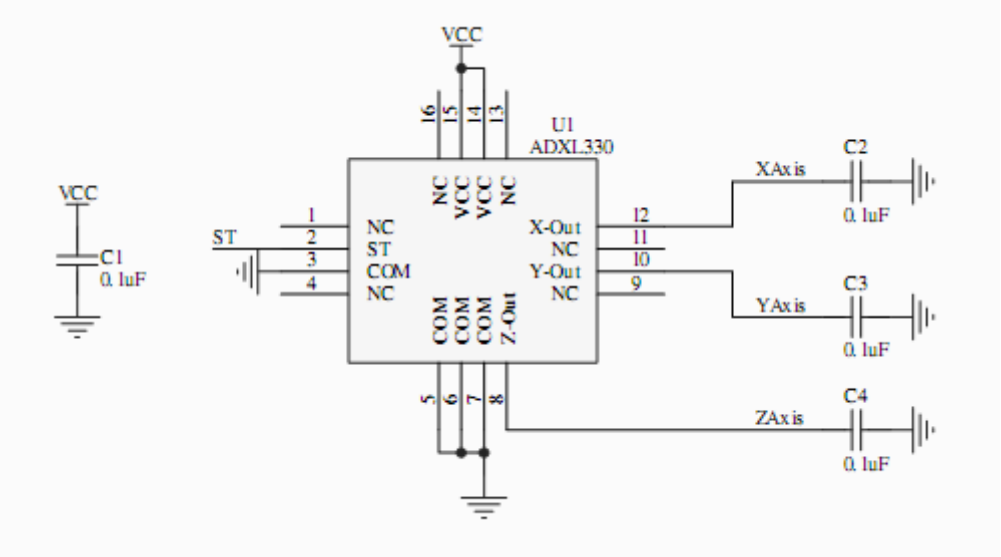


Figura 8 - Esquema elétrico placa montada – acelerômetro

3.2.1. Definição da sensibilidade do sensor.

A sensibilidade do sensor foi definida da seguinte forma. Posicionou-se o sensor de forma que seu eixo ficasse paralelo a mesa, assim na saída do acelerômetro realizou-se a medida de tensão equivalente a 0g. Posteriormente posicionou-se o sensor de forma que o mesmo ficasse perpendicular a mesa, assim obteve-se a medida de tensão equivalente a 1g e finalmente o sensor foi invertido 180º realizando a medida de tensão equivalente a -1g.

Todos os testes foram realizados com o sensor sendo alimentado com 5V. Os dados obtidos podem ser observados na tabela 3.2-2

Tabela 1 - Definição da sensibilidade do sensor

Posicionamento	**Medida de tensão (V)**
-1g	1,6205
0g	1,5573
1g	1,4940
Sensibilidade Obtida	63mV/g

Este teste se fez necessário, pois o sensor não apresentava a sensibilidade como descrita no *datasheet* quando comparada a outros acelerômetros.

Foi construída uma peça em acrílico para se obter uma melhor fixação da placa no objeto a ser medido, tentando se obter a máxima transferência de vibração com a maior fidelidade possível.

A foto da peça construída com o acelerômetro montado na placa pode ser observada na Figura 9.

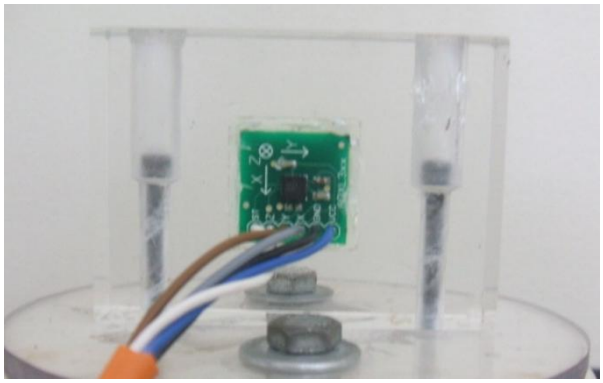


Figura 9 - Foto do Acelerômetro montado com base de fixação

3.3. *A placa de som como placa de aquisição*

Adquirir uma placa de aquisição de sinal geralmente envolve um alto custo, o que por muitas vezes acaba inviabilizando alguns projetos. Uma alternativa a essas placas de aquisição é a possibilidade de utilização da interface de áudio, ou multimídia, acoplada a um microcomputador PC. Essa opção torna-se viável, pois a presença deste tipo de interface é bastante usual nas configurações atuais de hardware. Apesar das limitações presentes por essa interface a mesma acaba sendo uma opção de baixo custo na realização de projetos acadêmicos.

No presente projeto, a placa de som do microcomputador PC, foi utilizada como um conversor AD e DA, assim possibilitando a interface do software presente no microcomputador PC com o mundo analógico.

3.3.1. Características Elétricas

Conforme mencionado anteriormente as limitações da placa de som devem ser observadas. Grandes partes dessas características são descritas no *datasheet* do *codec* da placa de som que pode ser visualizado no Anexo A.

As placas de áudio/multimídia encontradas atualmente no mercado apresentam, conforme ilustrado na Figura 10, as seguintes entradas/saídas.

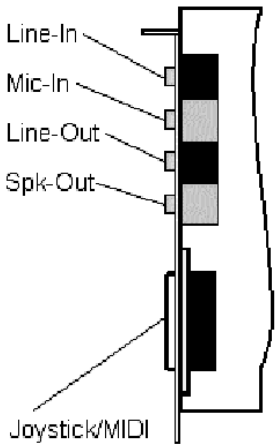


Figura 10 - Entradas/Saídas placa de som
Wikipédia(2009)

As características elétricas destas entradas e saídas estão listadas resumidamente conforme a Tabela 2.

Tabela 2 - Características Elétricas Placa de Som

Conector	**Tipo**	**Característica Típica**	**Impedância**
Line-In	Entrada de Sinal (Estéreo)	Sensibilidade ≅ 500 mV	10-47 kΩ
Mic-In	Entrada de Sinal (Mono)	Sensibilidade 5 - 500 mV	600-1500 Ω
Line-Out	Saída de Sinal (Estéreo)	Amplitude 0.5-2V pk-pk	10-47 kΩ
Spk-Out	Saída de Potência (Estéreo)	Potência 0.5-3W	≈ 8 Ω

Conforme a Tabela 2 a entrada de linha (*Line-In*) é uma entrada de sinal com baixa sensibilidade, sendo usada para adquirir sinais com amplitudes consideráveis. A saída de linha (*Line-Out*) corresponde a uma saída de áudio destinada à conexão com um amplificador enquanto a saída de potência (*Spk-Out*) é destinada à conexão direta com alto-falantes. Todas estas entradas e saídas

obedecem ao padrão usual de ligações (canal esquerdo/canal direito/neutro) usando *miniplugs* estéreos de 3.5 mm.

Uma atenção especial deve ser dada à entrada de microfone (*Mic-In*). Aqui a sensibilidade é apreciavelmente maior que a entrada de linha (Line-In), sendo tipicamente projetada para o uso em conjunto com microfones de eletreto. Essa sensibilidade varia conforme o ganho definido pelo usuário na entrada de microfone (*Mic-In*). Foi estabelecida uma sensibilidade alta, comparada a entrada de linha (*Line-In*) ajustando o volume do microfone ao nível mínimo (0%).

Como a impedância de entrada não é um item critico, pois temos o circuito de condicionamento de sinal que realiza essa compatibilidade, podemos trabalhar tanto com a entrada de microfone (*Mic-In*) ou com a entrada de linha (*Line-In*).

Outra característica importante da placa de som é o fato de que, tanto as entradas como as saídas, trabalharem apenas com sinais alternados. Assim não é possível adquirir sinais com níveis de tensão continuo, e também não é possível produzir sinais contínuos (como uma onda quadrada, ou uma onda triangular de baixa frequência).

3.3.2. Características de Frequência

A placa de som responde a uma determinada faixa de frequência correspondente a banda passante, essa faixa de frequência se estende por toda a faixa de áudio do espectro audível, ou seja, de 20 Hz a 20 k Hz.

Devido a filtros existentes tanto na entrada como na saída da placa de som, foi realizado o levantamento da resposta em frequência dos canais Line-Out e Mic-In, da placa de som que foi utilizada durante o projeto. Os resultados podem ser observados na Figura 11 e Figura 12.

O gráfico observado na Figura 11 foi obtido gerando um sinal através do gerador de sinais com uma amplitude constante de 315mV, enquanto se variava a frequência. Esse sinal foi observado no computador pessoal através do software de aquisição. Pode-se observar que não existe distorção em termos de amplitude do sinal adquirido.

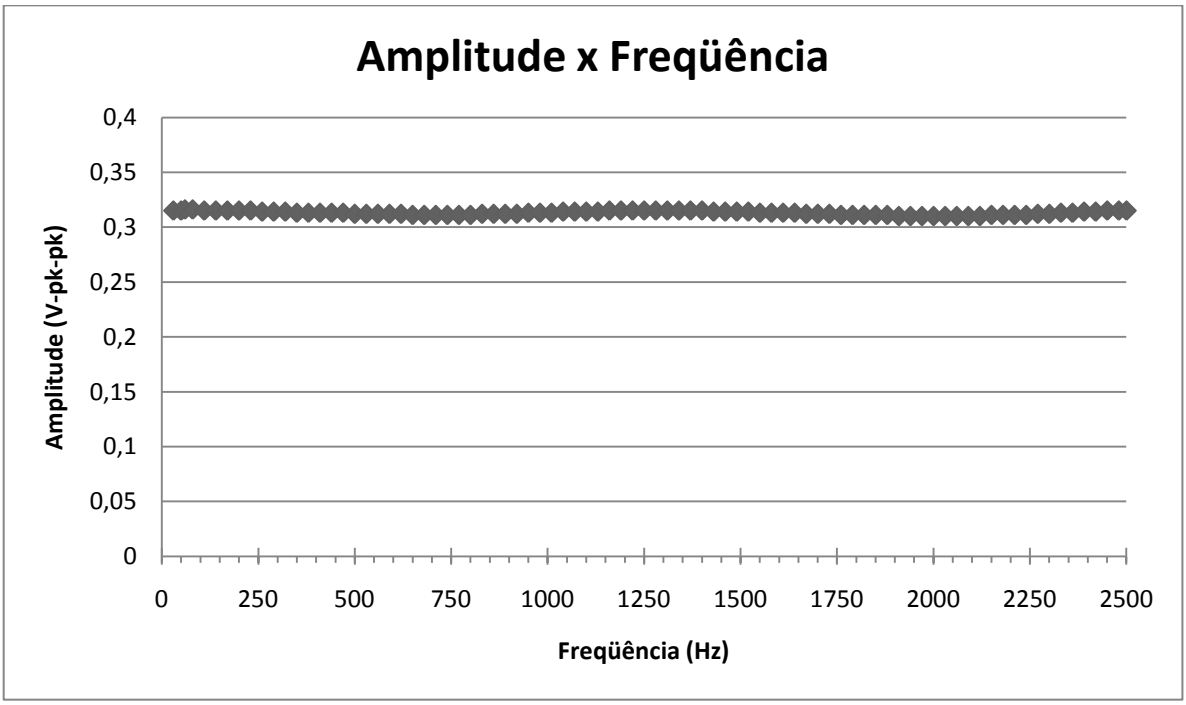


Figura 11 - Gráfico da entrada de Microfone (Mic-In)

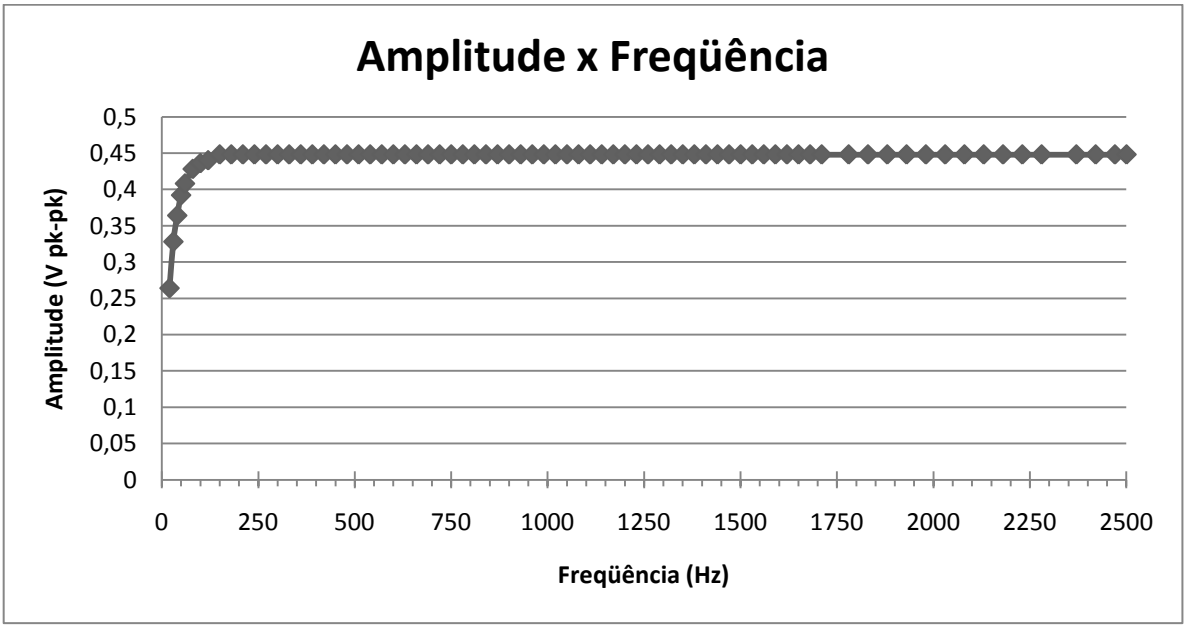


Figura 12 - Gráfico da resposta em frequência da saída de linha (Line-Out)

O gráfico da Figura 12 mostra a variação da amplitude em relação à frequência e foi obtido gerando-se uma onda senoidal através da placa de som com amplitude constante e variando-se a frequência. O sinal gerado foi observado no osciloscópio e sua amplitude pico-a-pico foi medida a cada variação de frequência

(foi utilizado um *step* de frequência de 10 Hz). Foi observado que a amplitude atinge seu máximo e se estabiliza com frequências superiores a 100 Hz.

Através dos gráficos observados acima (Figura 11 e Figura 12), estimamos a faixa de resposta da placa de som. É importante salientar que esses gráficos são de uma única placa de som, sendo assim os resultados observados não podem ser generalizados para qualquer placa de som.

3.3.3. Características do conversor AD e DA

Como mencionado anteriormente à placa de som foi utilizada como conversor analógico digital e conversor digital analógico.

O *codec* da placa de som, é o componente mais importante, pois nele está o conversor AD-DA que realiza as conversões. Atualmente é muito comum encontrarmos placa de som, com conversores AD-DA de 16 *bits*.

A maioria das placas de som permite regular a taxa de amostragem durante a aquisição do sinal, (no caso através da entrada Mic-In), essa taxa pode variar de 8000 amostras por segundo (*Sample Rate*) até 192000 amostras por segundo (*Sample Rate),* dependendo da taxa que é suportada pela placa de som. Essa taxa é corresponde a frequência de amostragem (fs).

Como foi utilizado um sinal de frequência máxima de 2500Hz, foi escolhida uma taxa de amostragem de 44100 amostras por segundo (fs), pois a placa que está sendo utilizada, aceita apenas duas taxas de amostragem, de 44100 amostras por segundo ou 192000 amostras por segundo.

3.4. Condicionador do Sinal

O circuito condicionador de sinal tem como finalidade adaptar o sinal proveniente do acelerômetro para que o mesmo fique em condições de ser adquirido pela placa de som. A Figura 13 apresenta o diagrama de blocos do circuito condicionador.

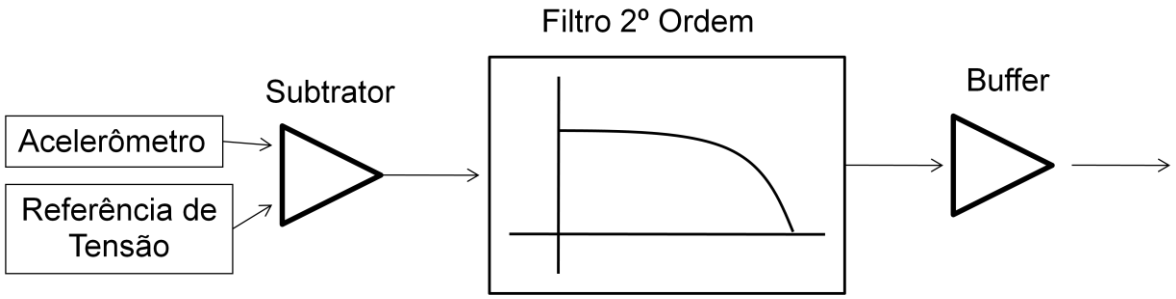


Figura 13 - Diagrama de blocos do circuito condicionador

Como visto anteriormente a placa de som admite apenas sinais alternados de 1V a -1V. Como o acelerômetro fornece o sinal de tensão apenas em níveis positivos (de 0V a 5V) foi necessário realizar a subtração deste nível de tensão DC.

Para o projeto do subtrator foi utilizado o amplificador de instrumentação AD620 (*datasheet* pode ser visualizado no Anexo C) da Analog Device, juntamente com o integrado LM10 (*datasheet* pode ser visualizado no Anexo B) para a geração do sinal de tensão DC a ser subtraído do sinal original do acelerômetro. A Figura 14 mostra o esquema elétrico do subtrator e fonte de referencia de tensão.

O potenciômetro 1 (POT 1) permitiu que fosse realizado um ajuste de ganho do amplificador de instrumentação, enquanto o potenciômetro 2 (POT 2) permitiu o ajuste fino na tensão de referência (nível DC) a ser subtraída do sinal original.

Outra medida tomada foi a implementação de um filtro passa-baixa de 2º ordem com frequência de corte em 2,5kHz com a finalidade de eliminar ruídos de alta frequência que são normalmente encontrados devido as fontes chaveadas de equipamentos eletrônicos.

Os dois pólos do filtro passa-baixa foram inseridos no mesmo local, a equação 3.4-1 foi utilizada para seu cálculo da frequência de corte. O circuito do filtro aplicado pode ser visualizado na Figura 15.

