PROPOSTA DE SISTEMA DE BAIXO CUSTO DE MONITORAÇÃO DE VIBRAÇÕES

Danilo Castor de Sousa

Projeto de Graduação apresentado ao Curso de Engenharia Mecânica da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientador: Prof. Fernando Augusto de Noronha Castro Pinto; Dr. Ing.

Rio de Janeiro

Fevereiro de 2018

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Departamento de Engenharia Mecânica
DEM/POLI/UFRJ

PROPOSTA DE SISTEMA DE BAIXO CUSTO DE MONITORAÇÃO DE VIBRAÇÕES

Danilo Castor de Sousa

PROJETO FINAL SUBMETIDO AO CORPO DOCENTE DO DEPARTAMENTO DE ENGENHARIA MECÂNICA DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO MECÂNICO.

Aprovado por:

_____
Prof. Fernando Augusto de Noronha Castro Pinto; Dr.Ing.

_____
Prof. Thiago Gamboa Ritto; D.Sc.

_____
Prof. Daniel Alves Castello; D.Sc.

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2018

de Sousa, Danilo Castor
Proposta de Sistema de Baixo Custo de Monitoração de Vibrações/ Danilo Castor de Sousa. – Rio de Janeiro: UFRJ/ Escola Politécnica, 2018.
VIII, 51 p.: il.; 29,7 cm.
Orientador: Fernando Augusto de Noronha Castro Pinto
Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia Mecânica, 2018.
Referencias Bibliográficas: p. 38-39.
1. Monitoração de Máquinas 2. Análise de Vibrações. 3. Programação de Microcontroladores. de Noronha Castro Pinto, Fernando Augusto II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia Mecânica. III. Proposta de Sistema de Baixo Custo de Monitoração de Vibrações

Agradecimentos

Aos meus pais, por terem me fornecido o ambiente propício para que pudesse me dedicar aos estudos. Por terem me ensinado o valor do conhecimento. Por terem me ensinado o que é correto. Por terem me ensinado o que é o amor. Por tudo.

Ao meu irmão, todo o meu agradecimento por estar ao meu lado sempre e por cuidar de nossa família.

À minha namorada Maria Luciana, pelo carinho de sempre, apoio nas horas difíceis, dedicação, humildade e amor que me fez seguir em frente.

A todos os meus professores, que moldaram minha forma de raciocinar e viver.

Ao professor Fernando Castro Pinto e a todos os integrantes do Laboratório de Acústica e Vibrações, que tornaram possível este projeto, com todo o auxílio, experiência e equipamentos cedidos.

Ao meu mentor, o Engenheiro Marcos Lopes, que fez da minha primeira experiência profissional a melhor possível. Muito obrigado.

Aos meus amigos da UFRJ, de equipe de competição, de iniciação científica, da *University of Birmigham*, de estágio e do trabalho, que tornaram cada uma dessas experiências ainda mais marcantes.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/ UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro Mecânico.

Proposta de Sistema de Baixo Custo de Monitoração de Vibrações

Danilo Castor de Sousa

Fevereiro/2018

Orientador: Fernando Augusto de Noronha Castro Pinto; Dr. Ing.

Curso: Engenharia Mecânica

O presente projeto visa criar um protótipo de monitoração de máquinas rotativas com comunicação Wi-Fi para auxílio na tomada de decisão da manutenção do equipamento. O objetivo do protótipo é de realizar o monitoramento contínuo do equipamento através de análises de vibrações e comparar seu nível global com valores de alarme definidos por um modelo estatístico, notificando o usuário por e-mail caso esse limite seja ultrapassado. Uma revisão da literatura sobre análise de vibrações é feita para os defeitos mais frequentes em máquinas rotativas, já que análises em frequência podem ser feitas pelo usuário através da última faixa de medições gravada no protótipo. Foi feita uma pesquisa de mercado das placas de desenvolvimento eletrônicas que poderiam ser utilizadas. Após a escolha dos componentes principais, foi definido o protocolo de comunicação com o acelerômetro, para então ser definido o hardware do sistema. Assim, foi escolhida a linguagem de programação e realizada a programação do microcontrolador. Para a fixação do sensor na máquina, foi fabricada uma base magnética utilizando manufatura aditiva e criação de uma unidade de controle, responsável pelo processamento e armazenamento de dados. Em seguida, foram realizados testes para verificação do funcionamento do protótipo sob condições operacionais.

Palavras-chave: Monitoração de Vibrações, Acelerômetros MEMS, Internet das Coisas, Manutenção Preditiva.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Mechanical Engineer.

Proposal of Low Cost Vibration Monitoring System

Danilo Castor de Sousa

February/2018

Advisor: Fernando Augusto de Noronha Castro Pinto; Dr. Ing.

Course: Mechanical Engineering

This project aims to create a rotary machine condition monitoring prototype with Wi-Fi communication to assist the maintenance decision making. The goal of the prototype is to continuously monitor the condition of the equipment using vibration analysis and compare its overall vibration level with alarm levels defined by statistical methods, notifying the user by email if the alarm values are reached. A vibration analysis review is made for the most frequent defects in rotary machines, since a frequency analysis can be made by the prototype user, using the acceleration data stored in it. A market research of electronic development boards that could fit the project was made. After the choice of the main components, the communication protocol with the accelerometer was defined, so that the hardware of the system could be set. Then, the programming language was chosen and the microcontroller programming was made. In order to fix the accelerometer in the machine, a magnetic probe was developed using additive manufacturing and a unit control was created, responsible for data processing and storage. Tests were realized under operational condition to verify the prototype functioning.