$R = 4.7k\Omega$

$C = 12nF$

$$f_{c-3dB} = \frac{1}{2\pi\sqrt{RC}}$$

eq. (3.4-1)

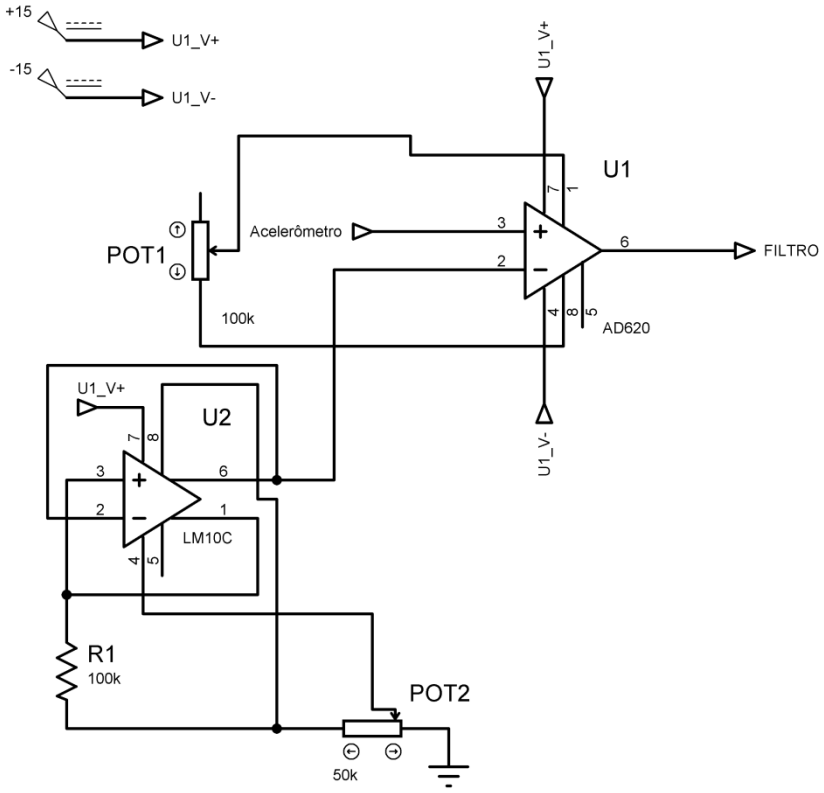


Figura 14 - Esquemático do circuito subtrator

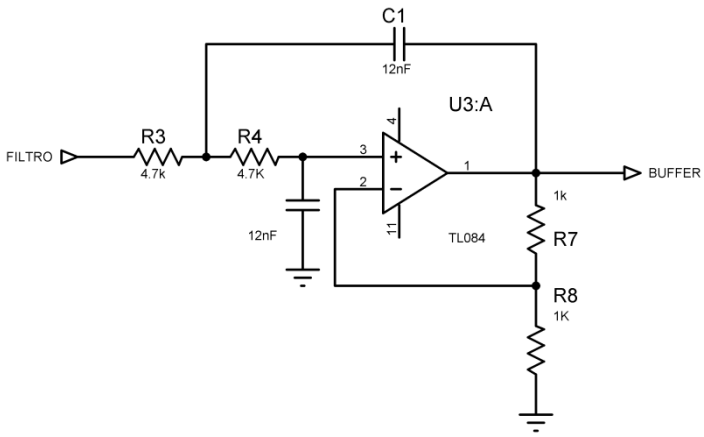


Figura 15 – Circuito Filtro Passa-Baixo de 2º ordem

A resposta simulada do filtro passa-baixas no domínio frequência pode ser observada na Figura 16.

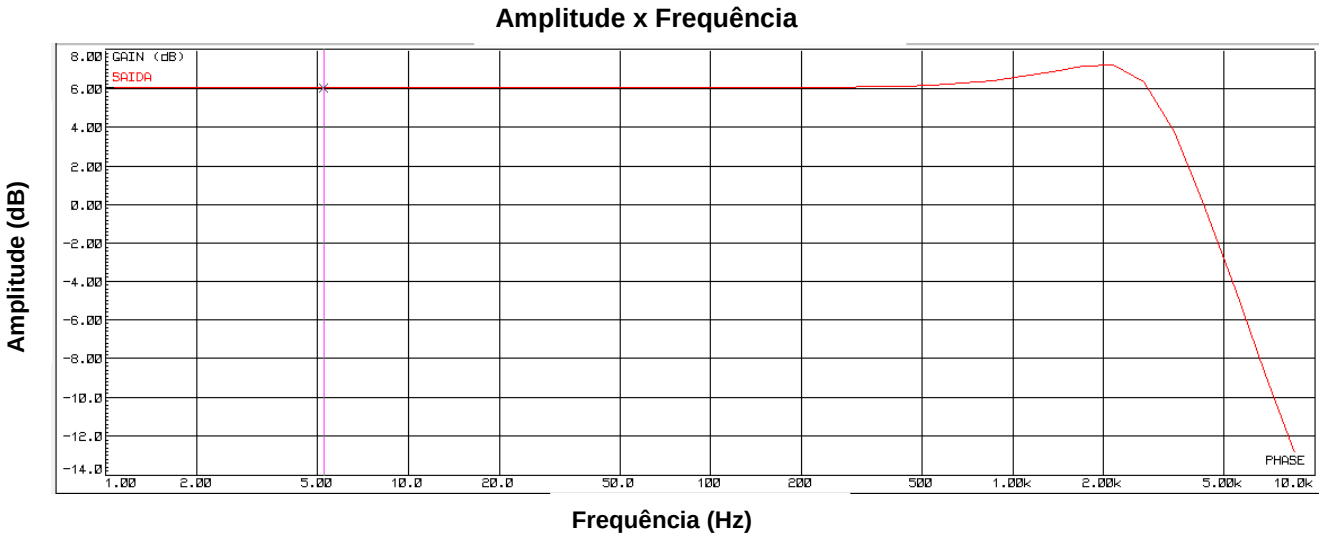


Figura 16- Gráfico da Amplitude de saída x Frequência

O ultimo estágio do condicionador de sinal é um b*uffer* utilizado para isolar o sinal proveniente do acelerômetro e assim proteger a placa de som. Juntamente com o Buffer foi implementado um divisor de tensão, com o objetivo de ajustar o nível de tensão da saída do condicionador de sinal e deixá-lo compatível com o nível de tensão da entrada da placa de som.

O circuito implementado "Divisor de Tensão + *Buffer*" pode ser observado na Figura 17.

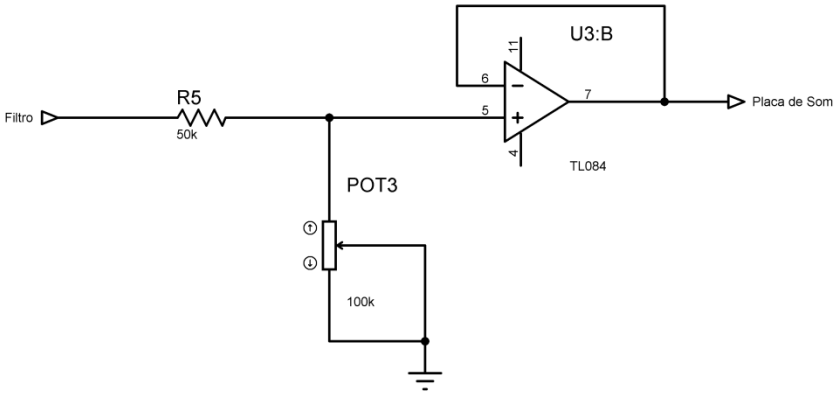


Figura 17 - Circuito Divisor de Tensão e *Buffer*

O nível de tensão da saída do *buffer* foi ajustado através do potenciômetro 3 (POT3).

Outra iniciativa tomada foi a construção de uma caixa em alumínio para o acondicionamento do circuito, assim, como a caixa é metálica e ligada à terra foi

obtido uma atenuação no nível de ruído externo. O circuito acondicionado na caixa pode ser observado na Figura 18.

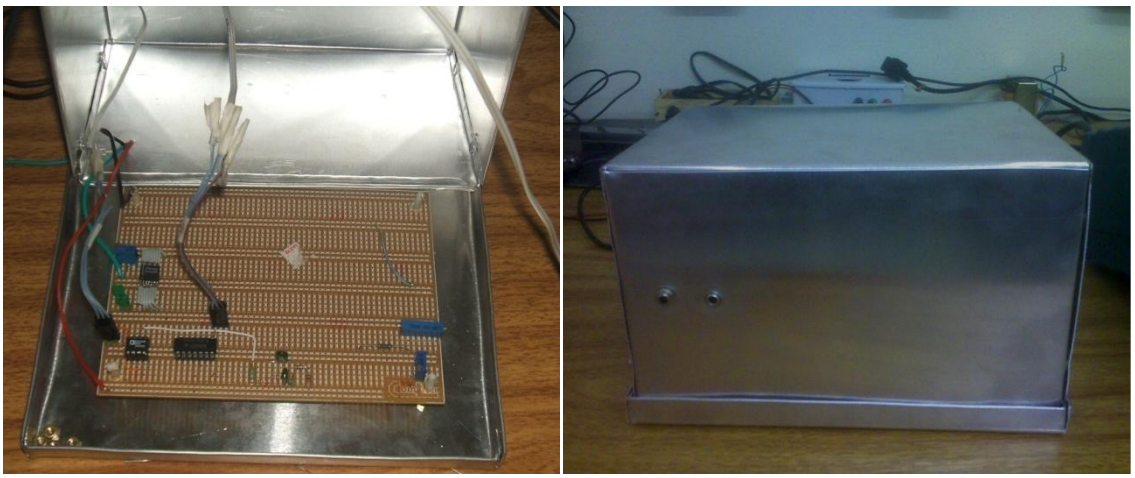


Figura 18 - Condicionador de Sinal

3.5. *A Mesa de Vibração*

A mesa de vibração tem o objetivo de gerar uma vibração mecânica para posteriormente ser adquirida pelo acelerômetro. A mesa é basicamente composta por um alto-falante, um módulo de amplificador automotivo e uma base metálica tubular onde o alto-falante e o módulo são fixados. Esse equipamento é utilizado pelo departamento de Engenharia Mecânica da ULBRA para ministrar aulas no laboratório sobre vibração. Anteriormente o Departamento de Engenharia Mecânica utiliza um gerador de frequência para excitação da mesa. A Figura 19 mostra a mesa de vibração com o acelerômetro já fixado.

Figura 19 - Mesa de Vibração

Foi levantada a curva de resposta em frequência do alto-falante, com relação à amplitude de vibração. Essa curva foi levantada utilizando-se um gerador de frequência com uma amplitude fixa de 400mV, e foi-se aumentado a frequência do sinal com um *step* de 10Hz. A amplitude de saída do acelerômetro foi medida através do osciloscópio e todas as medidas referem-se à amplitude pico-a-pico do sinal de tensão. O gráfico com os resultados deste ensaio podem ser observados da Figura 20.

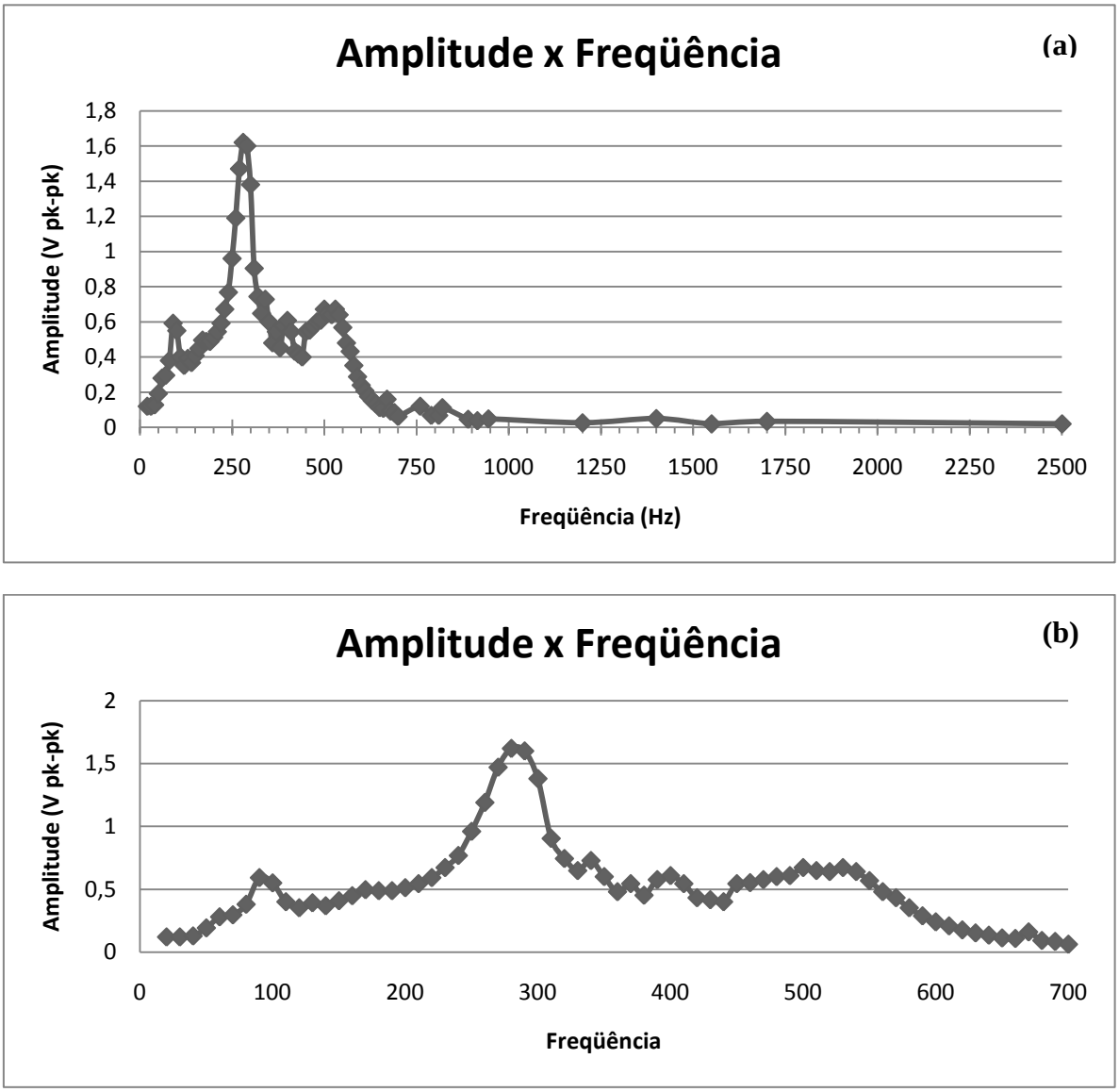


Figura 20 – (a)Gráfico da resposta da amplitude em relação à frequência (b) Gráfico da faixa de maior amplitude da Mesa de Vibração

3.6. *Protótipo de Um Equipamento Rotativo*

Para possibilitar a realização de simulações que reproduzam um defeito prático, foi desenvolvido um protótipo de uma máquina rotativa. Esse equipamento foi projetado para que se possa facilmente reproduzir uma falha de desbalanceamento ou a substituição de um rolamento bom por um rolamento defeituoso, assim possibilitando a visualização do defeito.

Uma foto do protótipo pode ser observada na Figura 21.

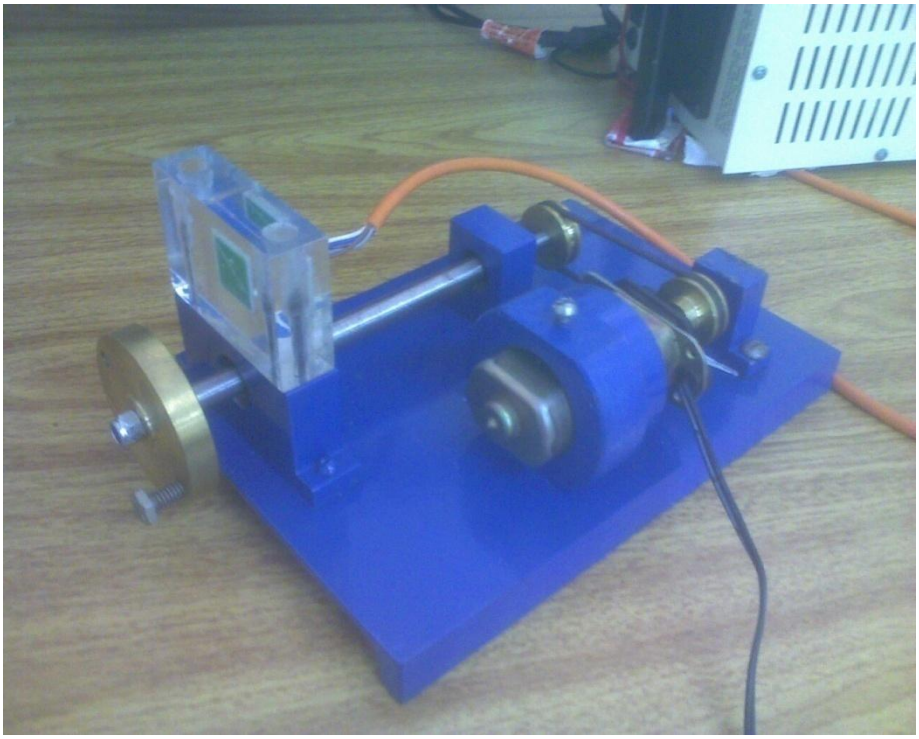


Figura 21 - Protótipo Máquina Rotativa

Os defeitos de máquinas rotativas quando observado através do espectro do sinal de vibração, estão relacionado com a rotação em que o equipamento trabalha. A relação entre alguns tipos de defeitos e onde o mesmo é esperado no espectro de vibração pode ser observado na Tabela 3.

Tabela 3 - Relação entre variáveis e causa de vibração [3]

CAUSA	FREQUÊNCIA	OBSERVAÇÃO
Desbalanceamento	1 X rpm	Geralmente aparece no primeiro harmônico
Desalinhamento ou eixo torto	1/2/3/4 X rpm	Desalinhamento entre mancai
Folgas nos mancais	$^1/_2$ X rpm	Como a frequência é menor que $^1/_2$ rpm a fase pode variar
Falta de firmeza mecânica	2 X rpm	Geralmente afeta o alinhamento
Engrenagens defeituosas	Alta, número de dentes X rpm	Frequência entre 15000 e 40000 rpm
Rolamentos deteriorados	Alta	Frequência entre 15000 e 40000 rpm

Foi utilizado um motor de corrente continua para a construção do protótipo. A relação entre a tensão aplicada ao motor e sua rotação foi levantada utilizando-se um tacômetro, os dados levantados estão relacionados na Tabela 4.

Tabela 4 - Relação entre tensão aplicada ao motor e sua rotação

Tensão(V)	Rotação (rpm)	Rotação (Hz)
6	3224	53,73
10	6423	107,05
12	8000	133,33

3.7. Descrição do Software

O software foi desenvolvido através do MATLAB utilizando-se a ferramenta GUI (*Graphical User Interface).* Com isso foi obtido um software com uma interface amigável com o usuário e de fácil operação.

Os principais objetivos do software são:

- Gerar um sinal através da placa de som onde o usuário possa escolher os valores de frequência, amplitude e o tipo de forma de onda desejada.
- Realizar a aquisição através da placa de som do sinal de vibração e apresentá-lo na tela do computador juntamente com a FFT do sinal. Isso permitirá ao usuário a visualização da frequência onde está ocorrendo o defeito e seu nível de severidade.

3.7.1. Data Acquisition Toolbox - MATLAB.

O MATLAB possui uma ferramenta chamada de *Data Acquisition Toolbox* que permite a aquisição de dados analógicos, o envio de dados analógicos e I/O digital de uma variedade de hardware de aquisição de dados compatíveis com a plataforma PC. O *Data Acquisition Toolbox* permite configurar seus dispositivos de hardware externos, enviar esses dados para o ambiente MATLAB ou o ambiente *Simulink*, para análise imediata dos dados adquiridos e posterior envio. A Figura 22 ilustra o funcionamento do *Data Acquisition Toolbox*.

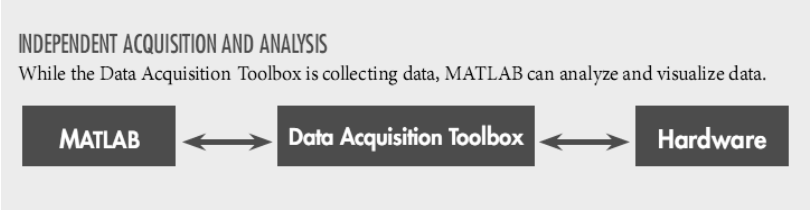


Figura 22 - Funcionamento da aquisição de dados no MATLAB
MathWorks (2009)

Da mesma forma que o Data Acquisition Toolbox pode ser configurado para placas de aquisição de diversos fabricantes compatível com computadores da plataforma PC, o mesmo também pode ser configurado para adquirir os dados de interesse através da placa de som do computador PC.

Com isso ele utiliza a placa de som como placa de aquisição, podendo ser configurados dados como a frequência de amostragem da placa (obedecendo aos limites da placa de som utilizada).