Key-words*:* Vibration monitoring, MEMS Accelerometers, Internet of Things, Predictive Maintenance.

Sumário

Lista de Tabelas

Lista de Figuras

1. Introdução

1.1 Motivação

O uso de tecnologias digitais ainda é pouco explorado pela indústria brasileira, já que apenas 58% conhecem a importância dessas tecnologias para a competitividade da indústria e menos da metade as utiliza. O principal motivo para a baixa adesão dessas tecnologias, para 66% das empresas, é o custo de implantação, seguidos de falta de clareza na definição sobre o retorno e a estrutura e cultura da empresa, com 26% e 24%, respectivamente (CNI, 2016).

A falta de familiaridade das empresas nacionais com as novas tecnologias no cenário atual pode ser encarada como uma oportunidade de novos negócios, já que essa é uma tendência mundial e uma necessidade para nossa indústria se manter competitiva. Assim, analisando os setores que já estão se posicionando para essa nova realidade, podemos ter uma ideia de nichos de mercado na indústria aptos a serem explorados por essas tecnologias:

Tabela 1 - Setores que mais utilizam tecnologias digitais

Setores	%
Equipamentos de informática, produtos eletrônicos e ópticos	61
Máquinas, aparelhos e materiais elétricos	60
Coque, derivados do petróleo e biocombustíveis	53
Máquinas e equipamentos	53
Metalurgia	51
Produtos de material plástico	49
Produtos diversos	49
Produtos têxteis	47
Veículos automotores	46
Químicos (exceto HPPC) (1)	45

(CNI, 2016)

Os setores que envolvem máquinas aparecem na 2ª e 4ª posição da Tabela 1, um indicativo de que a demanda por tecnologias digitais podem ser integradas a componentes intimamente relacionados à Engenharia Mecânica. É nesse contexto que se insere o trabalho atual, de forma a propor uma solução de baixo custo para o monitoramento contínuo de máquinas rotativas.

1.2 Objetivos

O objetivo deste trabalho se resume na criação de um protótipo capaz de monitorar vibrações de máquinas rotativas continuamente de forma autônoma e notificar o usuário por e-mail caso seu nível global de vibrações ultrapasse um valor definido. Além disso, o protótipo deve informar visualmente, no próprio local, se os limites de vibração estabelecidos foram ultrapassados através de LED´s e armazenar o histórico das medidas em um cartão SD. Um arquivo deve conter o registro das médias RMS da aceleração e o outro deve conter a última faixa de medição de acelerações, de forma a auxiliar na tomada de decisão da manutenção do equipamento que está sendo monitorado.

Além do código e estudo dos componentes a serem adquiridos, há a necessidade da fabricação de uma base magnética própria para o acelerômetro utilizado, já que o mesmo foi adaptado para esse propósito. A base deve ser leve, de fácil instalação, remoção e manutenção. A unidade de processamento de dados deve ser arquitetada dentro de uma caixa eletrônica de forma a manter a funcionalidade dos botões do controlador, possuir fácil conexão dos cabos e manter a funcionalidade de remoção do cartão SD, além de proteger os componentes eletrônicos utilizados. Como o custo de implantação é o principal motivo para a baixa adesão de tecnologias que formam a indústria 4.0 no Brasil (CNI, 2016), é de suma importância que o protótipo seja constituído de tecnologias de baixo custo.

2. Medição de vibrações

A análise de vibrações mecânicas é uma ferramenta explorada pela indústria para identificar falhas latentes em máquinas rotativas. Caso seja possível predizer quando ocorrerá a falha do equipamento, essa análise pode ser considerada uma técnica de manutenção preditiva, já que essa é definida como a assistência técnica dada ao equipamento sob o acompanhamento e análise de parâmetros indicativos do estado do sistema (ALVAREZ, 1988). Assim, a análise desses dados é de suma importância para a avaliação do estado da máquina.

Neste capítulo, será abordada a análise periódica que o sistema deve fazer do equipamento, a análise em frequência que pode ser feita com os dados de aceleração gravados no cartão SD e o princípio de funcionamento do acelerômetro utilizado no projeto.

2.1 Acelerômetros MEMS

Acelerômetros MEMS (*Micro Electro Mechanical Systems*) podem ser caracterizados em duas principais categorias: capacitivos e piezo-elétricos. Os acelerômetros capacitivos, que caracterizam o equipamento utilizado neste trabalho, baseiam-se no princípio de variação de capacitância das placas que formam o capacitor. Isso ocorre devido à variação na distância entre essas placas, dada por uma fixa e outra móvel, ocasionadas por forças externas que causam a aceleração do equipamento. Esse tipo de sensor foi escolhido para o projeto devido ao seu baixo custo e fácil integração com microcontroladores.