3.7.2. Funções Utilizadas.

O MATLAB como originalmente foi um software destinado ao cálculo matricial possui diversas funções que facilitam a programação. Abaixo serão mostradas as principais funções utilizadas no desenvolvimento deste projeto. Essas funções serão classificadas pelo tipo de funcionalidade implementada.

Funções do Data Acquisition Toolbox

As principais funções para se iniciar a aquisição de dados pela placa de som são:

- Criar um objeto de entrada analógica para a placa de som.

- *ai=analoginput ('winsound');* Cria "ai" como um objeto, sendo uma entrada analógica e inclui como *hardware* a placa de som (winsound);
- *addchannel (ai,1:2);* Adicionar o número de canais desejados por *hardware* (no caso a placa de som). Nesse caso foram criados dois canais para o objeto de entrada analógica "ai".
- *set (ai,'SampleRate',44100);* Configurar a taxa de amostragem (*Sampling Rate*)
- *set (ai,'SamplesPerTrigger',44100);* configurar quantas amostras serão coletadas em 1 segundo (*Samples Per Trigger*).
- *start (ai);* Iniciar a aquisição dos dados.
- *data=getdata (ai)*; Armazena em "data" o vetor de pontos adquiridos.

- Criar um objeto de saída analógica para a placa de som

- *ao=analogoutput ('winsound');* Cria "ao" como um objeto, sendo uma saída analógica e inclui como hardware a placa de som (*winsound*);

- *addchannel (ao,1:2);* Adicionar o número de canais desejados por *hardware* (no caso a placa de som). Nesse caso foram criados dois canais para o objeto de saída analógica "ao".
- *set (ao,'SampleRate',44100);* Configurar a taxa de amostragem (*Sampling Rate*).
- *set (ao, 'RepeatOutput', inf);* Configura o número de vezes que o sinal deve ser repetido, neste caso foi configurado para se repetir o sinal infinitas vezes.
- *data = sin(linspace(0,2*pi*500,44100)');* Criou-se um vetor "data" de 1 segundo com os pontos a serem enviados a placa de som.
- *putdata (ao,[data data]);* Envia-se o arquivo para a placa de som.
- *start (ao);* Inicia-se a saída de sinal.

Funções Matemáticas

Algumas instruções matemáticas já se encontram prontas para o uso no MATLAB, as principais funções usadas no projeto serão descritas abaixo:

- *Y=fft(X)*; Essa função retorna em "Y" um vetor com a Transformada Discreta de Fourier do vetor "X".
- *Z=cumtrapz(Y)*; Essa função retorna em "Z" um vetor com o cálculo da aproximação da integral cumulativa do vetor "Y" pelo método trapezoidal.
- *x=sawtooth(t, width)*; Essa função gera uma onda dente de serra ou triangular dependendo da inclinação escolhida (nesse caso se escolhermos *width* de 0.5 é gerada uma onda triangular) com período de 2π com os elementos do vetor t, similar a uma onda senoidal com um pico de onda de ±1.
- *X=square(t,duty)*; Essa função gera uma onda quadrada com período de 2π com os elementos do vetor t, similar a uma onda senoidal com um pico de onda de ±1. O *duty cicle* pode ser configurado de 0 a 100 que é equivalente ao percentual do período em que o sinal se encontra positivo.

Ambiente de Construção das Telas de Interação

A construção das imagens ou ambiente de trabalho pode ser realizada de duas maneiras. A primeira maneira é a construção do ambiente dentro do arquivo ".m" de forma descritiva junto à programação normal.

Outra forma possível é programação orientada a objetos. Semelhante a programação *C++ Builder*, essa programação é realizada através da ferramenta

GUIDE. O ambiente de construção das telas gráficas através do GUIDE pode ser observado na Figura 23.

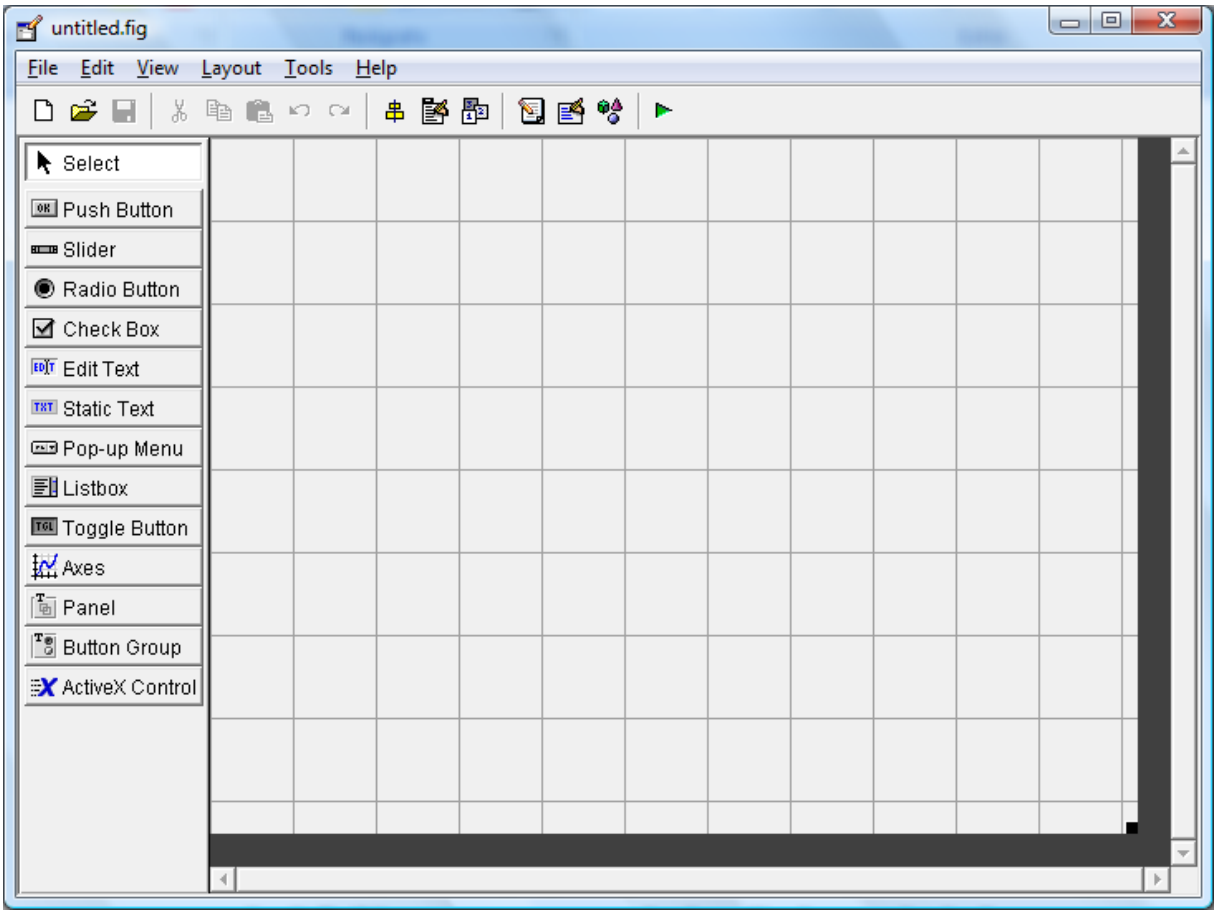


Figura 23 - Ambiente de trabalho do GUIDE.

3.7.3.Telas do Software

O Software desenvolvido apresenta quatro telas básicas que serão apresentadas a seguir:

- Tela Inicial

Ao se executar o programa a tela inicial do software apresentada na Figura 24 surgirá, nela estão disponíveis os botões a serem selecionados conforme a opção desejada. Cada botão "chama" um novo programa que é executado independentemente do outro, realizando todos os passos de configuração da placa de som individualmente. O código fonte da Tela Inicial encontra-se no Apêndice D.

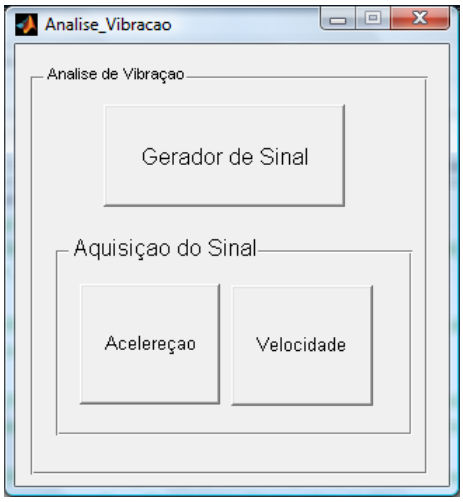


Figura 24 - Tela Inicial

- Geração do Sinal

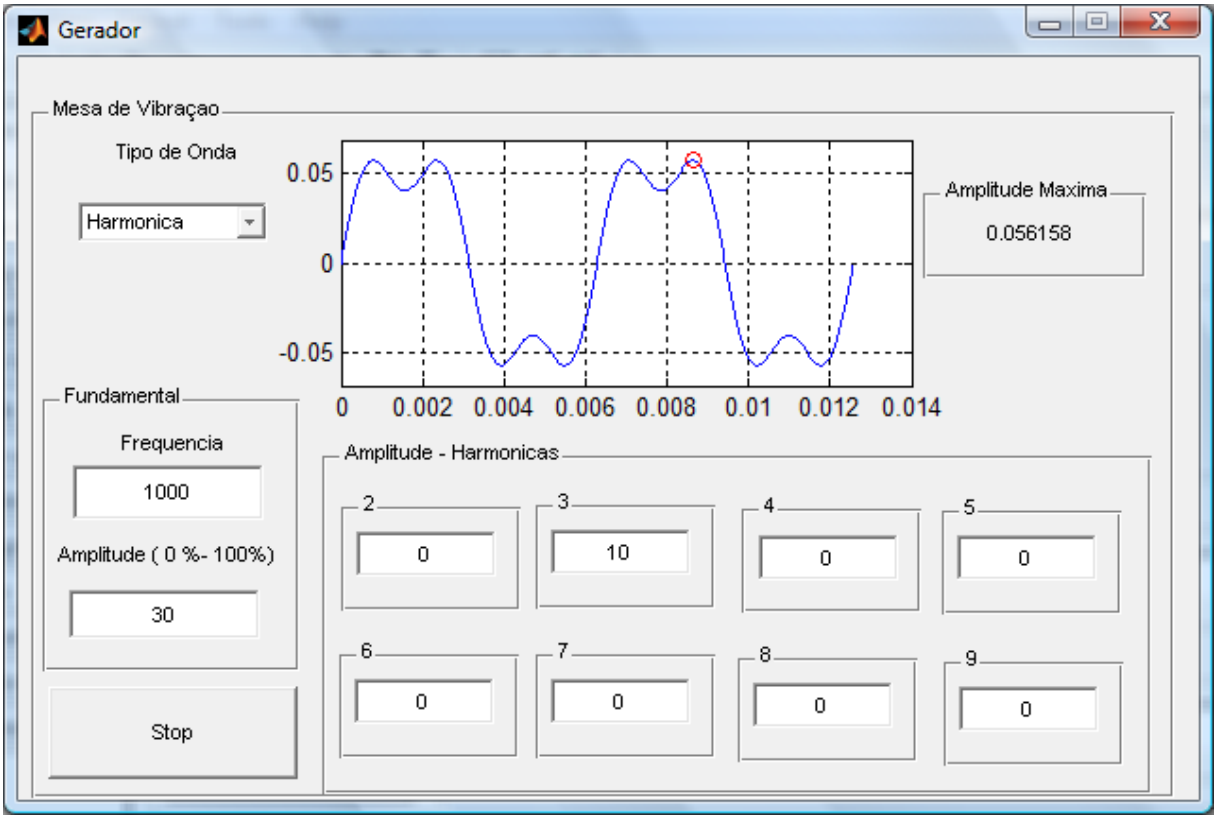


Figura 25 - Gerador de Sinal

A Figura 25 apresenta a tela do gerador de sinal. Nela é necessário escolher o tipo de onda a ser gerado (senoidal, harmônicas, quadrada e triangular), digitar um valor para frequência e amplitude. Se a forma de onda harmônica foi selecionada se faz necessário também digitar a amplitude desejada em cada múltiplo da frequência fundamental. O código fonte do Gerador encontra-se no Apêndice A.

- Sinal de Aceleração

A tela do sinal de aceleração apresenta o sinal no domínio tempo na janela superior e o espectro de frequência na janela inferior conforme mostra a Figura 26. O código fonte da tela de aquisição do sinal de aceleração encontra-se no Apêndice B.

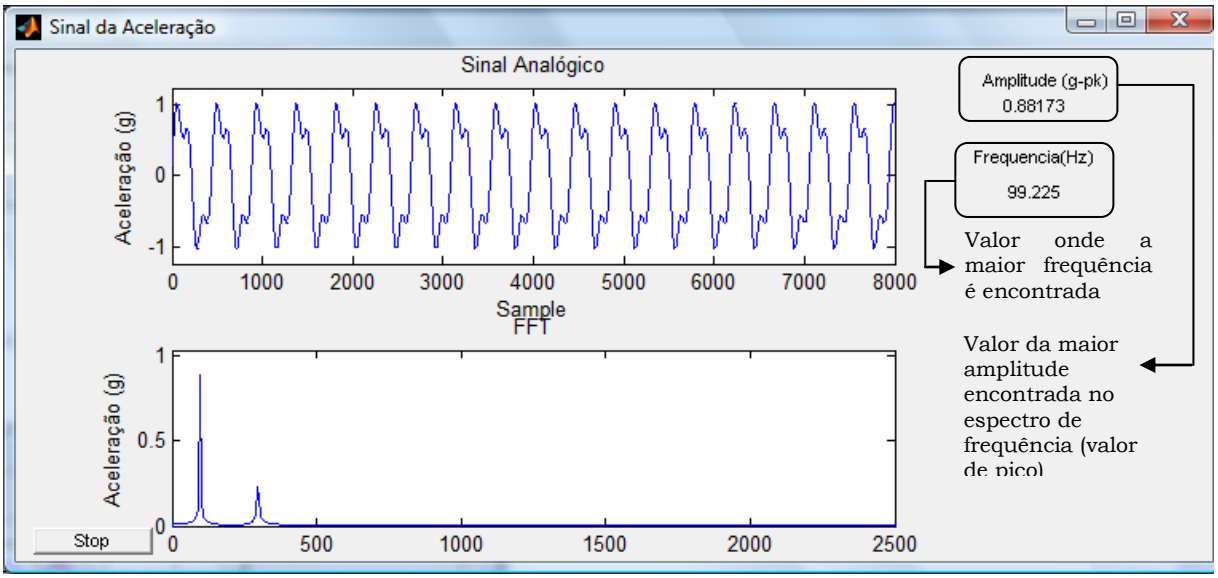


Figura 26 - Tela de Aquisição do Sinal de Aceleração

- Sinal de Velocidade

A tela do sinal de velocidade apresenta a mesma configuração da tela do sinal de aceleração e pode ser observada na Figura 27. O código fonte da tela de aquisição do sinal de velocidade encontra-se no Apêndice C.

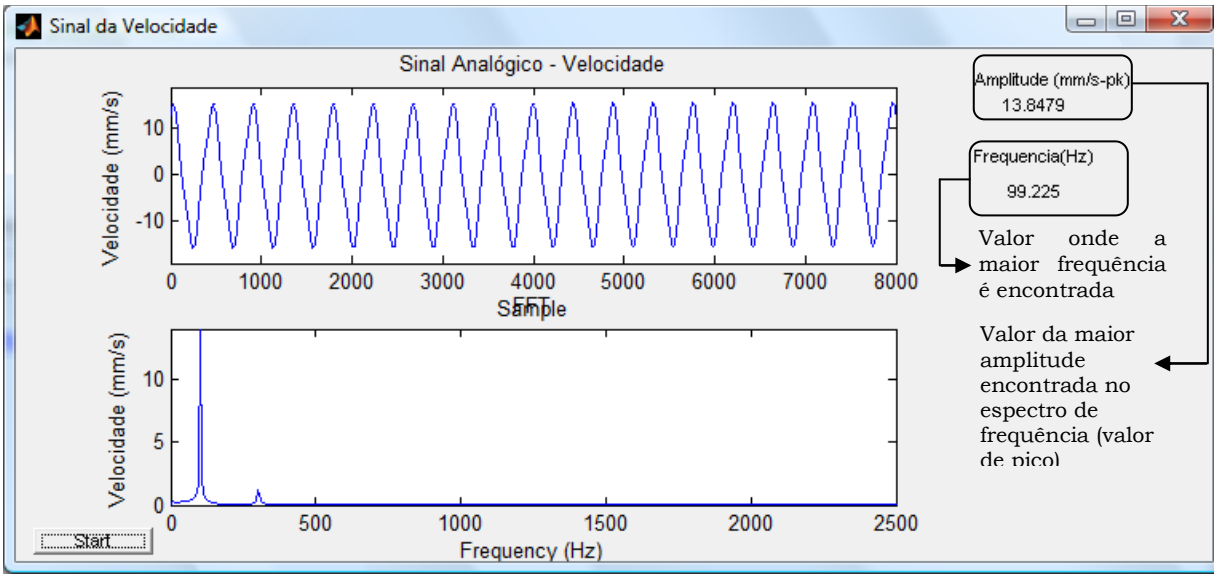


Figura 27 - Tela de Aquisição do Sinal de Aceleração

3.7.4. Fluxograma Simplificado do Software

Como mensionado anteriormente, cada programa é executado individualmente do outro, cada um possuindo seu próprio código fonte. O fluxograma do *software* de aquisição do sinal de aceleração é apresentado da Figura 28.

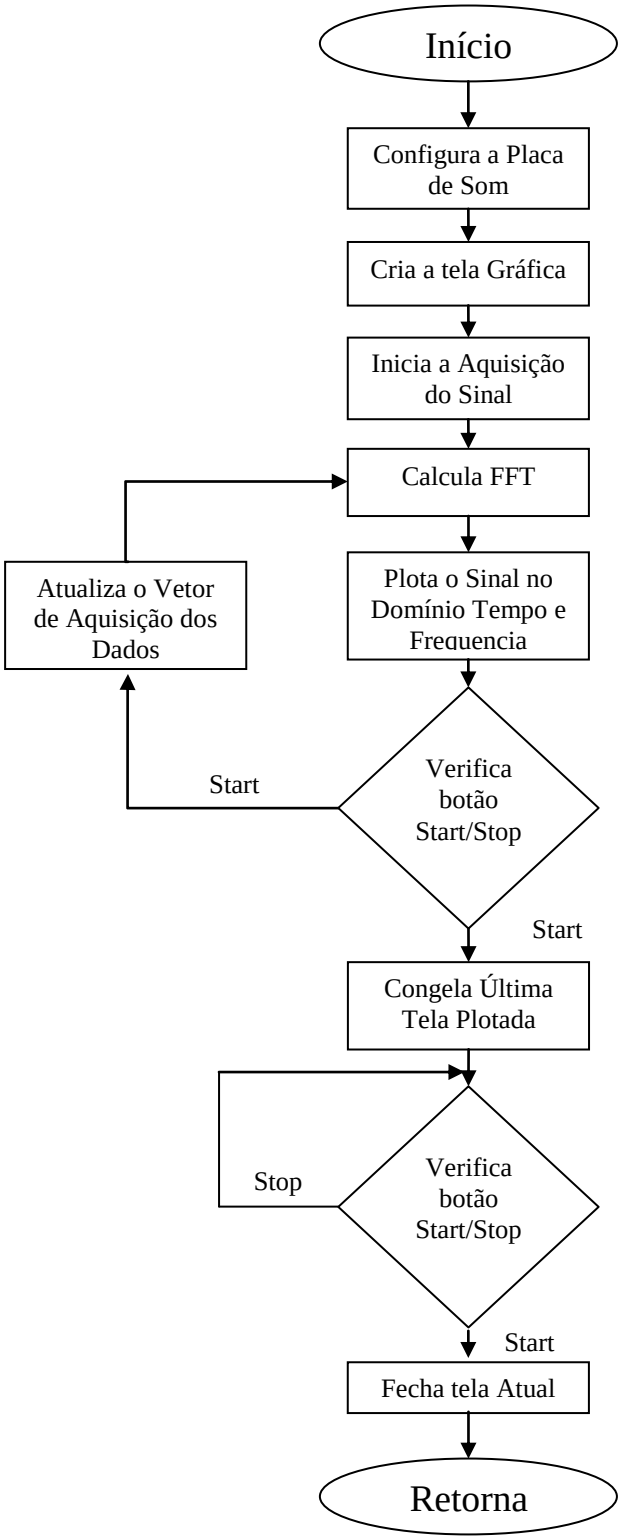


Figura 28 - Fluxograma do Software de Aquisição do Sinal da Aceleração

O fluxograma do *software* de aquisição do sinal de velocidade pode ser observado na Figura 29.

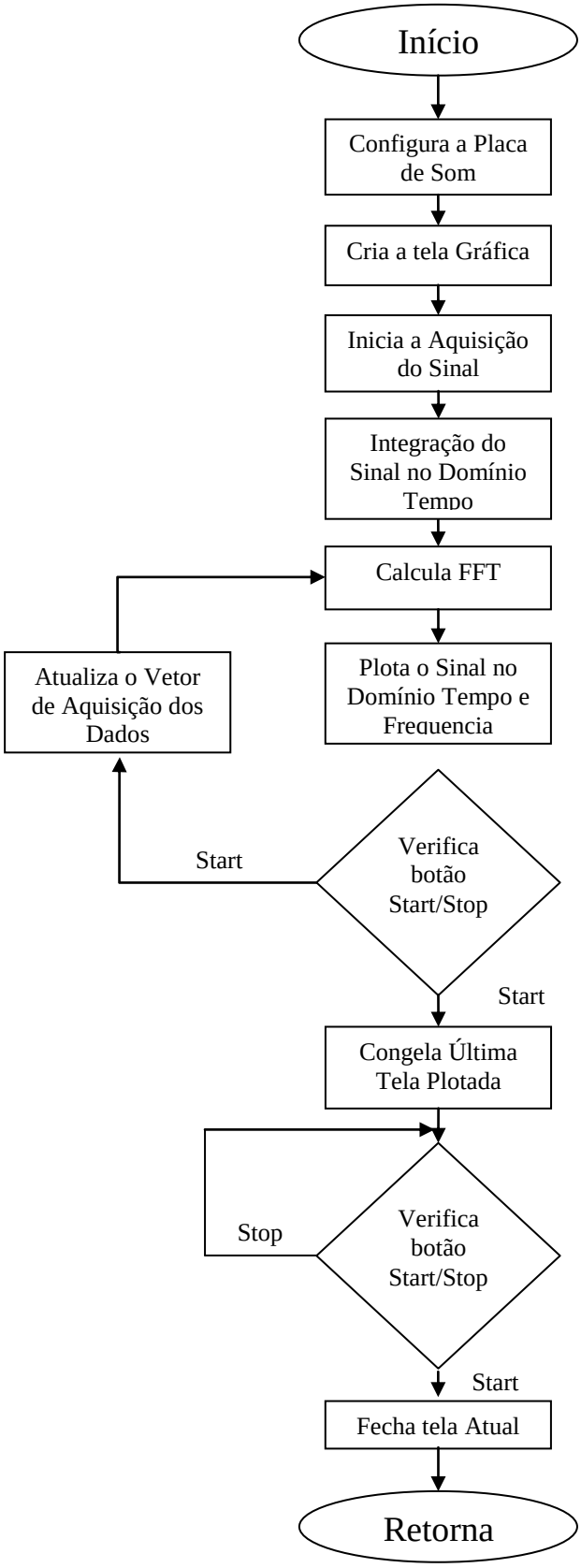


Figura 29 - Fluxograma do Software de Aquisição do Sinal de Velocidade

No momento em que a tela inicial é executada, existirá a possibilidade de se executar três programas distintos. Em todos os programas existirão as seguintes etapas básicas:

- configuração da placa de som;
- inicialização da tela gráfica;
- cálculos matemáticos;
- atualização da tela gráfica com os novos resultados.

4. RESULTADOS EXPERIMENTAIS

4.1. Funcionamento do Equipamento

Para analisar o comportamento do equipamento como um todo foram realizadas as aquisições dos sinais em alguns pontos conforme a 30 ilustra.

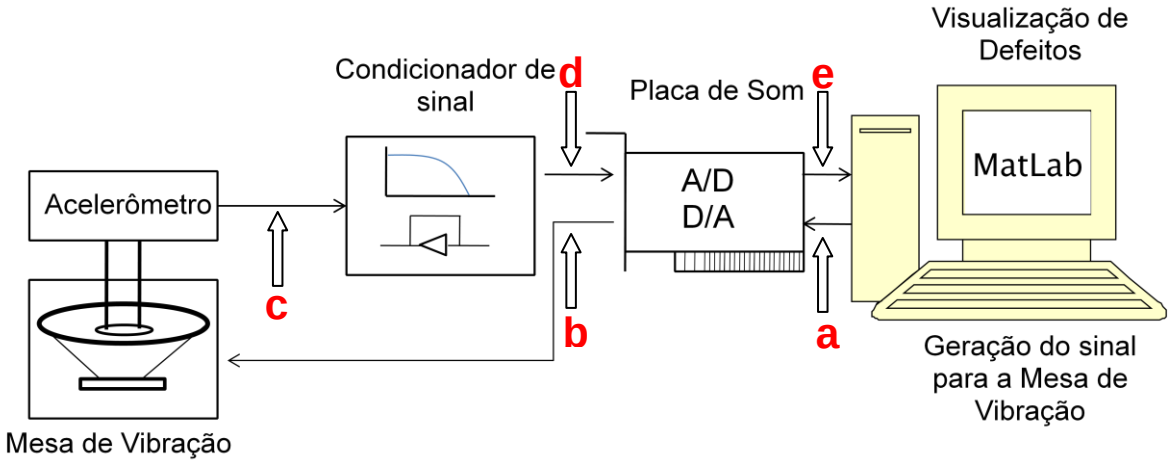


Figura 30 – Diagrama em blocos dos pontos de Aquisição do Sinal

O ponto “a” da Figura 30 faz referência à escolha do sinal a ser gerado para a placa de som, através do software desenvolvido no MATLAB. O ponto “b” é o sinal de saída da placa de som, que vai estimular a mesa de vibração. A saída do acelerômetro é referenciada pelo ponto “c”. O sinal após passar pelo condicionador de sinais é adquirido através do ponto “d”. E finalmente o ponto “e” é o sinal na tela de visualização do sinal de aceleração e velocidade de vibração.

Nas Figuras 31 e 32 a seguir pode-se ver o sinal nos pontos “a” e “b” do diagrama. Trata-se do sinal criado a partir de uma forma de onda harmônica com a frequência fundamental em 200Hz, amplitude de 70% e sua 3º harmônica com amplitude de 50%. Conforme visto no capítulo 3, a placa de som realiza uma atenuação em sua saída em frequencias abaixo de 280 Hz. Isso fica claro ao comparmos o sinal gerado (Figura 31) e o sinal medido (Figura 32) no osciloscópio.

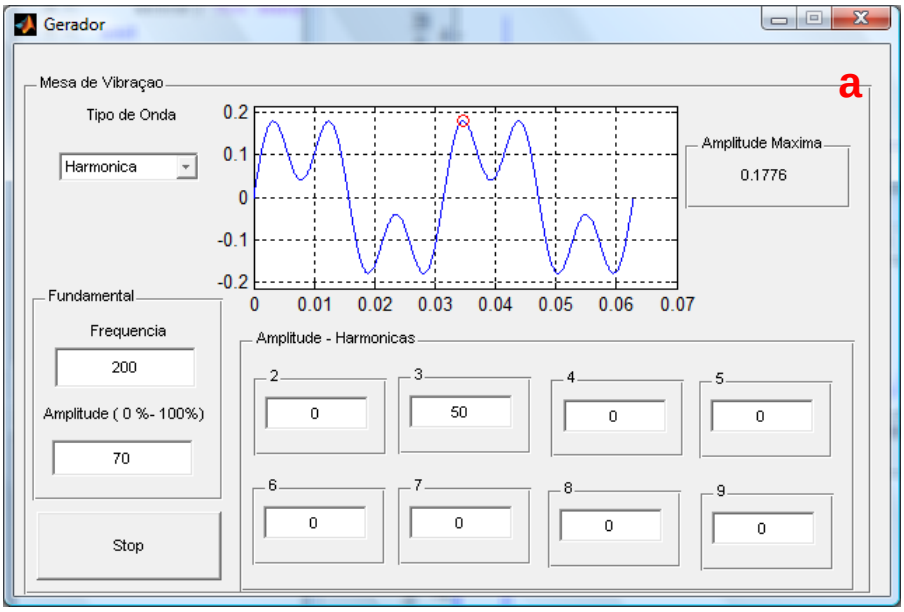


Figura 31 - Posição "a" do diagrama em blocos - Tela do MATLAB

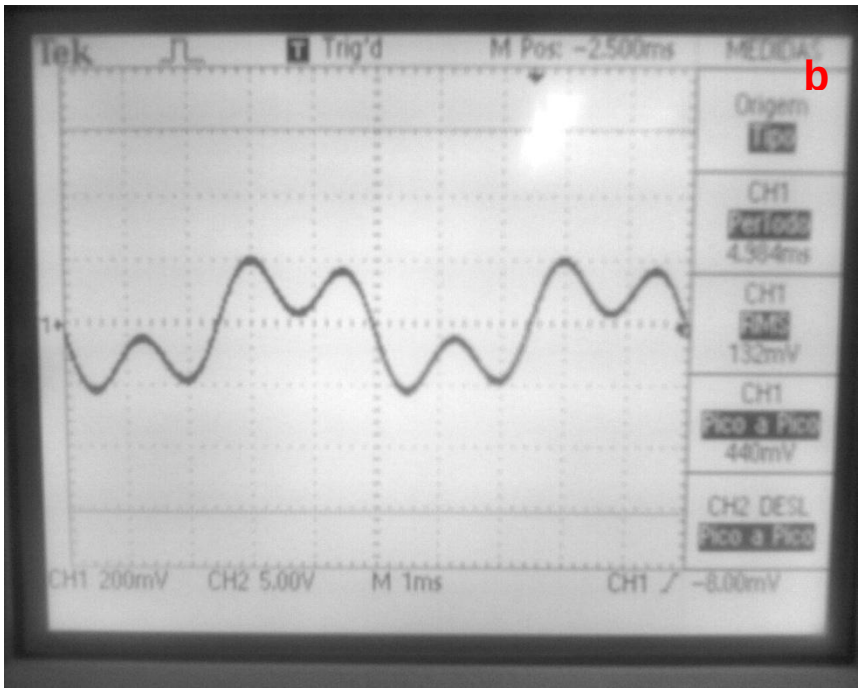


Figura 32 - Posição "b" do diagrama em blocos - Saída da placa de som

As Figuras 33 e 34 mostram as tensões nos pontos “c” (saída do acelerômetro) e “d” (saída do condicionador de sinais) referente ao diagrama da Figura 29.

Figura 33 - Posição "c" do diagrama em blocos - Saída acelerômetro

Figura 34 - Posição "d" do diagrama em blocos – Saída do Condicionador/Entrada Placa de Som

O último pondo do diagrama, o ponto “e” representa o sinal já processado pelo MATLAB, e pode ser visualizado através do sinal de aceleração e velocidade da vibração através das Figuras 35 e 36.

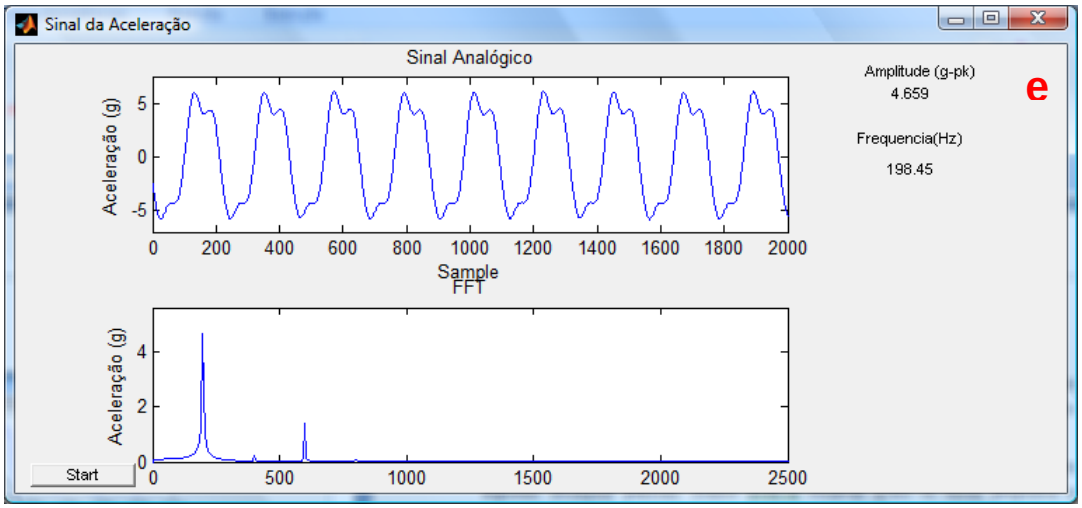


Figura 35 - Posição "e" do diagrama em blocos - Tela do sinal da aceleração.

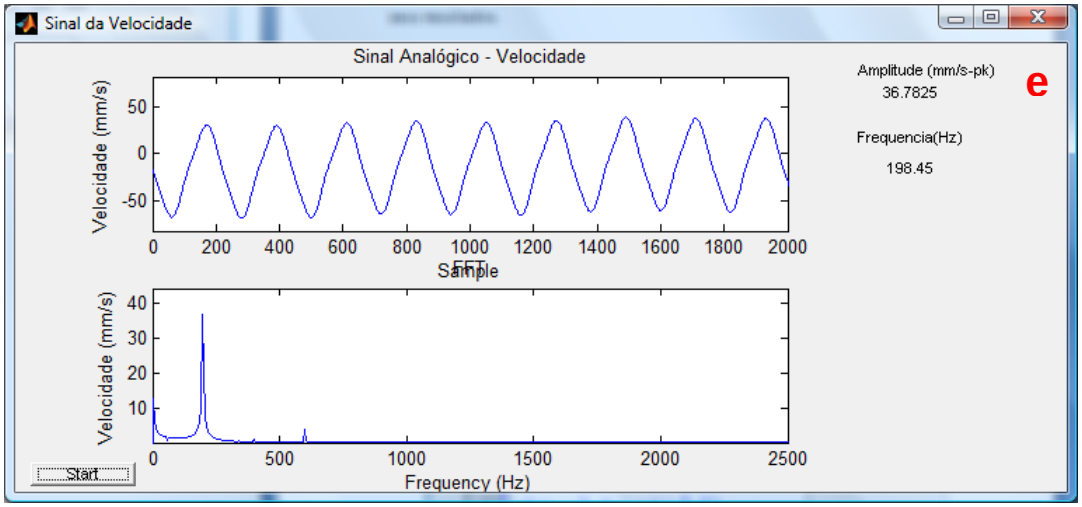


Figura 36 - Posição "e" do diagrama em blocos - Tela do sinal de velocidade

Através dessa sequência de imagens foi possível observar como o sinal se comporta em cada etapa do equipamento. Foi possível observar também que a Mesa de Vibração não responde linearmente em relação ao sinal elétrico enviado pela placa de som e o sinal vibratório gerado pela mesa conforme ilustrado nas Figuras 32 e 33 (pontos “b” e “c”).

Outro ponto observado foi o ruído enquanto o acelerômetro está em repouso (sem vibração). Acredita-se que esse ruído é proveniente das conexões expostas do acelerômetro. Esse problema foi resolvido subtraindo-se o nível do ruído no espectro da frequência, assim foi possível detectar menores níveis de amplitude de aceleração e velocidade. O sinal no domínio tempo foi preservado, assim foi possível visualizar a amplitude do ruído sem que o mesmo afetasse a visualização do sinal do domínio frequência.

4.2. *Teste do Protótipo*

Conforme descrito anteriormente no capitulo três, foi desenvolvido um protótipo de um equipamento rotativo para reproduzir alguns defeitos que são encontrados normalmente na prática. Testes foram realizados com o objetivo de validar o sistema para algumas situações práticas. Abaixo se pode observar quais os testes propostos e seus resultados.

A correlação entre o sinal de velocidade e o sinal de aceleração pode ser observada na Figura 37. Conforme foi mencionado no capítulo 2, a velocidade é a integral da aceleração, mas também pode ser obtida através da equação presente na Figura 37.

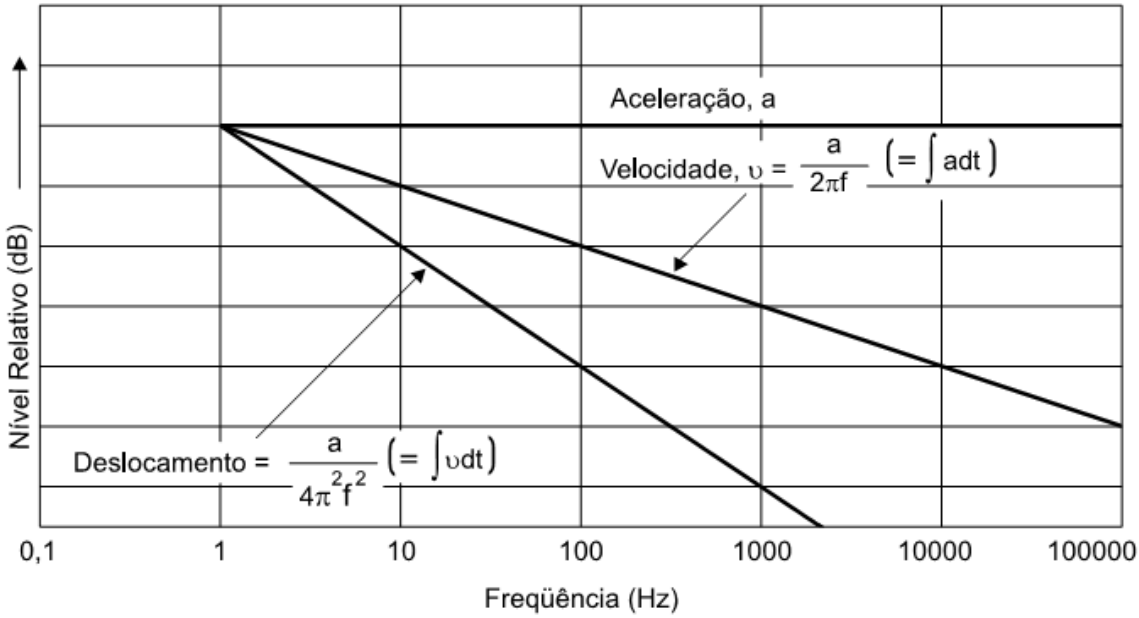


Figura 37 - Relação entre aceleração e velocidade de vibração (6)

4.2.1. Assinatura do Equipamento

Inicialmente realizou-se a aquisição da assinatura do equipamento (sinal característico do equipamento em condições ditas como ideais de funcionamento), nas três frequências que foram testadas. Conforme capitulo três foram realizados testes alimentando o Motor DC do Equipamento com 6V (frequência de rotação de 50Hz), 10V (frequência de rotação de 50Hz) e 12V (frequência de rotação de 50Hz). Foram adquiridos os sinais de aceleração e velocidade para cada tensão de alimentação. Os resultados podem ser observados nas Figuras 38 a 43.