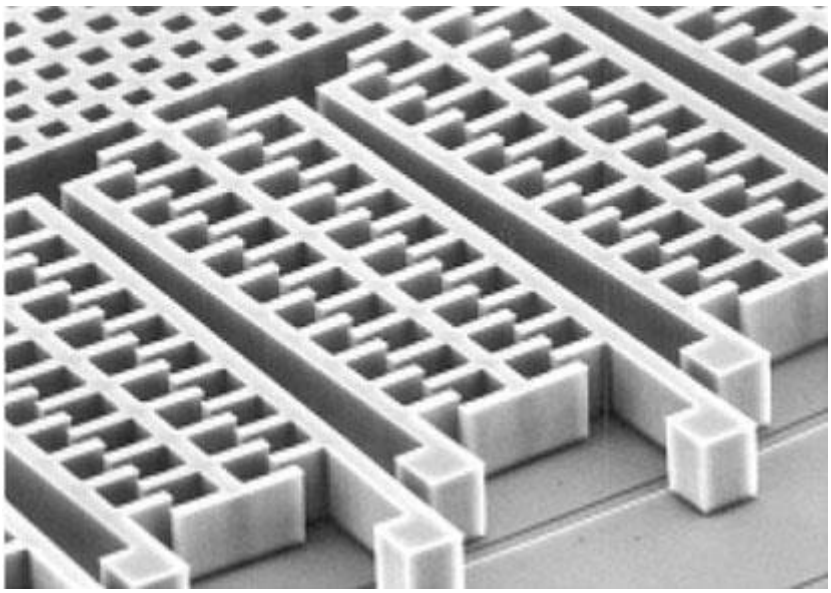


Figura 1 - Placas capacitivas de um acelerômetro MEMS
Fonte: http://anuva.com/blog/a-resource-guide-to-wearable-device-sensors/

Diversos estudos foram realizados para avaliar a performance de acelerômetros MEMS. Albarbar *et al* (2008) comparou acelerômetros MEMS capacitivos com acelerômetros piezo-elétricos comerciais da marca PCB e encontrou resultados similares para ambos em excitações aleatórias e senoidais, apesar de encontrar discrepâncias no ângulo de fase entre os dois. Thanagasundram e Schlindwein (2006) validaram o uso de um modelo de acelerômetro MEMS para o diagnóstico de máquinas, encontrando espectros em frequência similares para os dois sensores.

Além disso, Thanagasundram e Schlindwein (2006) analisaram a diferença nos espectros entre os acelerômetros MEMS com fixações diferentes: base magnética e sensor colado do equipamento. As medições foram comparadas com um sensor piezo-elétrico comercial: o acelerômetro MEMS fixado com base magnética mostrou divergência nas amplitudes do sinal para faixas de frequência em torno de 6.7 kHz, enquanto o sensor colado no equipamento continuou com resultados compatíveis. Assim, o uso de uma base magnética para o presente projeto pode ser utilizada, já que o objetivo principal do trabalho não é fazer análises em frequência dessa ordem, mas criar um protótipo portátil capaz de fazer o monitoramento remoto e contínuo de máquinas rotativas.

2.2 Análise de dados

2.2.1 Nível global de vibrações

A análise periódica do protótipo é feita utilizando a raiz do valor quadrático médio dos valores de aceleração, valor efetivo ou RMS (do inglês *root mean square*), por ser uma média que representa a energia do sinal, definida pela equação (1):

$$RMS = \sqrt{\frac{\sum_{i=1}^{n} {a_i}^2}{n}} \tag{1}$$

Vale ressaltar que os valores de aceleração medidos pelo acelerômetro MEMS são absolutos, ou seja, medem a parcela da gravidade relativa à inclinação do sensor. Para obtermos valores coerentes da média RMS para uma posterior análise, entretanto, as medidas do sinal devem oscilar em torno do valor zero. Isso é possível ao subtrairmos a média do sinal de cada medição de aceleração, obtendo o valor a_i da equação acima. O código do projeto com a implementação dessa rotina está mostrado no Anexo II e uma explicação mais detalhada do mesmo será feita no Capítulo 4.

Assim, podemos comparar os valores de RMS da máquina periodicamente com valores de referência que indiquem o estado do equipamento. Para a definição desses valores, foi utilizado um modelo proposto por Budik *et al* (2016), que consiste em definir os limites de alarme estatisticamente para cada equipamento. O método consiste em

realizar medições de RMS suficientes para uma análise estatística dessa variável (30 medições foram feitas em seu artigo). Com isso, é calculada a variação entre medições de RMS consecutivas, onde a i-ésima variação de medições é dada por:

$$D_i = |RMS_i - RMS_{i-1}| \quad (2)$$

Com isso, podemos medir o limite de alerta (L) através da equação (3):

$$L = \overline{RMS} + 2{,}66\overline{D} \quad (3)$$

Esse limite se encontra a uma distância de 3 desvios padrões da média. Budik *et al* (2016) também propõe um limite de alerta para o aumento da variação de medições de RMS (D_i) estatisticamente, porém esse alarme poderia causar falsos alertas em um monitoramento contínuo, já que um equipamento com variações na sua condição de operação ou até mesmo que seja usado de forma intermitente acusaria uma variação na condição do equipamento, sem que esse necessariamente possua uma falha latente.

Com a teoria proposta, portanto, é possível definir um limite para a média RMS das acelerações. Apesar da ISO 10816 estabelecer condições e procedimentos gerais para a medição e avaliação de vibrações, fornecendo uma tabela com limites de vibrações em máquinas rotativas para cada faixa de potência de máquina, uma análise estatística do sinal específica para cada máquina é preferível para a definição do limite de alarme. Além disso, as variáveis de referência são as médias RMS das velocidades, o que exigiria um maior trabalho computacional do microcontrolador utilizado para realizar a conversão de dados digitais de aceleração para velocidade.