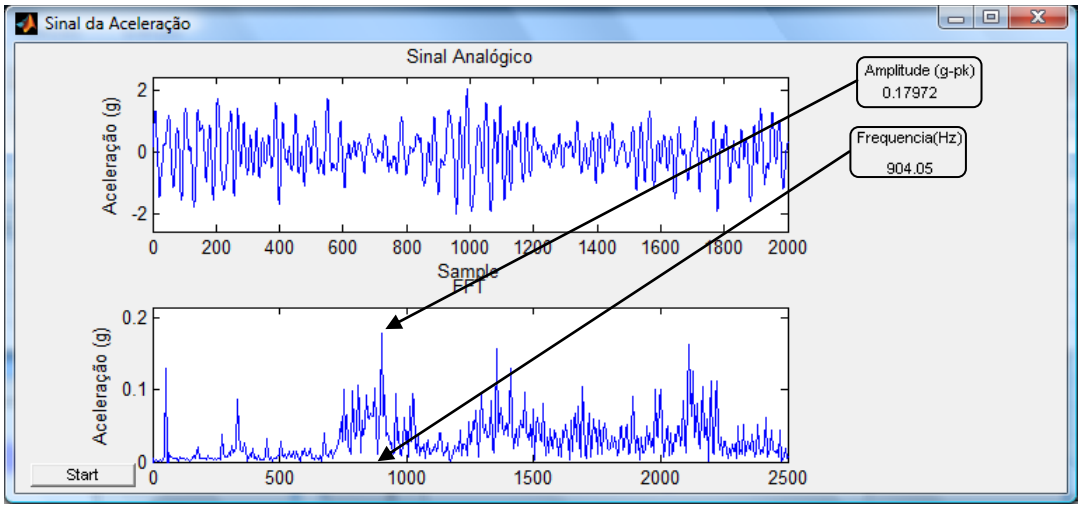


Figura 38 - Assinatura do sinal de aceleração do equipamento alimentado com 6Vdc ≅53Hz.

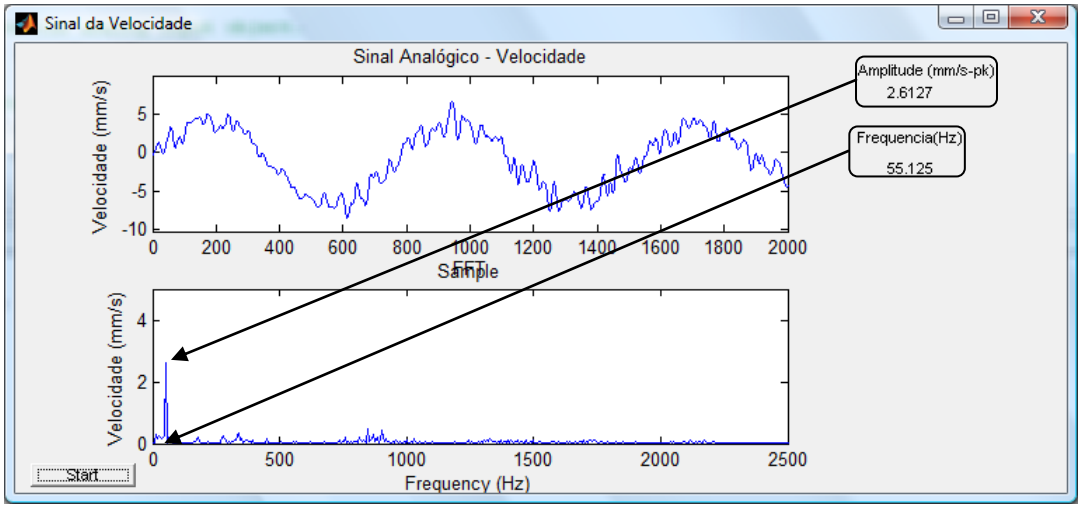


Figura 39 - Assinatura do sinal de velocidade do equipamento alimentado com 6Vdc ≅53Hz.

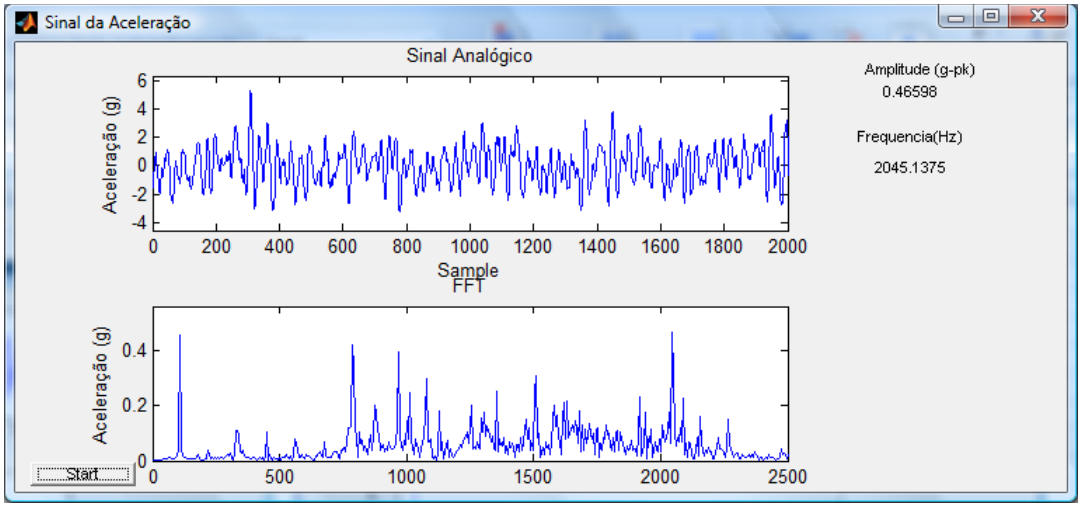


Figura 40 - Assinatura do sinal de aceleração do equipamento alimentado com 10Vdc ≅107Hz.

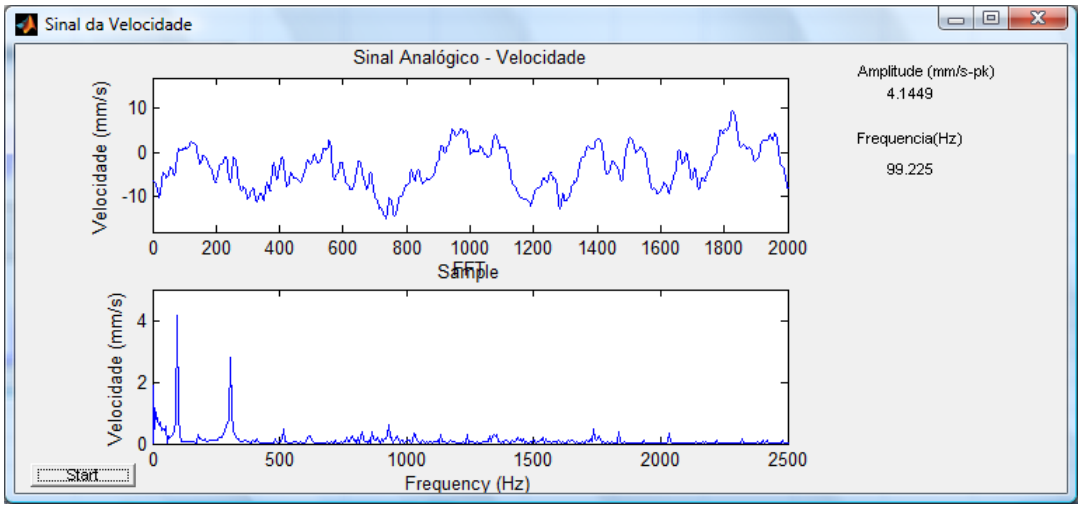


Figura 41 - Assinatura do sinal de velocidade do equipamento alimentado com 10Vdc ≅107Hz.

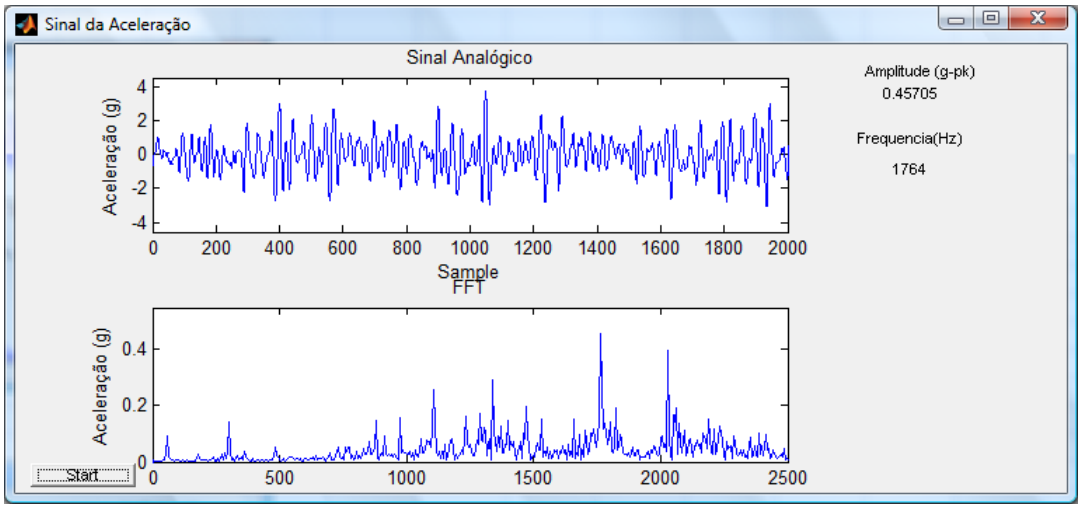


Figura 42 - Assinatura do sinal de aceleração do equipamento alimentado com 12Vdc ≅130Hz.

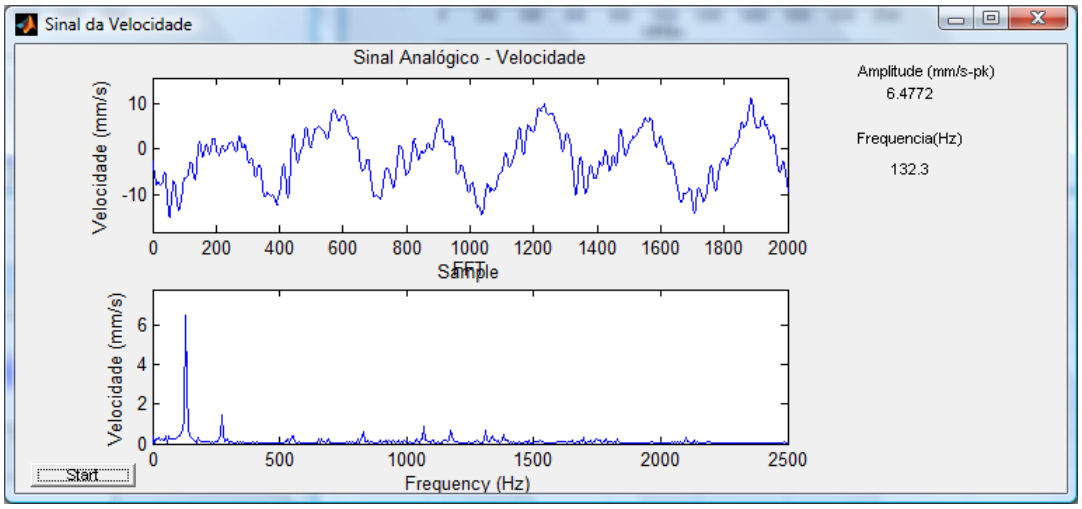


Figura 43 - Assinatura do sinal de velocidade do equipamento alimentado com 12Vdc ≅130Hz.

Estes espectros de frequência foram considerados como os espectros de operação do equipamento em condições normais (livre de defeito), e as vibrações encontradas foram consideradas como um fator intrínseco ao equipamento. Todos os outros espectros de frequência adquiridos foram comparados com estes em suas respectivas frequências de operação, assim foi possível a visualização do defeito gerado.

Para facilitar a comparação entre os parâmetros foi criada a Tabela 5, que sumariza os níveis máximos de amplitude encontrados para cada tensão de alimentação do motor.

Tabela 5 - Assinatura do equipamento - Amplitudes máximas encontradas

Tensão de Alimentação	Aceleração(g)	Velocidade(mm/s)
6V	0,179	2,61
10V	0,465	4,15
12V	0,457	6,48

4.2.2.Defeito de desbalanceamento.

Um dos defeitos comumente encontrados em indústrias é o desbalanceamento da parte girante do equipamento rotativo (hélices de ventiladores, rotores de bombas, etc.). Este defeito será reproduzido no protótipo com o objetivo de verificar se o sistema proposto é capaz de fornecer a visualização do defeito. O equipamento foi desbalanceado com duas massas diferentes (5 gramas e 10 gramas). Assim foi possível verificar se o equipamento alem de possibilitar a visualização do defeito também é capaz de fornecer a informação sobre a severidade da vibração.

O resultado esperado conforme mencionado anteriormente no capítulo 3, é que aparece no espectro de frequência uma amplitude que se sobressaia das demais na mesma frequência de trabalho do equipamento.

As Figuras 44 e 45 os resultados obtidos para o motor alimentado com 10V e desbalanceado com 5 gramas. As Figuras 46 e 47 mostram os resultados para o motor alimentado com 10V e desbalanceado com 10 gramas.

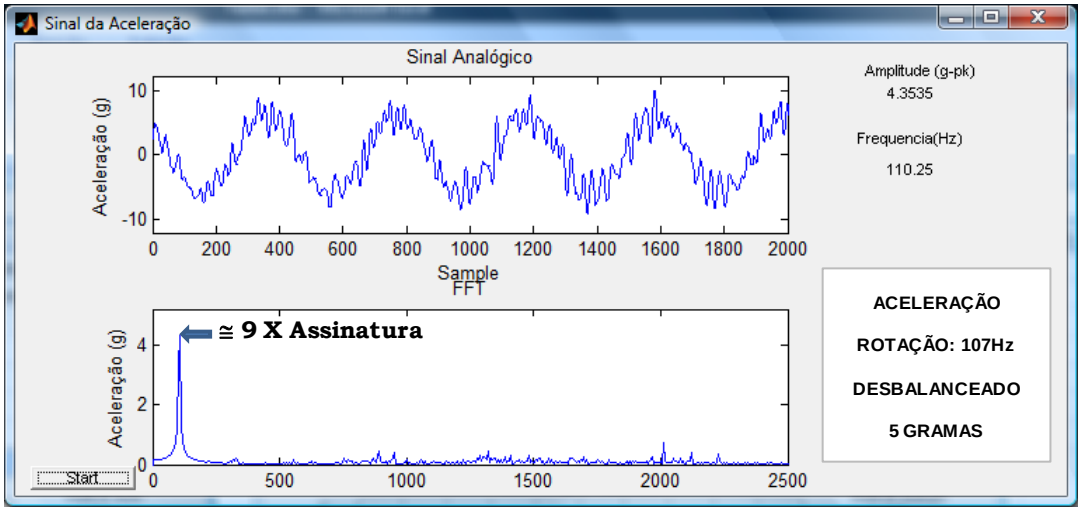


Figura 44 - Sinal de aceleração do equipamento alimentado com 10Vdc (≅107Hz) desbalanceado com 5 gramas.

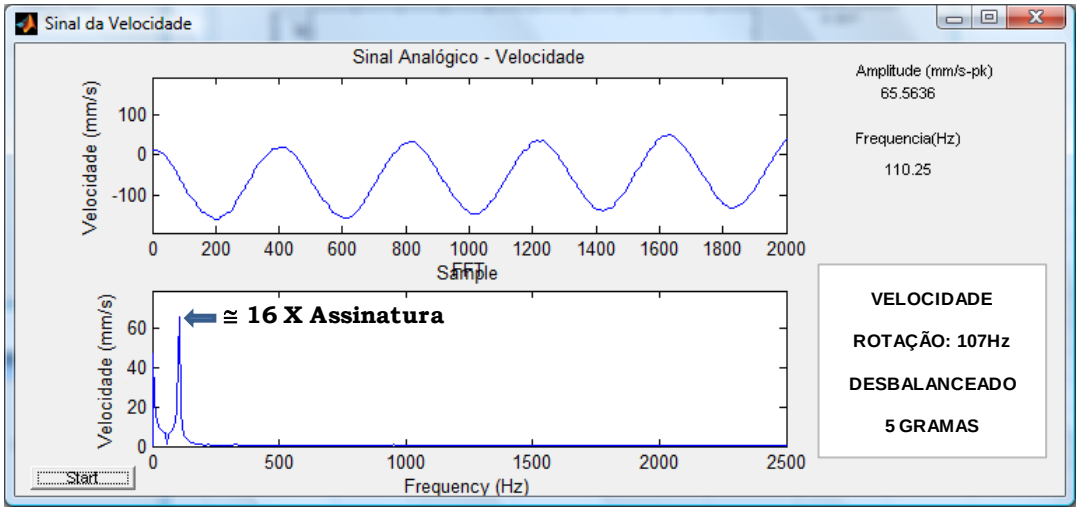


Figura 45 - Sinal de velocidade do equipamento alimentado com 10Vdc (≅107Hz) desbalanceado com 5 gramas.

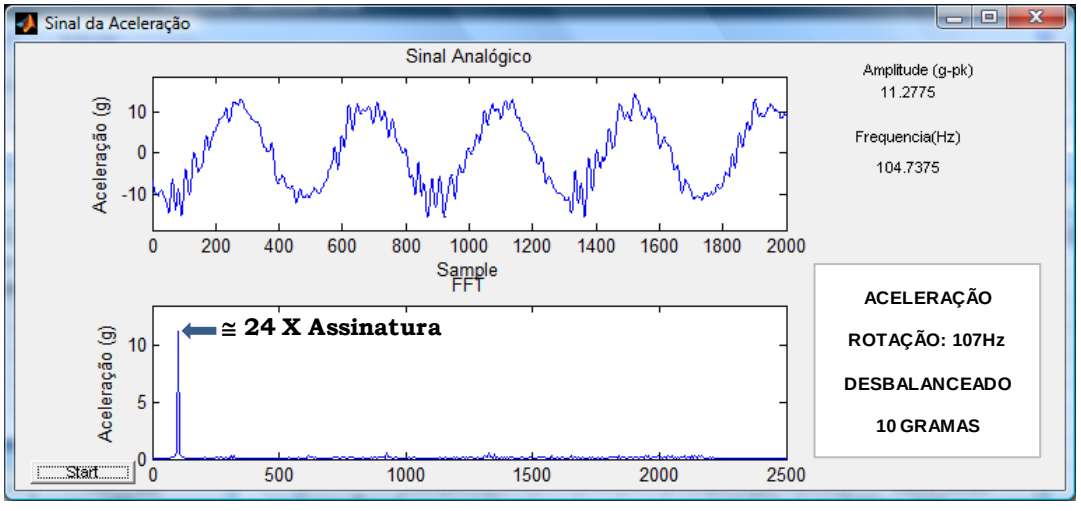


Figura 46 - Sinal de aceleração do equipamento alimentado com 10Vdc (≅107Hz) desbalanceado com 10 gramas.

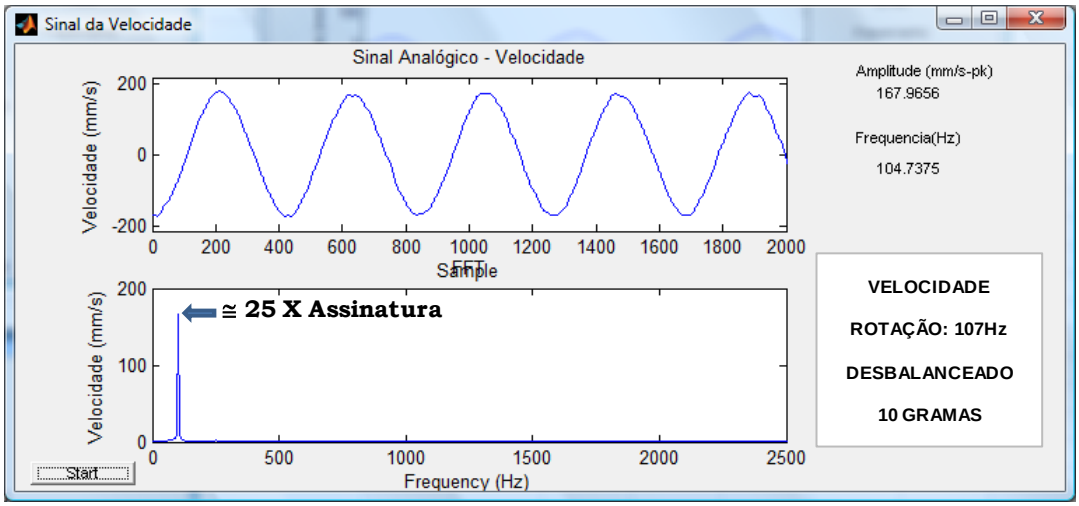


Figura 47 - Sinal de velocidade do equipamento alimentado com 10Vdc (≅107Hz) desbalanceado com 10 gramas.

A análise dos testes realizados mostra que o conjunto hardware/software, foi capaz de fornecer a visualização do defeito gerado conforme o esperado. A amplitude do sinal respondeu conforme o esperado, com alteração em sua amplitude no momento em que variava-se a massa do desbalanceamento. A frequência em que a maior amplitude foi detectada correspondeu à frequência em que o equipamento estava operando no momento.

Os demais resultados para as tensões de alimentação de 6V e 12V podem ser observados na Tabela 6 e Tabela 7.

Tabela 6 - Resultados desbalanceamento 5 gramas

Tensão de Alimentação	**Assinatura**		**Desbalanceamento 5g**	
	Aceleração(g)	Velocidade(mm/s)	Aceleração(g)	Velocidade(mm/s)
6V	0,179	2,61	1,06	31
10V	0,465	4,15	4,35	65,56
12V	0,457	6,48	8,14	93,85

Tabela 7 - Resultados desbalanceamento 10 gramas

Tensão de Alimentação	**Assinatura**		**Desbalanceamento 10g**	
	Aceleração(g)	Velocidade(mm/s)	Aceleração(g)	Velocidade(mm/s)
6V	0,179	2,61	1,97	60,5
10V	0,465	4,15	11,28	104,74
12V	0,457	6,48	13,32	132,3

4.2.3.Rolamento defeituoso

Com o objetivo de explorar mais o projeto, foram realizados testes com um rolamento defeituoso. O rolamento "bom" que está posicionado no mancal onde se posicionou o sensor de vibração foi substituído por um rolamento usado (trocado durante a manutenção de um equipamento).

Como defeitos em rolamentos não possuem um local específico no espectro de frequência onde sua amplitude possa ser detectada (varia devido a características construtivas, e a severidade do defeito), espera-se com esse teste que apareçam picos no espectro de frequência com maior amplitude comparando-se com a assinatura do equipamento.

Apenas o espectro da aceleração da vibração foi avaliado nos testes realizados (A variável utilizada para a avaliação de defeitos em rolamentos é a aceleração). Nas Figuras 48 a 50 pode-se observar os espectros de frequência avaliados no teste realizado.

A Tabela 8 mostra os resultados comparativos entre a amplitude máxima detectada no espectro do rolamento defeituoso em comparação com a amplitude máxima do espectro da assinatura do equipamento.

Tabela 8 - Resultado rolamento defeituoso

Tensão de Alimentação	**Assinatura**	**Rolamento Defeituoso**
	Aceleração(g)/Frequência	Aceleração(g)/Frequência
6V	0,179 / 904Hz	0,457 / 1764Hz
10V	0,465 / 2045Hz	0,652 / 1730Hz
12V	0,457 / 1764Hz	0,775 / 1786Hz

Pode-se observar que apesar de pequena, a variação na amplitude do sinal foi detectada. È importante salientar que o rolamento defeituoso é um modelo de rolamento de alto desempenho feito para trabalhar em altas rotações. Trabalhando em baixas rotações o rolamento não apresenta um nível de vibração alto, assim poderia ser utilizado em baixas rotações por mais algum tempo.

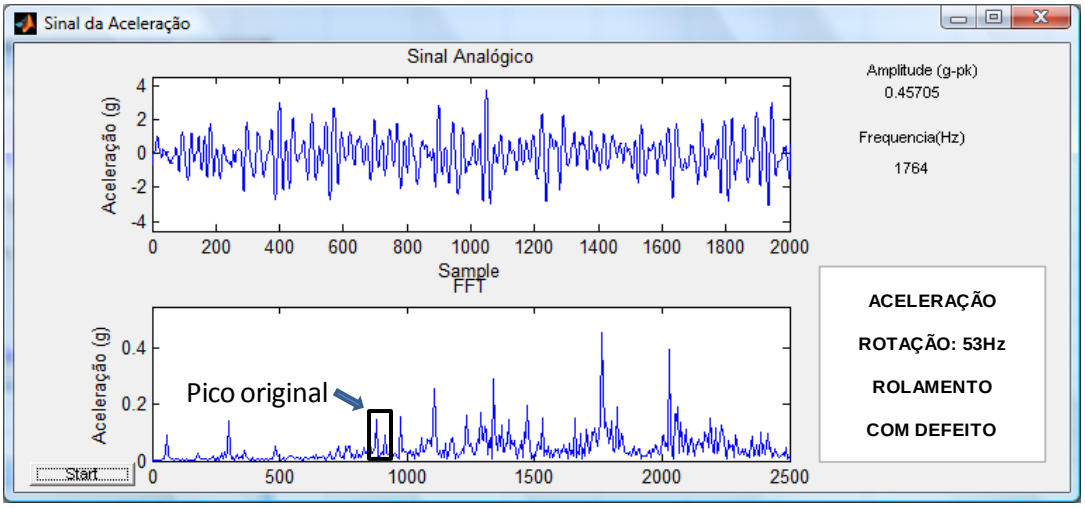


Figura 48- Sinal da aceleração - rolamento defeituoso (≅53HZ).

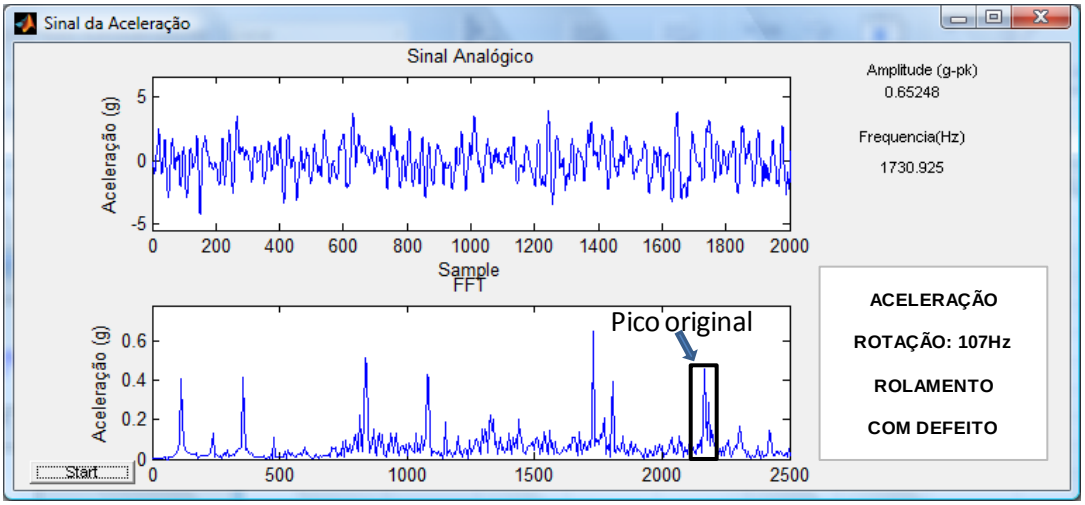


Figura 49 -Sinal da aceleração - rolamento defeituoso (≅107Hz).

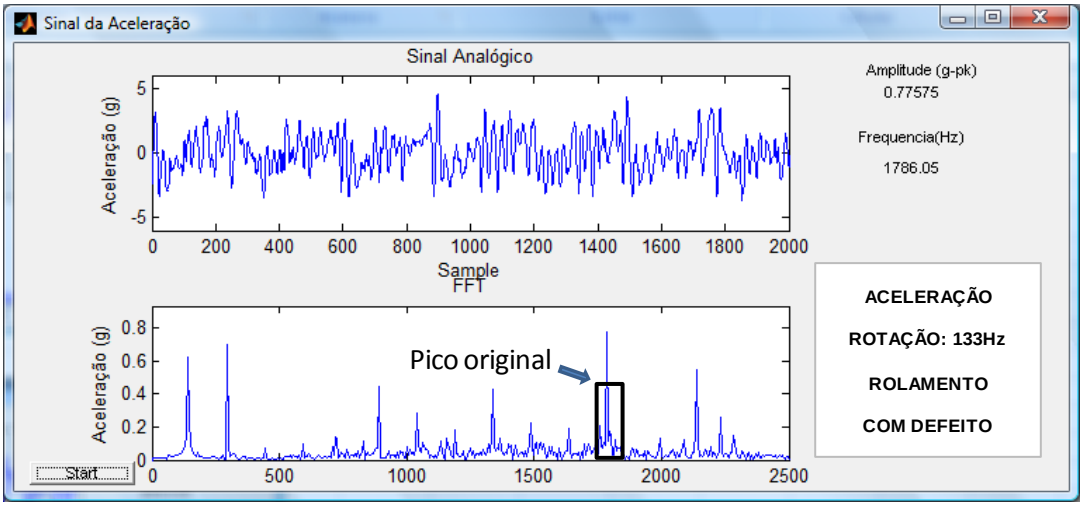


Figura 50 - Sinal da aceleração - rolamento defeituoso (≅133Hz).

5. CONSIDERAÇÕES FINAIS

O objetivo principal deste trabalho, que consistia em desenvolver um equipamento que permitisse a visualização do espectro de vibração de equipamentos rotativos, foi alcançado. Juntamente ao objetivo principal foi desenvolvido um software implementado em MATLAB capaz de gerar um sinal para a mesa de vibração possibilitando assim, que testes de vibração sejam realizados sem a necessidade de equipamentos adicionais como osciloscópio e gerador de sinais.

Um protótipo de equipamento rotativo foi construído com o objetivo de validar a visualização de problemas práticos encontrados na indústria. A visualização do defeito ocorreu conforme o esperado, gerando assim um resultado positivo do teste proposto.

Apesar de todos os cuidados tomados no circuito de condicionamento, o fato de não ter sido encontrado um cabo de instrumentação com o número de condutores suficiente e flexibilidade necessária permitiu a exposição do acelerômetro a ruídos. Um dos ruídos encontrados foi o ruído em 60Hz, que não poderia ser filtrado pois pertence a faixa de frequência de trabalho. A solução encontrada foi a subtração do ruído de 60Hz no espectro de frequência a nível de software. Essa solução apresentou uma resposta satisfatória, atendendo assim o objetivo principal do equipamento projetado.

A escolha de utilizar um auto-falante como mesa de vibração não é a melhor opção, mas para um uso acadêmico/didático supre as necessidades básicas. Ficando como sugestão para trabalhos futuros o estudo da linearização da resposta de vibração do auto-falante em relação à frequência.

O acelerômetro tipo MEMS mostrou-se uma boa opção para projetos didáticos devido ao seu baixo custo e sinal de fácil condicionamento. Atualmente já existem projetos de acelerômetros tipo MEMS com uma faixa de frequência maior, o que possibilitará uma aplicação a nível industrial em breve.

Os equipamentos existentes hoje na indústria para a visualização de níveis de vibração não são muito diferente do equipamento proposto, tendo apenas uma forma construtiva mais robusta e ferramentas de integração com outro tipo de hardware/software como *PLC's* e *SDCD's*.

6. REFERÊNCIAS

[1] MARÇAL, R.F. Martins – **Um método para detectar falhas incipientes em máquinas rotativas baseado em análise de vibrações e lógica Fuzzy** – Proposta de Tese de Doutorado, PPGEM/UFRGS, Porto Alegre, 2000.

[2] FIGUEIREDO, LIGIA J. ; GAFANIZ, ANA R.; LOPES, GUSTAVO S.; PEREIRA, RUBEN – **Aplicações de Acelerômetros** – Trabalho de Mestrado Integrado de Engenharia Biomédica – Instituto Superior Técnico – Lisboa – Portugal 2007.

[3] YA'CUBSOHN, R. V.; **El Diagnostico de fallhas por analisis vibratório**. – Editora Die Techik Ltda. São Paulo, 1983

[4] ALLOCCA, J. A.; **Transducers:theory and application** – Reston Company, Inc. A Prentice-Hall Company, Reston, Virgínia, 1984

[5] DOEBELIN, E. D.; **Measurement Systems Apllication and Design** – Fourth Editon. McGraw Hill, New York, 1990.

[6] FLORA, LEANDRO. D. – **Controle de Aceleração de uma Máquina de Vibração Eletrodinâmica** – Dissertação Curso de Mestrado do PPGEE, Área de Concentração em Controle de Processos, da Universidade Federal de Santa Maria (UFSM,RS), Santa Maria, 2005.

OBRAS CONSULTADAS

CHAPMAN, STEPHEN J. – **Programação em MATLAB para engenheiros**; tradução técnica Flávio Soares Correa da Silva – São Paulo: Editora Pioneira Thomson Learning, 2003.

STOUT, DAVID F. – **Handbook of operational amplifiers circuit design** - McGraw-Hill Book Company.

MARCHAND, PATRICK – **Graphics and GUIs with MATLAB** – 3 ª. Ed – Chapmam & Hall/CRC, 2003.

CHEATLE, KEITH – **Fundamentals of test measurement instrumentation** – ISA-Instrumentation, Systems, and Automation Society.

SOUTH, TIM – **Managing Noise and Vibration at Work** – Elsevier Butterworth-Heinemann; 2004.

SILVA,CLARENCE W. DE. – **Vibration Monitoring, Testing, and Instrumentation** – The University of British Columbia – Canada; 2007.

APÊNDICE A – CÓDIGO FONTE GERADOR

```
function Gerador(varargin)
% GERADOR M-file for Gerador.fig
%      GERADOR, by itself, creates a new GERADOR or raises the existing
%      singleton*.%
%      H = GERADOR returns the handle to a new GERADOR or the handle to
%      the existing singleton*.%
%      GERADOR('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GERADOR.M with the given input arguments.%
%      GERADOR('Property','Value',...) creates a new GERADOR or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Gerador_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Gerador_OpeningFcn via varargin.%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Copyright 2002-2003 The MathWorks, Inc.
% Edit the above text to modify the response to help Gerador
% Last Modified by GUIDE v2.5 16-Oct-2009 18:27:45
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Gerador_OpeningFcn, ...
                   'gui_OutputFcn',  @Gerador_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before Gerador is made visible.
function Gerador_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Gerador (see VARARGIN)
% Choose default command line output for Gerador
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes Gerador wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = Gerador_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double
% --- Executes during object creation, after setting all properties.
```

```
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ao;
switch get(handles.pushbutton1, 'String');
    case ('Start')
    status=0;
    case ('Stop')
    status=1;
end
if (status ==0)
ao=analogoutput('winsound');
addchannel(ao,1:2);
set(ao, 'SampleRate', 44100);
set(ao, 'TriggerType', 'Immediate');
set(ao, 'RepeatOutput', inf);
set(ao,'Tag','Teste');
end
if (status == 0)
F  = str2num(get(findobj('Tag','edit1'), 'String'));
Amplitude  = str2num(get(findobj('Tag','edit2'), 'String'));
A=Amplitude/1000;
A2  = str2num(get(findobj('Tag','edit7'), 'String'));
A2=A2/1000;
A3  = str2num(get(findobj('Tag','edit8'), 'String'));
A3=A3/1000;
A4  = str2num(get(findobj('Tag','edit9'), 'String'));
A4=A4/1000;
A5  = str2num(get(findobj('Tag','edit10'), 'String'));
A5=A5/1000;
A6  = str2num(get(findobj('Tag','edit11'), 'String'));
A6=A6/1000;
A7  = str2num(get(findobj('Tag','edit12'), 'String'));
A7=A7/1000;
A8  = str2num(get(findobj('Tag','edit13'), 'String'));
A8=A8/1000;
A9  = str2num(get(findobj('Tag','edit3'), 'String'));
A9=A9/1000;
axes(handles.axes1);
cla; % linpa os eixos do gráfico
popup_sel_index = get(handles.popupmenu1, 'Value');
switch popup_sel_index
    case 1
        %Criando o Gráfico
        x= 0:1/44100:(4*pi/F); %linspace(0,2*pi,16000);
        y=A*2*sin(F*x);
        Ymax=max(y)*1.2;
        plot(x,y);
        set(handles.axes1, 'YLim', [-Ymax Ymax]);
        grid on;
        %criando o sinal pra a placa de som
        numPoints = 44100/F;
        t = linspace(0,2*pi,numPoints+1);
        t = t(1:end-1);
        waveout=A*sin(t);
    case 2
        %Criando o Gráfico
        x= 0:1/44100:(4*pi/F);
        y=2*(A*sin(F*x)+A2*sin(x*2*F)+A3*sin(x*3*F)+A4*sin(x*4*F)+A5*sin(5*x*F)+A6*sin(6*x*F)+A7*sin(7*x*F)+A8*sin(8*x*F)+A9*sin(9*x*F));
        Ymax=max(y)*1.2;
        plot(x,y);
        set(handles.axes1, 'YLim', [-Ymax Ymax]);
        grid on;
        %criando o sinal pra a placa de som
        numPoints = 44100/F;
        t = linspace(0,2*pi,numPoints+1);
        t = t(1:end-1);
        waveout=A*sin(t)+A2*sin(2*t)+A3*sin(3*t)+A4*sin(4*t)+A5*sin(5*t)+A6*sin(6*t)+A7*sin(7*t)+A8*sin(8*t)+A9*sin(9*t);
    case 3
        D  = str2num(get(handles.edit15, 'String'));
        %Criando o Gráfico
        x= 0:1/44100:(4*pi/F);
        y=2*A*square(F*x,D);
        Ymax=max(y)*1.2;
        plot(x,y);
        set(handles.axes1, 'YLim', [-Ymax Ymax]);
        grid on;
        %criando o sinal pra a placa de som
        numPoints = 44100/F;
        t = linspace(0,2*pi,numPoints+1);
        t = t(1:end-1);
        waveout=A*square(t,D);
    case 4
            %Criando o Gráfico
        x= 0:1/44100:(4*pi/F);
        y=2*A*sawtooth(F*x,0.5);
        Ymax=max(y)*1.2;
        plot(x,y);
        set(handles.axes1, 'YLim', [-Ymax Ymax]);
        grid on;
```

```
        %criando o sinal pra a placa de som
        numPoints = 44100/F;
        t = linspace(0,2*pi,numPoints+1);
        t = t(1:end-1);
        waveout=A*sawtooth(t,0.5);
      end
  % Determine the maximum y data value
[max_y_value,max_y_index] = max(y);

max1=max(waveout);
max2=max1*2;

set(findobj('Tag','text4'),'String', num2str(max2));
corresponding_x_value = x(max_y_index);
% Put a red circle symbol at the maximum data point
hold on
plot(corresponding_x_value,max_y_value,'or');
hold off
% Create a string vector
our_string = sprintf('%g is the maximum data point',...
            max_y_value);
% Put the string into the graph at the max y value
text(corresponding_x_value,max_y_value+0.5,our_string);
putdata(ao, [waveout' waveout']);
start(ao);
set(handles.pushbutton1, 'String','Stop');
end
if(status == 1)
stop(ao);
delete(ao);
set(handles.pushbutton1, 'String','Start');
end

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1

popup_sel_index = get(handles.popupmenu1, 'Value');
switch popup_sel_index
    case 1
        set(findobj('Tag','uipanel2'),'Visible', 'off');
        set(handles.uipanel16,'Visible', 'off');
    case 2
         set(findobj('Tag','uipanel2'),'Visible', 'on');
        set(handles.uipanel16,'Visible', 'off');
    case 3
        set(findobj('Tag','uipanel2'),'Visible', 'off');
        set(handles.uipanel16,'Visible', 'on');
        case 4
        set(findobj('Tag','uipanel2'),'Visible', 'off');
        set(handles.uipanel16,'Visible', 'off');
  end
% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double
% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double
% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a double
% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a double
% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a double
% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11 as a double
% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit12 as text
%        str2double(get(hObject,'String')) returns contents of edit12 as a double
% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function edit13_Callback(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit13 as text
%        str2double(get(hObject,'String')) returns contents of edit13 as a double
% --- Executes during object creation, after setting all properties.
function edit13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit14 as text
%        str2double(get(hObject,'String')) returns contents of edit14 as a double
% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --------------------------------------------------------------------
function Untitled_1_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --------------------------------------------------------------------
function Untitled_2_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
demoai_fft;
% --------------------------------------------------------------------
function Untitled_3_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background, change
%       'usewhitebg' to 0 to use default.  See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --- Executes during object creation, after setting all properties.
function figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% --------------------------------------------------------------------
function Arquivo_Callback(hObject, eventdata, handles)
% hObject    handle to Arquivo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
function edit15_Callback(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit15 as text
%        str2double(get(hObject,'String')) returns contents of edit15 as a double
% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

APÊNDICE B – CÓDIGO AQUISIÇÃO ACELERAÇÃO