Tabela 2 – Classificação de severidade de vibrações segundo a ISO 10816

SEVERIDADE DE VIBRAÇÃO PELA ISO 10816						
Máquina			Classe I - Máquinas pequenas	Classe II - Máquinas médias	Classe III - Fundação rígida e larga	Classe IV - Fundação macia e larga
	pol/s	mm/s				
Velocidade de Vibração (RMS)	0,01	0,28				
	0,02	0,45		BOM		
	0,03	0,71				
	0,04	1,12				
	0,07	1,80				
	0,11	2,80		SATISFATÓRIO		
	0,18	4,50				
	0,28	7,10		INSATISFATÓRIO		
	0,44	11,20				
	0,70	18,00				
	0,71	28,00		INACEITÁVEL		
	1,10	45,00				

Adaptado de <https://www.reliabilitydirectstore.com/articles.asp?id=122>

2.2.2 FFT e ângulo de fase entre sinais

Além da análise periódica feita pelo protótipo para avaliar o nível global de vibrações, os dados de aceleração devem ser gravados em um cartão SD, permitindo uma análise em frequência do sinal adquirido pelo usuário e ângulo de fase entre sinais. Cada tipo de máquina vibrará de acordo com as frequências características de seus componentes, possuindo, assim, uma assinatura espectral. Essa análise em frequência pode ser feita através da Transformada Rápida de Fourier (FFT – *Fast Fourier Transform*) do sinal discreto. A implementação da FFT feita em um código na linguagem Python utilizando a Spyder IDE (Python 3.6) está disponível no Anexo I.

Para ser feita a análise espectral do equipamento, é importante conhecer os componentes presentes no mesmo que podem ser fontes de vibração, como rolamentos, engrenagens, correias e número de pás do impelidor, caso o objeto de estudo seja uma bomba centrífuga. A velocidade de rotação é um parâmetro fundamental, que deve ser medido para determinar a frequência de vibração de cada componente presente na análise espectral. Defeitos frequentes em máquinas rotativas e suas respectivas características espectrais estão mostrados abaixo:

- Desalinhamento paralelo

O desalinhamento pode fazer com que os rolamentos do eixo sofram cargas maiores do que as especificadas em projeto, causando a falha desse componente. No desalinhamento paralelo, um pico na frequência correspondente a duas vezes a velocidade de rotação (denominado de “2x rpm” adiante, por praticidade) é visualizado no espectro. Na direção radial, uma diferença de 180º no ângulo de fase

é observada entre medições verticais e horizontais no rolamento (SCHEFFER, 2008) (SKF, 2002).

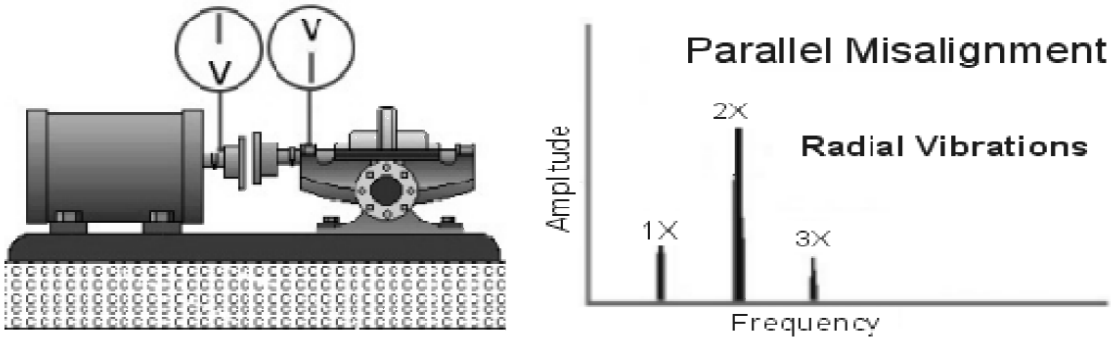


Figura 2 - Representação e espectro de vibrações de um desalinhamento paralelo
Fonte: SCHEFFER, 2008

- Desalinhamento angular

Nesse tipo de desalinhamento, um pico na mesma frequência de rotação da máquina é observada em vibrações axiais, onde harmônicos no dobro e triplo da rotação da máquina também podem aparecer. Na direção axial, um ângulo de fase de 180º é observado quando os sensores são posicionados nos rolamentos anterior e posterior ao acoplamento, conforme mostra a Figura 3, e possuem a mesma orientação (SCHEFFER, 2008).

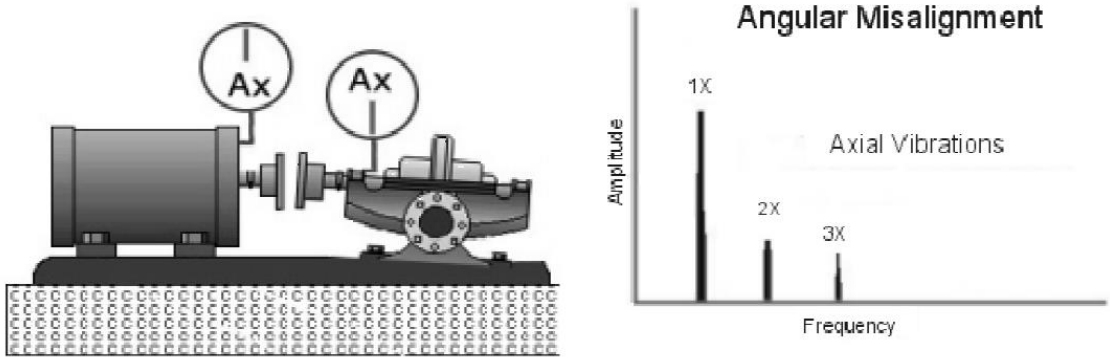


Figura 3 - Representação e espectro de um desalinhamento angular
Fonte: SCHEFFER, 2008

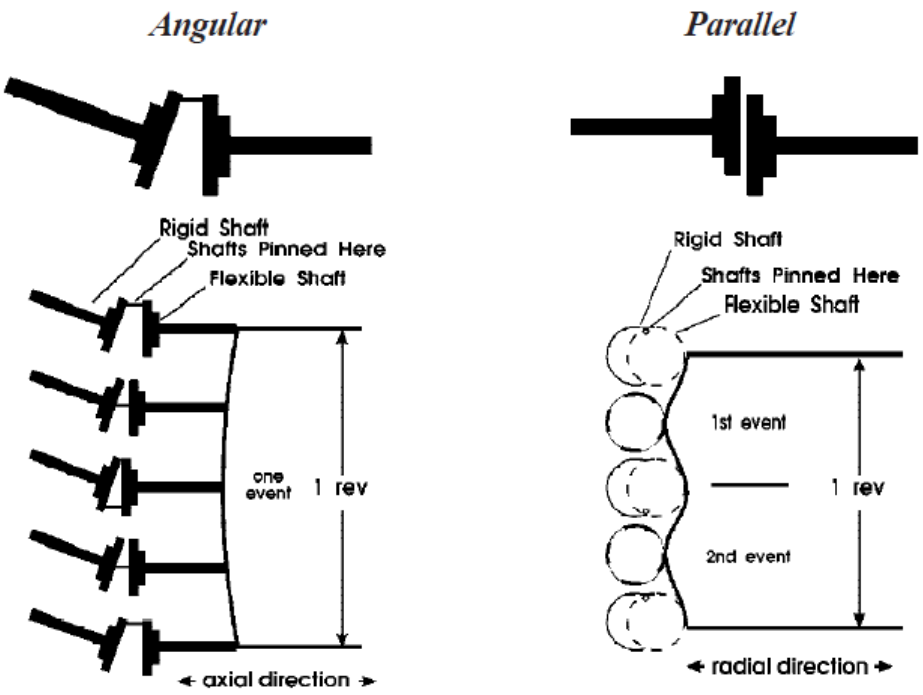


Figura 4 - Modos de vibração de desalinhamento angular e paralelo
Fonte: CM5003 – SKF Vibration Diagnostic Guide

- Desbalanceamento

Existem 3 tipos de desbalanceamento: estático, conjugado e dinâmico, caracterizado como uma combinação dos dois anteriores. Nas 3 condições, é possível observar um pico na mesma frequência de rotação da máquina. Além disso, o ângulo de fase na direção radial entre a medição horizontal e vertical de um mesmo rolamento de 90º é outro indicativo de desbalanceamento (SCHEFFER, 2008) (SKF, 2002).

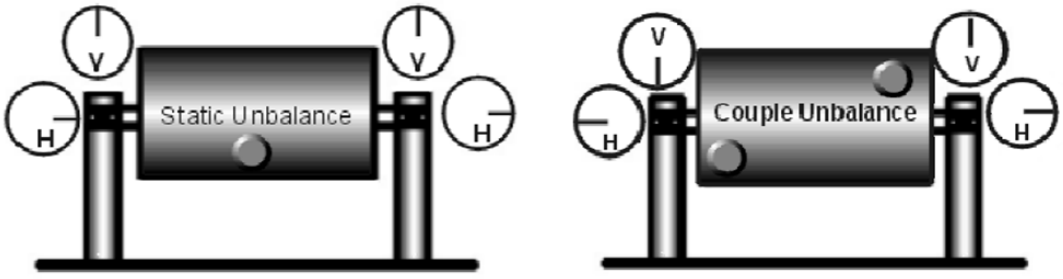


Figura 5 - Desbalanceamento estático e conjugado
Fonte: SCHEFFER, 2008

- Folgas mecânicas

Vibrações causadas por folgas mecânicas podem ser identificadas por múltiplos síncronos da frequência de rotação em seu espectro. Segundo Scheffer (2008), harmônicos e sub-harmônicos de 1/2 ou 1/3 da frequência de rotação da máquina podem estar presentes. Além disso, é um fenômeno altamente direcional: medições radiais com espaçadas de 30º podem mostrar diferenças significativas. SKF (2002) sugere que harmônicos e sub-harmônicos de ½ com magnitudes 20% maiores que a da frequência de rotação são indicativos de folgas mecânicas.