```
function fft(varargin)
%%
% Error if an output argument is supplied.
if nargout > 0
   error('Too many output arguments.');
end
%%
% Based on the number of input arguments call the appropriate
% local function.
switch nargin
case 0
   % Create the analog input object.
   data = localInitAI;
   % Create the  figure.
   data = localInitFig(data);
   hFig = data.handle.figure;
case 1
   error('The ADAPTORNAME, ID and CHANID must be specified.');
case 2
   % Initialize variables.
   data = [];
   action=varargin{1};
   hFig=varargin{2};
   % This may fail if only the ADAPTORNAME and ID were input.  However,
   % if the ID was 0, this will work.
   try
      data=get(hFig,'UserData');
   end
   % DATA will be empty if CHANID was not specified or if ID = 0.
   if isempty(data)
      error('The CHANID must be specified.');
   end
   % Based on the action, call the appropriate local function.
   switch action
   case 'close'
      localClose(data);
   case 'stop'
      data = localStop(data);
   end
case 3
      % User specified the input - adaptor, id, chanNum.
      % Create the analog input object.
      [data, errflag] = localInitAI(varargin{:});
      if errflag
         error(lasterr)
      end
      % Create the  figure.
      data = localInitFig(data);
      hFig = data.handle.figure;
end
%%
% Update the figure's UserData.
if ~isempty(hFig)&ishandle(hFig),
   set(hFig,'UserData',data);
end
%%
% Update the analog input object's UserData.
if isvalid(data.ai)
   set(data.ai, 'UserData', data);
end
%% ************************************************************************
% Create the object and get the first fft.
function [data, errflag] = localInitAI(varargin)
% Initialize variables.
errflag = 0;
data = [];
%%
% Either no input arguments or all three - ADAPTORNAME, ID and CHANNELID.
switch nargin
case 0
   adaptor = 'winsound';
   id = 0;
   chan = 1;
case 3
   adaptor = varargin{1};
   id = varargin{2};
   chan = varargin{3};
otherwise
   lasterr('The ADAPTORNAME, ID and CHANID must be specified.');
   errflag = 1;
   return;
end
%%
% Error if more than one channel was specified.
if length(chan) > 1
   lasterr('Only a single channel can be created.');
   errflag = 1;
   return
end
%%
% Channel 2 for sound card is not allowed.
if strcmp(lower(adaptor), 'winsound') & chan == 2
   warning('Channel 1 must be used for device Winsound.');
```

```
    chan = 1;
end
%%
% Object Configuration.
% Create an analog input object with one channel.
ai = analoginput(adaptor, id);
addchannel(ai, chan);
%%
% Configure the analog input object.
set(ai, 'SampleRate', 44100);
%%
% Configure the analog input object to trigger manually twice.
set(ai, 'SamplesPerTrigger', 8000);
set(ai, 'TriggerRepeat', 1);
set(ai, 'TriggerType', 'manual');
%%
% Initialize callback parameters.  The TimerAction is initialized
% after figure has been created.
set(ai, 'TimerPeriod', 0.1);
set(ai, 'BufferingConfig',[8000,20]);
%%
% Object Execution.
% Start the analog input object.
start(ai);
trigger(ai);
%%
% Obtain the available time and data.
[d,time] = getdata(ai, ai.SamplesPerTrigger);
%%
% Calculate the fft.
Fs = get(ai, 'SampleRate');
blockSize = get(ai, 'SamplesPerTrigger');
[f,mag] = localDaqfft(d,Fs,blockSize);
%%
% Update the data structure.
data.ai = ai;
data.getdata = [d time];
data.daqfft = [f mag];
data.handle = [];
%%
% Set the object's UserData to data.
set(data.ai, 'UserData', data);
%% ************************************************************************
% Create the display.
function data = localInitFig(data)
%%
% Initialize variables.
btnColor=get(0,'DefaultUIControlBackgroundColor');
%%
% Position the GUI in the middle of the screen
screenUnits=get(0,'Units');
set(0,'Units','pixels');
screenSize=get(0,'ScreenSize');
set(0,'Units',screenUnits);
figWidth=750;
figHeight=320;
figPos=[(screenSize(3)-figWidth) (screenSize(4)-figHeight)/1.1  ...
     figWidth                    figHeight];
%%
% Create the figure window.
hFig=figure(...
   'Color'             ,btnColor                 ,...
   'IntegerHandle'     ,'off'                    ,...
   'DoubleBuffer'      ,'on'                     ,...
   'DeleteFcn'         ,'fft(''close'',gcbf)',...
   'MenuBar'           ,'none'                   ,...
   'HandleVisibility'  ,'on'                     ,...
   'Name'              ,'Sinal da Aceleração'  ,...
   'Tag'               ,'Analog Input FFT'  ,...
   'NumberTitle'       ,'off'                    ,...
   'Units'             ,'pixels'                 ,...
   'Position'          ,figPos                   ,...
   'UserData'          ,[]                       ,...
   'Colormap'          ,[]                       ,...
   'Pointer'           ,'arrow'                  ,...
   'Visible'           ,'off'                    ...
   );
%%
% Create Data subplot.
hAxes(1) = axes(...
   'Position'          , [0.1300 0.5811 0.6050 0.3439],...
   'Parent'            , hFig,...
   'XLim'              , [0 get(data.ai, 'SamplesPerTrigger')],...
   'YLim'              , [-1 1]...
   );
%%
% Plot the data.
hLine(1) = plot(data.getdata(:,1));
%set(hAxes(1), 'XLim', [0 get(data.ai, 'SamplesPerTrigger')]);
set(hAxes(1), 'XLim',[0 2000]);
%%
% Label the plot.
xlabel('Sample');
ylabel('Aceleração (g)');
title('Sinal Analógico');
%%
% Create the FFT subplot.
hAxes(2) = axes(...
   'Position'          , [0.1300 0.0700 0.6050 0.3439],...
   'Parent'            , hFig,...
   'XLim'              , [0 1000]...
   );
%%
```

```
% Plot the data.
hLine(2) = plot(data.daqfft(:,1),data.daqfft(:,2));
%set(hAxes(2), 'XLim', [0 max(data.daqfft(:,1))]);
set(hAxes(2), 'XLim', [0 2500]);
%%
% Label the plot.
xlabel('Frequency (Hz)');
ylabel('Aceleração (g)');
title('FFT');
%%
% Criar Text.
htoggle = uicontrol(...
   'Parent'          , hFig,...
   'Style'           , 'text',...
   'Units'           , 'normalized',...
   'Position'        , [0.80 0.9 0.12 0.06],... %[x,y largura, altura]
   'Value'           , 1,...
   'String'          , 'Amplitude (g-pk)',...
   'Tag'             ,'text1'      ,...
   'Callback'        , ' ' );
% Criar text.
htoggle2 = uicontrol(...
   'Parent'          , hFig,...
   'Style'           , 'text',...
   'Units'           , 'normalized',...
   'Position'        , [0.80 0.85 0.10 0.06],... %[x,y largura, altura]
   'Value'           , 1,...
   'String'          , '',...
   'Tag'             ,'text2'      ,...
   'Callback'        , '%automatic');
% Criar text.
htoggle3 = uicontrol(...
   'Parent'          , hFig,...
   'Style'           , 'text',...
   'Units'           , 'normalized',...
   'Position'        , [0.80 0.68 0.10 0.06],... %[x,y largura, altura]
   'Value'           , 1,...
   'String'          , '',...
   'Tag'             ,'text3'      ,...
   'Callback'        , '%automatic');
% Criar Text.
htoggle4 = uicontrol(...
   'Parent'          , hFig,...
   'Style'           , 'text',...
   'Units'           , 'normalized',...
   'Position'        , [0.80 0.75 0.10 0.06],... %[x,y largura, altura]
   'Value'           , 1,...
   'String'          , 'Frequencia(Hz)',...
   'Tag'             ,'text4'      ,...
   'Callback'        , ' ' );
%%
% Criar .
htoggle = uicontrol(...
   'Parent'          , hFig,...
   'Style'           , 'pushbutton',...
   'Units'           , 'normalized',...
   'Position'        , [0.0150 0.0111 0.1 0.0556],...
   'Value'           , 1,...
   'String'          , 'Stop',...
   'Tag'             ,'pushbutton1',...
   'Callback'        , 'fft(''stop'', gcbf);');
%%
% Store the handles in the data matrix.
data.handle.figure = hFig;
data.handle.axes = hAxes;
data.handle.line = hLine;
data.handle.toggle = htoggle;
data.handle.toggle2 = htoggle2;
data.handle.toggle3 = htoggle3;
data.handle.toggle4 = htoggle4;
data.state = 0;
%%
% Set the axes handlevisibility to off.
set(hAxes, 'HandleVisibility', 'off');
%%
% Store the data matrix and display figure.
set(hFig,'Visible','on','UserData',data,'HandleVisibility', 'off');
%%
% Configure the callback to update the display.
set(data.ai, 'TimerFcn', @localfftShowData);
%% ************************************************************************
% Close the figure window.
function localClose(data)
%%
% Stop the device if it is running and delete the object.
if isvalid(data.ai)
   if strcmp(get(data.ai, 'Running'), 'On')
      stop(data.ai);
   end
   delete(data.ai);
end
%%
% Close the figure window.
delete(data.handle.figure);
%% ************************************************************************
% Stop or start the device.
function data = localStop(data)
%%
% Based on the state either stop or start.
if strcmp(data.ai.Running, 'On')
   % Stop the device.
   stop(data.ai);
   set(data.handle.toggle, 'String', 'Start');
```

```
    % Store the new state.
    data.state = 1;
else
    % Toggle the Start/Stop string.
    set(data.handle.toggle, 'String', 'Stop');
    % Store the new state.
    data.state = 0;
    delete(data.ai);
    delete(data.handle.figure);
    fft;
end
%% ************************************************************************
% Calculate the fft of the data.
function [f, mag] = localDaqfft(data,Fs,blockSize)
%%
% Calculate the fft of the data.
xFFT1 = fft(data);
xFFT = xFFT1/4000;%escalona a FFT pelo numero de amostras (no caso 44100 - Sample Rate)
xfft = abs(xFFT);
%%
% Avoid taking the log of 0.
index = find(xfft == 0);
%xfft(index) = 1e-17;
%mag = 20*log10(xfft);
mag = xfft;
mag = mag(1:blockSize/2);
mag(1)=0;
mag(12)=mag(12)-1.04e-001;%elimina do espectro de vibração o ruido em 60Hz
f = (0:length(mag)-1)*Fs/blockSize;
f = f(:);
%% ************************************************************************
% Update the plot.
function localfftShowData(obj,event)
%%
% Get the handles.
data = obj.UserData;
hFig = data.handle.figure;
hAxes = data.handle.axes;
hLine = data.handle.line;
%%
% Execute a peekdata.
x1 = peekdata(obj, obj.SamplesPerTrigger);
media=mean(x1);
x1=x1-media;
x= x1/0.063;
% 63mV/g é a sensibilidade do sensor
%%
% FFT calculation.
Fs = obj.SampleRate;
blockSize = obj.SamplesPerTrigger;
[f,mag] = localDaqfft(x,Fs,blockSize);
%%
% Dynamically modify Analog axis as we go.
maxX=max(x);
minX=min(x);
yax1=get(hAxes(1),'YLim');
%if minX<yax1(1),
%   yax1(1)=minX*1.2;
%end
%if maxX>yax1(2),
%   yax1(2)=maxX*1.2;
%end
yax1(1)=minX*1.2;
yax1(2)=maxX*1.2;
set(hAxes(1),'YLim',yax1)
%%
% Dynamically modify Frequency axis as we go.
maxF=max(f);
minF=min(f);
xax=get(hAxes(2),'XLim');
if minF<xax(1),
   xax(1)=minF;
end
if maxF>xax(2),
   xax(2)=maxF;
end
%set(hAxes(2),'XLim',xax)
% Dynamically modify Magnitude axis as we go.
maxM=max(mag);
minM=min(mag);
FreqMax=find(mag == maxM);
FF=f(FreqMax);
set(data.handle.toggle2,'String',num2str(maxM));
set(data.handle.toggle3,'String',num2str(FF));
yax2=get(hAxes(2),'YLim');
if maxM<0.1,
   yax2(2)=0.1;
end
if maxM>0.1,
   yax2(2)=maxM*1.2;
end
yax2(1)=0;
%yax2(2)=maxM*1.2;
set(hAxes(2),'YLim',yax2)
%%
% Update the plots.
set(hLine(1), 'YData', x(:,1));
set(hLine(2), 'XData', f(:,1), 'YData', mag(:,1));
str=get(data.handle.toggle, 'String');
if strcmp(str,'Start')
    set(data.handle.toggle, 'String', 'Stop');
end
drawnow;
```

APÊNDICE C – CÓDIGO AQUISIÇÃO VELOCIDADE

```
function Int_fft(varargin)
%%
% Error if an output argument is supplied.
if nargout > 0
   error('Too many output arguments.');
end
%%
% Based on the number of input arguments call the appropriate
% local function.
switch nargin
case 0
   % Create the analog input object.
   data = localInitAI;
   % Create the  figure.
   data = localInitFig(data);
   hFig = data.handle.figure;
case 1
   error('The ADAPTORNAME, ID and CHANID must be specified.');
case 2
   % Initialize variables.
   data = [];
   action=varargin{1};
   hFig=varargin{2};
   % This may fail if only the ADAPTORNAME and ID were input.  However,
   % if the ID was 0, this will work.
   try
      data=get(hFig,'UserData');
   end
   % DATA will be empty if CHANID was not specified or if ID = 0.
   if isempty(data)
      error('The CHANID must be specified.');
   end
   % Based on the action, call the appropriate local function.
   switch action
   case 'close'
      localClose(data);
   case 'stop'
      data = localStop(data);
   end
case 3
      % User specified the input - adaptor, id, chanNum.
      % Create the analog input object.
      [data, errflag] = localInitAI(varargin{:});
      if errflag
         error(lasterr)
      end
      % Create the  figure.
      data = localInitFig(data);
      hFig = data.handle.figure;
end
%%
% Update the figure's UserData.
if ~isempty(hFig)&ishandle(hFig),
   set(hFig,'UserData',data);
end
%%
% Update the analog input object's UserData.
if isvalid(data.ai)
   set(data.ai, 'UserData', data);
end
%% ************************************************************************
% Create the object and get the first fft.
function [data, errflag] = localInitAI(varargin)
% Initialize variables.
errflag = 0;
data = [];
%%
% Either no input arguments or all three - ADAPTORNAME, ID and CHANNELID.
switch nargin
case 0
   adaptor = 'winsound';
   id = 0;
   chan = 1;
case 3
   adaptor = varargin{1};
   id = varargin{2};
   chan = varargin{3};
otherwise
   lasterr('The ADAPTORNAME, ID and CHANID must be specified.');
   errflag = 1;
   return;
end
%%
% Error if more than one channel was specified.
if length(chan) > 1
   lasterr('Only a single channel can be created.');
   errflag = 1;
   return
end
%%
% Channel 2 for sound card is not allowed.
if strcmp(lower(adaptor), 'winsound') & chan == 2
   warning('Channel 1 must be used for device Winsound.');
```

```
    chan = 1;
end
%%
% Object Configuration.
% Create an analog input object with one channel.
ai = analoginput(adaptor, id);
addchannel(ai, chan);
%%
% Configure the analog input object.
set(ai, 'SampleRate', 44100);
%%
% Configure the analog input object to trigger manually twice.
set(ai, 'SamplesPerTrigger', 8000);
set(ai, 'TriggerRepeat', 1);
set(ai, 'TriggerType', 'manual');
%%
% Initialize callback parameters.  The TimerAction is initialized
% after figure has been created.
set(ai, 'TimerPeriod', 0.1);
set(ai, 'BufferingConfig',[8000,20]);
%%
% Object Execution.
% Start the analog input object.
start(ai);
trigger(ai);
%%
% Obtain the available time and data.
[d,time] = getdata(ai, ai.SamplesPerTrigger);
%Integral=cumtrapz(d);
%d=Integral;
%%
% Calculate the fft.
Fs = get(ai, 'SampleRate');
blockSize = get(ai, 'SamplesPerTrigger');
[f,mag] = localDaqfft(d,Fs,blockSize);
%%
% Update the data structure.
data.ai = ai;
data.getdata = [d time];
data.daqfft = [f mag];
data.handle = [];
%%
% Set the object's UserData to data.
set(data.ai, 'UserData', data);
%% ************************************************************************
% Create the display.
function data = localInitFig(data)
%%
% Initialize variables.
btnColor=get(0,'DefaultUIControlBackgroundColor');
%%
% Position the GUI in the middle of the screen
screenUnits=get(0,'Units');
set(0,'Units','pixels');
screenSize=get(0,'ScreenSize');
set(0,'Units',screenUnits);
figWidth=750;
figHeight=320;
figPos=[(screenSize(3)-figWidth) (screenSize(4)-figHeight)/6  ...
     figWidth                    figHeight];
%%
% Create the figure window.
hFig=figure(...
   'Color'              ,btnColor                  ,...
   'IntegerHandle'      ,'off'                     ,...
   'DoubleBuffer'       ,'on'                      ,...
   'DeleteFcn'          ,'Int_fft(''close'',gcbf)',...
   'MenuBar'            ,'none'                    ,...
   'HandleVisibility'   ,'on'                      ,...
   'Name'               ,'Sinal da Velocidade'  ,...
   'Tag'                ,'Analog Input FFT'  ,...
   'NumberTitle'        ,'off'                     ,...
   'Units'              ,'pixels'                  ,...
   'Position'           ,figPos                    ,...
   'UserData'           ,[]                        ,...
   'Colormap'           ,[]                        ,...
   'Pointer'            ,'arrow'                   ,...
   'Visible'            ,'off'                     ...
   );
%%
% Create Data subplot.
hAxes(1) = axes(...
   'Position'           , [0.1300 0.5811 0.6050 0.3439],...
   'Parent'             , hFig,...
   'XLim'               , [0 get(data.ai, 'SamplesPerTrigger')],...
   'YLim'               , [-1 1]...
   );
%%
% Plot the data.
hLine(1) = plot(data.getdata(:,1));
set(hAxes(1), 'XLim', [0 2000]);
%%
% Label the plot.
xlabel('Sample');
ylabel('Velocidade (mm/s)');
title('Sinal Analógico - Velocidade');
%%
% Create the FFT subplot.
hAxes(2) = axes(...
   'Position'           , [0.1300 0.1100 0.6050 0.3439],...
   'Parent'             , hFig,...
   'XLim'               , [0 1000]...
   );
```