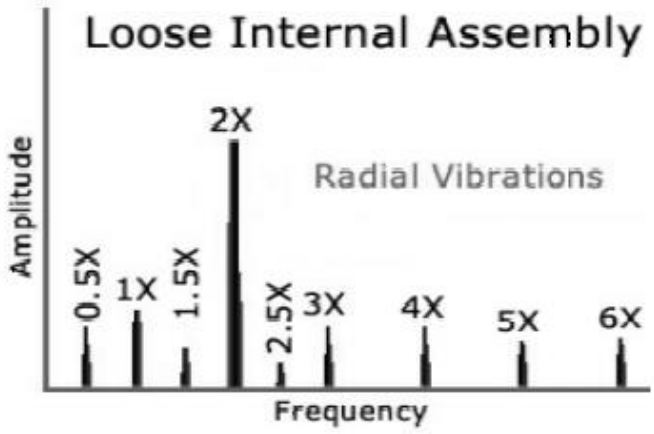


Figura 6 - Espectro de vibrações com folgas mecânicas
Fonte: SCHEFFER, 2008

- Eixo empenado

Segundo Scheffer (2008), quando há empenamento do eixo, as vibrações axiais podem ser maiores que as radiais. Os picos no espectro de frequência são de 1x rpm, quando o empenamento está perto do centro do eixo, e 2x rpm, quando este se localiza próximo à sua extremidade. Os ângulos de fase nas direções radiais e axiais são de 180º. SKF (2002) afirma que o nível de vibrações globais e espectro dessa condição são idênticos a de um desalinhamento: a diferença se dá apenas pela análise do ângulo de fase na direção axial em extremidades opostas da máquina que deve ser de 180º, em concordância com a análise de ângulo de fase de Scheffer (2008).

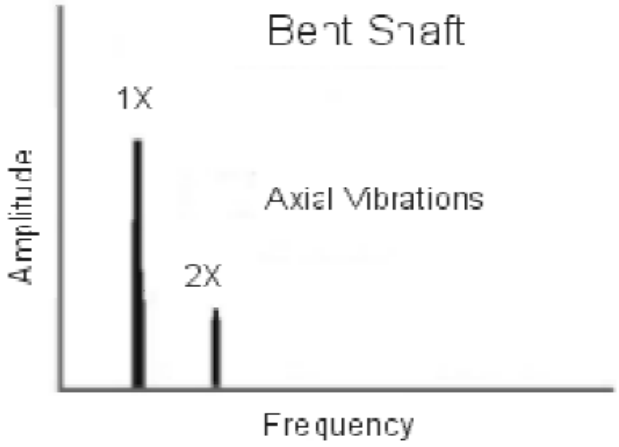


Figura 7 - Espectro de vibrações de um eixo empenado
Fonte: SCHEFFER, 2008

- Defeitos em rolamentos

As frequências de vibração geradas pelos elementos rolantes de um rolamento quando possuem ou passam por um defeito são determinadas pela velocidade de rotação da máquina e características geométricas do modelo do rolamento. Estão mostrados abaixo os cálculos feitos para determinar a frequência de cada defeito, segundo Graney (2011):

$$BPFI = \left(\frac{N}{2}\right) F \left(1 + \frac{B}{P}\cos\theta\right) \tag{4}$$

$$BPFO = \left(\frac{N}{2}\right) F \left(1 - \frac{B}{P}\cos\theta\right) \tag{5}$$

$$FTF = \frac{F}{2}\left(1 - \frac{B}{P}\cos\theta\right) \tag{6}$$

$$BPF = \left(\frac{P}{2B}\right) F \left[1 - \left(\frac{B}{P}\cos\theta\right)^2\right] \tag{7}$$

Onde:

BPFI = Frequência de defeito na pista interna (Hz)
BPFO = Frequência de defeito na pista externa (Hz)
FTF = Frequência da gaiola (Hz)
BPF = Frequência de defeito na esfera
N = Número de esferas
F = Frequência de rotação do eixo (Hz)

B = Diâmetro da esfera (mm)
P = Pitch diameter (mm)
Θ = Ângulo de contato

Com as fórmulas acima, é possível determinar as frequências de cada defeito e determinar a evolução da amplitude de cada frequência através de análises espectrais.

Figura 8 - Defeitos na pista interna de um rolamento
Fonte: http://www.machinerylubrication.com/Read/664/wear-bearings-gears

- Desalinhamento de rolamentos

Na presença de rolamentos cuja linha de centro não esteja alinhada com a de seu eixo, vibrações axiais consideráveis podem ser geradas. Análises espectrais feitas na direção axial indicam frequências de 1x, 2x e 3x RPM. Os ângulos de fase entre medições em lados opostos são de aproximadamente 180º, segundo Scheffer (2008).

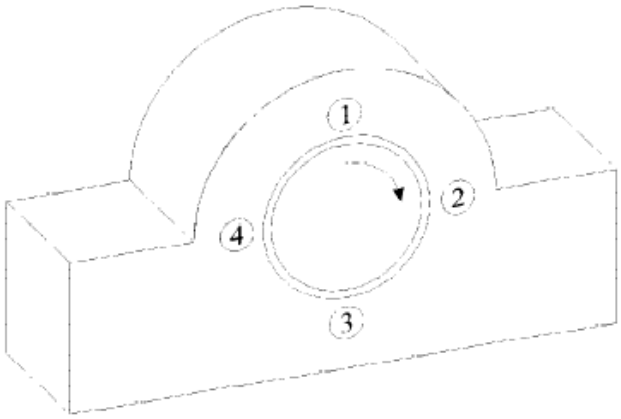


Figura 9 - Pontos de medição para verificação de desalinhamento de rolamentos
Fonte: CM5003 – SKF Vibration Diagnostic Guide

- Impelidores de bombas

Vibrações geradas pela rotação das pás são inerentes a bombas e ventiladores. Geralmente, não é destrutiva para a máquina, mas pode ser a causa de falha de outros componentes do sistema, caso haja vibrações excessivas geradas por problemas na distância entre as pás do impelidor e partes estacionárias, segundo Scheffer (2008). Esse é o sinal gerado por forças hidráulicas e aerodinâmicas mais comum, segundo SKF (2002) e a frequência de passagem das pás é calculada conforme a equação (8).

$$FPP = N_{pás}\,(RPM) \tag{8}$$

Onde:

FPP = Frequência de passagem das pás
$N_{pás}$ = Número de pás
RPM = Frequência de rotação da máquina

- Cavitação

O fenômeno da cavitação ocorre em bombas quando a pressão de entrada é reduzida de forma a atingir a pressão de vapor do fluido, formando bolhas de vapor. Ao passar pelo impelidor, sua pressão aumenta, fazendo com que essas bolhas implodam e causem ruído e vibração. Esse fenômeno é extremamente nocivo para a bomba e deve ser evitado. Segundo Scheffer (2008), o espectro de vibração causado por esse fenômeno é composto de uma banda de altas frequências, conforme mostrado na Figura 10.

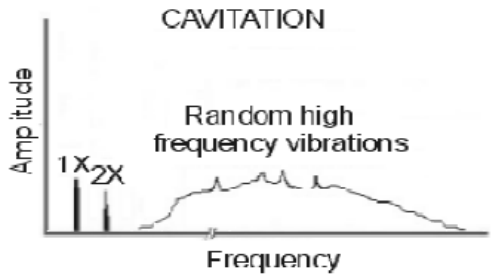


Figura 10 - Espectro de vibrações de uma bomba com cavitação
Fonte: SCHEFFER, 2008

- Defeitos em engrenagens

A frequência de contato entre os dentes das engrenagens do sistema equivale ao número de dentes de uma das engrenagens multiplicado pela velocidade de rotação do eixo na qual essa gira, conforme a equação (9):

$$GMF = N_{dentes}(RPM) \tag{9}$$

Onde:

GMF = Frequência de contato entre os dentes das engrenagens
N_{dentes} = Número de dentes da engrenagem
RPM = Velocidade de rotação do eixo

Vale observar que a frequência gerada por esse contato não ocorre apenas na *GMF*, mas no dobro ou triplo dessa. Na verdade, a frequência mais comum de ser visualizada é detectada em 3x *GMF*. Isso ocorre devido aos três movimentos de interação entre os dentes: deslizamento, rolamento e deslizamento na saída. Assim, é recomendável que a frequência máxima do espectro seja de 3,25 *GMF*, para visualização completa desse fenômeno.

3. Protótipo

3.1 Escolha dos componentes

3.1.1 Plataforma de prototipagem NodeMCU

O NodeMCU é, assim como o Arduino, uma plataforma de prototipagem eletrônica *open-source* que, além de permitir sua programação para utilização de portas analógicas e digitais, permite o desenvolvimento de projetos para a Internet das Coisas (IoT), por já possuir em seu hardware o módulo Wi-Fi ESP8266, que permite a conexão de microcontroladores com a internet.

A escolha do componente foi baseada de acordo com os seguintes parâmetros: preço com módulo Wi-Fi, capacidade de processamento, memória RAM e número de pinos GPIO (*General-Purpose Input/Output*). Comparando com as placas de desenvolvimento de capacidades similares, temos os dados da Tabela 3:

Tabela 3 - Comparação entre placas de desenvolvimento

Componente	Preço do componente	Acessórios	Processamento	Memória RAM	Número de pinos GPIO
Arduino Uno R3	USD 16,20	USD 6,71	16 MHz	2 KB	14
Raspberry Pi Zero W	USD 24,99	USD 7,57	1 GHz	512 MB	28
NodeMCU	USD 3,58	USD 0,00	80 ou 160 MHz	~ 50 KB	10

O Arduino Uno R3 é a placa mais barata da família Arduino e necessita de um *shield* Wi-Fi para o projeto em questão, enquanto o Raspberry Pi Zero W é a placa de menor preço da família Raspberry Pi com a funcionalidade Wi-Fi imbutida, porém necessita de um cartão Micro SD para sua programação. Os preços desses acessórios estão mostrados na Tabela 3 e todos os preços da tabela foram retirados do site AliExpress, no dia 18/12/2017. As especificações técnicas foram retiradas do manual de cada fabricante (ESPRESSIF, 2017), (ARDUINO, 2017) e (ADA, 2017). A memória RAM do ESP8266, CPU do NodeMCU, é dada por um valor aproximado no manual, já que este valor é dado quando o dispositivo está em *station mode* e conectado ao roteador.

O projeto deve conectar um acelerômetro, um módulo cartão SD e 3 LED´s. O primeiro, consome 2 pinos digitais (GPIO´s), caso seja usada a comunicação I2C. O segundo utiliza 4 pinos digitais e os LED´s, mais 3 pinos. Assim, o projeto precisará de 9 pinos no total, fazendo com que as três placas satisfaçam esse quesito. Como um cartão SD será conectado para armazenar os dados, a memória RAM será utilizada apenas para variáveis do código. Logo, o NodeMCU é a placa que atende às necessidades do projeto, com o menor preço.