```
%%
% Plot the data.
hLine(2) = plot(data.daqfft(:,1),data.daqfft(:,2));
%set(hAxes(2), 'XLim', [0 max(data.daqfft(:,1))]);
set(hAxes(2), 'XLim', [0 2500]);
%%
% Label the plot.
xlabel('Frequency (Hz)');
ylabel('Velocidade (mm/s)');
title('FFT');
%%
% Create a start/stop pushbutton.
htoggle = uicontrol(...
   'Parent'          , hFig,...
   'Style'           , 'text',...
   'Units'           , 'normalized',...
   'Position'        , [0.80 0.9 0.13 0.06],... %[x,y largura, altura]
   'Value'           , 1,...
   'String'          , 'Amplitude (mm/s-pk)',...
   'Tag'             ,'text1'      ,...
   'Callback'        , ' ' );

% Create a start/stop pushbutton.
htoggle2 = uicontrol(...
   'Parent'          , hFig,...
   'Style'           , 'text',...
   'Units'           , 'normalized',...
   'Position'        , [0.80 0.85 0.10 0.06],... %[x,y largura, altura]
   'Value'           , 1,...
   'String'          , '',...
   'Tag'             ,'text2'      ,...
   'Callback'        , '%automatic');

% Criar text.
htoggle3 = uicontrol(...
   'Parent'          , hFig,...
   'Style'           , 'text',...
   'Units'           , 'normalized',...
   'Position'        , [0.80 0.68 0.10 0.06],... %[x,y largura, altura]
   'Value'           , 1,...
   'String'          , '',...
   'Tag'             ,'text3'      ,...
   'Callback'        , '%automatic');
% Criar Text.
htoggle4 = uicontrol(...
   'Parent'          , hFig,...
   'Style'           , 'text',...
   'Units'           , 'normalized',...
   'Position'        , [0.80 0.75 0.10 0.06],... %[x,y largura, altura]
   'Value'           , 1,...
   'String'          , 'Frequencia(Hz)',...
   'Tag'             ,'text4'      ,...
   'Callback'        , ' ' );
%%
% Criar .
htoggle = uicontrol(...
   'Parent'          , hFig,...
   'Style'           , 'pushbutton',...
   'Units'           , 'normalized',...
   'Position'        , [0.0150 0.0111 0.1 0.0556],...
   'Value'           , 1,...
   'String'          , 'Stop',...
   'Tag'             ,'pushbutton1',...
   'Callback'        , 'Int_fft("stop", gcbf);');
%%
% Store the handles in the data matrix.
data.handle.figure = hFig;
data.handle.axes = hAxes;
data.handle.line = hLine;
data.handle.toggle = htoggle;
data.handle.toggle2 = htoggle2;
data.handle.toggle3 = htoggle3;
data.handle.toggle4 = htoggle4;
data.state = 0;
%%
% Set the axes handlevisibility to off.
set(hAxes, 'HandleVisibility', 'off');
%%
% Store the data matrix and display figure.
set(hFig,'Visible','on','UserData',data,'HandleVisibility', 'off');
%%
% Configure the callback to update the display.
set(data.ai, 'TimerFcn', @localfftShowData);
%% ************************************************************************
% Close the figure window.
function localClose(data)
%%
% Stop the device if it is running and delete the object.
if isvalid(data.ai)
   if strcmp(get(data.ai, 'Running'), 'On')
      stop(data.ai);
   end
   delete(data.ai);
end
%%
% Close the figure window.
delete(data.handle.figure);
%% ************************************************************************
% Stop or start the device.
function data = localStop(data)
%%
% Based on the state either stop or start.
```

```
if strcmp(data.ai.Running, 'On')
   % Stop the device.
   stop(data.ai);
   set(data.handle.toggle, 'String', 'Start');
   % Store the new state.
   data.state = 1;
else
   % Toggle the Start/Stop string.
   set(data.handle.toggle, 'String', 'Stop');
   % Store the new state.
   data.state = 0;
   % Stop the device if it is running and delete the object.
if isvalid(data.ai)
   if strcmp(get(data.ai, 'Running'), 'On')
      stop(data.ai);
   end
   delete(data.ai);
end
%%
% Close the figure window.
   delete(data.handle.figure);
%Start uma nova janela
   Int_fft;
end
%% ***********************************************************************
% Calculate the fft of the data.
function [f, mag] = localDaqfft(data,Fs,blockSize)
%%
% Calculate the fft of the data.
xFFT1 = fft(data);
xFFT = xFFT1/4000;
xfft = abs(xFFT);
%%
% Avoid taking the log of 0.
%index = find(xfft == 0);
%xfft(index) = 1e-17;
%mag = 20*log10(xfft);
mag = xfft;
mag = mag(1:blockSize/2);
mag(1)=1e-11;
mag(2)= mag(2)-1;
mag(3)=mag(3)-1;
mag(11)-mag(11)-0.34;
mag(12)=mag(12)-2.7;
index =find(mag < 0);
mag(index)= 1e-11;
f = (0:length(mag)-1)*Fs/blockSize;
f = f(:);
%% ***********************************************************************
% Update the plot.
function localfftShowData(obj,event)
%%
% Get the handles.
data = obj.UserData;
hFig = data.handle.figure;
hAxes = data.handle.axes;
hLine = data.handle.line;
%%
% Execute a peekdata.
x2a = peekdata(obj, obj.SamplesPerTrigger);
media=mean(x2a);
x2=x2a-media;
%Calculando o Filtro Digital
%Fs1 = get(data.ai, 'SampleRate');
%Wn=(10/(Fs1));
%f = [0 Wn Wn*1.1 1];
%m = [0 0 1 1];
%b=fir2(50,f,m);
%hd = dfilt.dffir(b);
%x1=filter(hd,x2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x1=((x2*9.8)/0.063);
xa=cumtrapz(x1);
xmedia=mean(xa);
x=(xa-xmedia)*1e3*2.27621943225e-5;%passa de m/s para mm/s
%%
% FFT calculation.
Fs = obj.SampleRate;
blockSize = obj.SamplesPerTrigger;
[f,mag] = localDaqfft(x,Fs,blockSize);
%%
% Dynamically modify Analog axis as we go.
maxX=max(x);
minX=min(x);
yax1=get(hAxes(1),'YLim');
%if minX<yax1(1),
 %  yax1(1)=minX;
%end
%if maxX>yax1(2),
 %  yax1(2)=maxX;
%end
yax1(1)=minX*1.2;
yax1(2)=maxX*1.2;
set(hAxes(1),'YLim',yax1)
%%
% Dynamically modify Frequency axis as we go.
maxF=max(f);
minF=min(f);
xax=get(hAxes(2),'XLim');
if minF<xax(1),
   xax(1)=minF;
end
if maxF>xax(2),
```

```
   xax(2)=maxF;
end
%set(hAxes(2),'XLim',xax)
%%
% Dynamically modify Magnitude axis as we go.
maxM=max(mag);
minM=min(mag);
FreqMax=find(mag == maxM);
FF=f(FreqMax);
set(data.handle.toggle2,'String',num2str(maxM));
set(data.handle.toggle3,'String',num2str(FF));
yax2=get(hAxes(2),'YLim');
if maxM<5,
   yax2(2)=5;
end
if maxM>5,
   yax2(2)=maxM*1.2;
end
yax2(1)=0;
set(hAxes(2),'YLim',yax2)
%%
% Update the plots.
set(hLine(1), 'YData', x(:,1));
set(hLine(2), 'XData', f(:,1), 'YData', mag(:,1));
str=get(data.handle.toggle, 'String');
if strcmp(str,'Start')
   set(data.handle.toggle, 'String', 'Stop');
end

drawnow;
```

APÊNDICE D – CÓDIGO TELA INICIAL

```
function varargout = Analise_Vibracao(varargin)
% ANALISE_VIBRACAO M-file for Analise_Vibracao.fig
%      ANALISE_VIBRACAO, by itself, creates a new ANALISE_VIBRACAO or raises the existing
%      singleton*.
%
%      H = ANALISE_VIBRACAO returns the handle to a new ANALISE_VIBRACAO or the handle to
%      the existing singleton*.
%
%      ANALISE_VIBRACAO('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ANALISE_VIBRACAO.M with the given input arguments.
%
%      ANALISE_VIBRACAO('Property','Value',...) creates a new ANALISE_VIBRACAO or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Analise_Vibracao_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Analise_Vibracao_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Copyright 2002-2003 The MathWorks, Inc.
% Edit the above text to modify the response to help Analise_Vibracao
% Last Modified by GUIDE v2.5 31-Oct-2009 16:03:59
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Analise_Vibracao_OpeningFcn, ...
                   'gui_OutputFcn',  @Analise_Vibracao_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);

if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before Analise_Vibracao is made visible.
function Analise_Vibracao_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Analise_Vibracao (see VARARGIN)
% Choose default command line output for Analise_Vibracao
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes Analise_Vibracao wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = Analise_Vibracao_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Gerador; %CHAMA O GERADOR DE FUNÇÕES
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
fft;%CHAMA A AQUISIÇÃO DO SINAL DA ACELERAÇÃO DE VIBRAÇÃO
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
int_fft;% CHAMA A AQUISIÇÃO DO SINAL DA VELOCIDADE DE VIBRAÇÃO
```

ANEXO A – DATASHEET CX20549

1.2 Features

1.2.1 Audio Features

- Compliant with Intel High Definition Audio (HDA) Rev.1.0
- 24-bit stereo DAC and ADC
- Independent sample rate for DAC and ADC. Supports audio formats ranging from 16-bit, 44.1 kHz to 24-bit, 192 kHz
- Voice processing algorithm (optional) – SmartMic technology enhances the clarity of VoIP and the accuracy of voice dictation and command
- SmartJack sensing technology detects up to four jacks using only one sense pin
- Retaskable input/out jacks for either headphone, line out, microphone, and line in
- SmartAudio GUI (optional)—advanced audio control
- Digital Parametric SmartEQ virtually enhances the sound quality on low cost speakers
- Sony Philips Digital Interface (S/PDIF) output
- Simultaneous DAC and SPDIF engines
- +3.3 V analog and I/O operation; uses Vaux for power management modes
- Support both 1.5V and 3.3V signaling with the core logic chipset
- 3D sound effects for games (optional)—EAX and HRTF
- Meets performance requirements for audio on PC2001 systems
- Supports 32/64 bit Windows OS and Linux

1.2.2 General Features

- System compatibility
  - Windows 2000/XP operating system on a MMX 300 MHz-based computer with 64 MB RAM, or equivalent
  - Microsoft's PC 2001 Design Initiative compliant
  - Linux Kernel (contact local Conexant Sales Office for details)
  - Microsoft Windows Logo Program (WLP) V2.2 compliant
- Supports Power Management
  - ACPI Power Management Registers
  - APM support
- Thin package supports low profile designs
  - CX20549-11Z Audio Codec: 32-pin QFN
- +3.3V operation

1.3 Audio Operation

The basic audio capabilities are supported in the HD Audio audio driver.

Optional enhanced audio features are also available (contact local Conexant Sales Office for additional information). These features include:

- HRTF: Head Related Transfer Function
- EAX: Environmental Audio Extension
- Microphone Noise Reduction support
- Bass Management Function
- SmartAudio user interface to manage audio jack retasking, soft equalization, and audio customization
- SmartMic
- Voice Processing Algorithm

1.4 Reference Design

Reference design boards are available to minimize application design time and costs.

Reference design boards are pretested to pass FCC Part 15, FCC Part 68, and TBR 21.

A design package is available in electronic form for each design. The design package includes schematics, bill of materials (BOM), vendor parts list (VPL), board layout files in Gerber format, and complete documentation.

The reference designs are production ready for immediate manufacturing. The design can also be used for the basis of a custom design by the OEM to accelerate design completion for rapid market entry.

HQP **must** be performed for all OEM designs.

ANEXO B – DATASHEET LM10

National Semiconductor

August 2000

LM10
Operational Amplifier and Voltage Reference

General Description

The LM10 series are monolithic linear ICs consisting of a precision reference, an adjustable reference buffer and an independent, high quality op amp.

The unit can operate from a total supply voltage as low as 1.1V or as high as 40V, drawing only 270µA. A complementary output stage swings within 15 mV of the supply terminals or will deliver ±20 mA output current with ±0.4V saturation. Reference output can be as low as 200 mV.

The circuit is recommended for portable equipment and is completely specified for operation from a single power cell. In contrast, high output-drive capability, both voltage and current, along with thermal overload protection, suggest it in demanding general-purpose applications.

The device is capable of operating in a floating mode, independent of fixed supplies. It can function as a remote comparator, signal conditioner, SCR controller or transmitter for analog signals, delivering the processed signal on the same line used to supply power. It is also suited for operation in a wide range of voltage- and current-regulator applications, from low voltages to several hundred volts, providing greater precision than existing ICs.

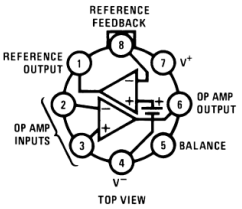
This series is available in the three standard temperature ranges, with the commercial part having relaxed limits. In addition, a low-voltage specification (suffix "L") is available in the limited temperature ranges at a cost savings.

Features

- input offset voltage: 2.0 mV (max)
- input offset current: 0.7 nA (max)
- input bias current: 20 nA (max)
- reference regulation: 0.1% (max)
- offset voltage drift: 2µV/°C
- reference drift: 0.002%/°C

Connection and Functional Diagrams

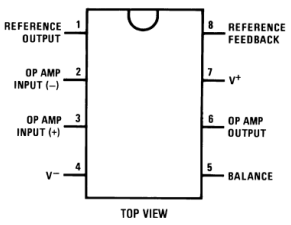
Metal Can Package (H)



00565201

Order Number LM10BH, LM10CH, LM10CLH or LM10H/883 available per SMA# 5962-8760401
See NS Package Number H08A

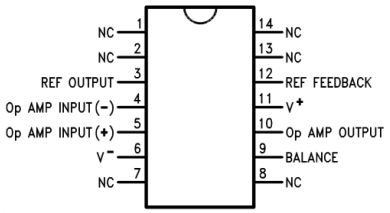
Dual-In-Line Package (N)



00565215

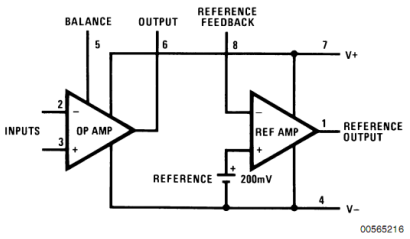
Order Number LM10CN or LM10CLN
See NS Package Number N08E

Small Outline Package (WM)



00565217

Order Number LM10CWM or LM10CWMX
See NS Package Number M14B



00565216

ANEXO C – DATASHEET AD620

ANALOG DEVICES

Low Cost, Low Power Instrumentation Amplifier

AD620

FEATURES
EASY TO USE
Gain Set with One External Resistor (Gain Range 1 to 1000)
Wide Power Supply Range (±2.3 V to ±18 V)
Higher Performance than Three Op Amp IA Designs
Available in 8-Lead DIP and SOIC Packaging
Low Power, 1.3 mA max Supply Current

EXCELLENT DC PERFORMANCE ("B GRADE")
50 μV max, Input Offset Voltage
0.6 μV/°C max, Input Offset Drift
1.0 nA max, Input Bias Current
100 dB min Common-Mode Rejection Ratio (G = 10)

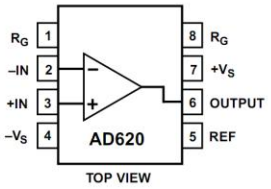
LOW NOISE
9 nV/√Hz, @ 1 kHz, Input Voltage Noise
0.28 μV p-p Noise (0.1 Hz to 10 Hz)

EXCELLENT AC SPECIFICATIONS
120 kHz Bandwidth (G = 100)
15 μs Settling Time to 0.01%

APPLICATIONS
Weigh Scales
ECG and Medical Instrumentation
Transducer Interface
Data Acquisition Systems
Industrial Process Controls
Battery Powered and Portable Equipment

CONNECTION DIAGRAM

8-Lead Plastic Mini-DIP (N), Cerdip (Q) and SOIC (R) Packages



TOP VIEW

PRODUCT DESCRIPTION

The AD620 is a low cost, high accuracy instrumentation amplifier that requires only one external resistor to set gains of 1 to 1000. Furthermore, the AD620 features 8-lead SOIC and DIP packaging that is smaller than discrete designs, and offers lower power (only 1.3 mA max supply current), making it a good fit for battery powered, portable (or remote) applications.

The AD620, with its high accuracy of 40 ppm maximum nonlinearity, low offset voltage of 50 μV max and offset drift of 0.6 μV/°C max, is ideal for use in precision data acquisition systems, such as weigh scales and transducer interfaces. Furthermore, the low noise, low input bias current, and low power of the AD620 make it well suited for medical applications such as ECG and noninvasive blood pressure monitors.

The low input bias current of 1.0 nA max is made possible with the use of Superβeta processing in the input stage. The AD620 works well as a preamplifier due to its low input voltage noise of 9 nV/√Hz at 1 kHz, 0.28 μV p-p in the 0.1 Hz to 10 Hz band, 0.1 pA/√Hz input current noise. Also, the AD620 is well suited for multiplexed applications with its settling time of 15 μs to 0.01% and its cost is low enough to enable designs with one in-amp per channel.

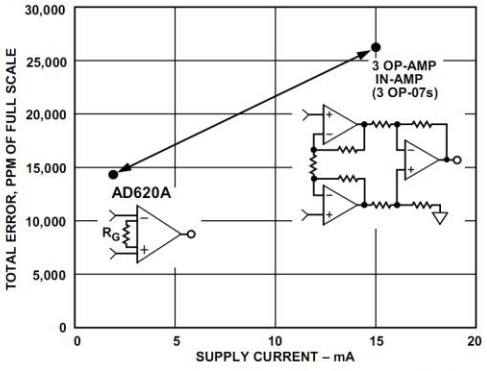


Figure 1. Three Op Amp IA Designs vs. AD620

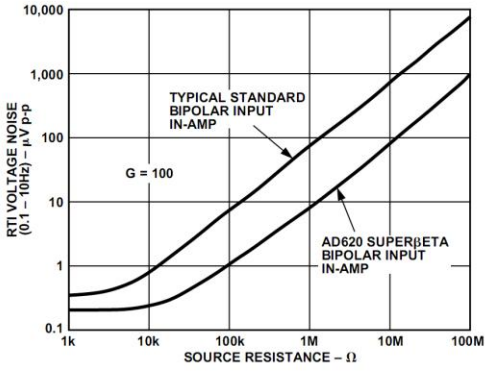


Figure 2. Total Voltage Noise vs. Source Resistance

REV. E

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781/329-4700 World Wide Web Site: http://www.analog.com
Fax: 781/326-8703