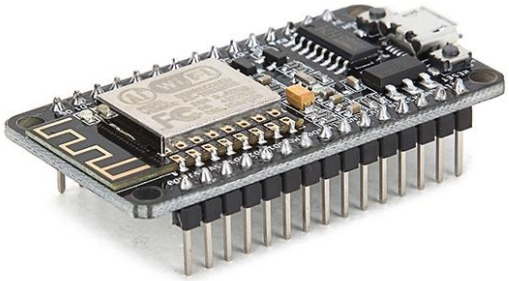


Figura 11 - Placa de desenvolvimento NodeMCU
Fonte: https://www.filipeflop.com/produto/modulo-wifi-esp8266-nodemcu-esp-12/

A programação do NodeMCU pode ser feita nas linguagens LUA ou Arduino. O código deste trabalho foi criado utilizando o software Arduino versão 1.8.3, já que essa linguagem dispõe de uma vasta fonte de referências na página oficial do programa.

3.1.2 Acelerômetro MPU-6050

O sensor MPU-6050 contém em seu chip um acelerômetro e giroscópio MEMS, que mede cada grandeza nos três eixos e possui um conversor analógico-digital de 16 bits. Assim, o sensor fornece dados em uma saída digital, entre uma das faixas de medições programáveis na tabela a seguir e sua comunicação pode ser feita pelos protocolos SPI ou I2C. Além disso, essa placa contém um sensor de temperatura, permitindo medições entre -40 e 85 graus (InvenSense, 2013a).

Tabela 4 - Especificações do acelerômetro MPU-6050

Especificação	Valor	Unidade
Frequência de aquisição máxima	1000	Hz
Frequência de aquisição utilizada	400	Hz
Faixas de medições	±2, ±4, ±8, ±16	g
Faixa de medição utilizada	±4	g
Resolução	16	bits
Fator de conversão utilizado	8192	LSB/g

A frequência de aquisição do projeto na Tabela 4 é definida simplesmente pelo *delay* entre um comando de medição e outro na estrutura do programa. A faixa de medição de -4g a +4g é escolhida ao enviarmos o dado hexadecimal 0x08 para o registro de memória 0x1C do acelerômetro pela comunicação I2C. O mapa de registros de memória

para alteração em configurações do dispositivo está disponível em InvenSense (2013b). Já o fator de conversão para a faixa de medição selecionada é utilizado para o cálculo da aceleração em m/s^2, já que os dados enviados correspondem ao valor em bits. A equação será explicada detalhadamente no Capítulo 4.

Figura 12 - Acelerômetro e giroscópio MPU-6050
Fonte: https://playground.arduino.cc/Main/MPU-6050

3.1.3 Módulo cartão SD

Este módulo permite, quando conectado a outros microcontroladores, a leitura e escrita em um cartão SD. O módulo aceita cartões formatados em FAT16 ou FAT32, alimentação de 3.3 ou 5V e utiliza a comunicação via interface SPI por meio dos pinos MOSI, SCK, MISO e CS, mostrados na Figura 13. O cartão SD utilizado foi o SanDisk Micro SDHC com adaptador, com 8 GB de memória.

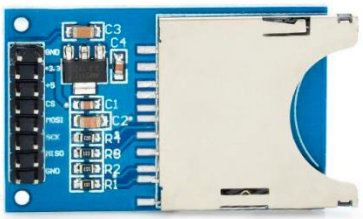


Figura 13 - Módulo cartão SD
Fonte: https://www.filipeflop.com/produto/modulo-cartao-sd-card/

3.2 Hardware

A comunicação do NodeMCU com o acelerômetro MPU-6050 pode ser feita de duas formas: via protocolo SPI (*Serial-Peripheral Interface*) ou I2C (*Inter-Integrated*

Circuit). A escolha de utilização do tipo de comunicação impacta diretamente o hardware do sistema. Assim, essa deve ser analisada para melhor utilização das portas disponíveis.

A comunicação I2C trabalha no modelo *master-slave*, onde, utilizando apenas duas portas do controlador (pinos SDA e SCL), podemos ter até 128 dispositivos *slaves* diferentes conectados, já que a comunicação I2C padrão reserva 7 bits de memória (NXP, 2014) para endereçamento *slave*. O modelo de conexão utilizando as mesmas linhas SDA e SCL para múltiplos dispositivos é mostrado na Figura 14:

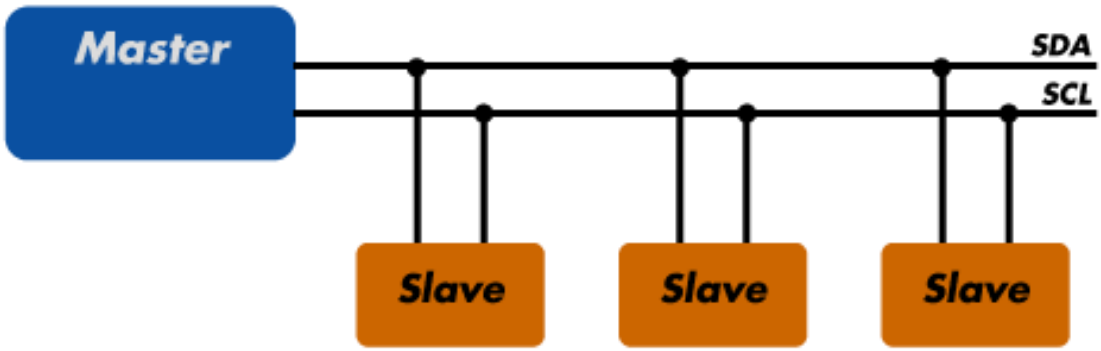


Figura 14 - Modo de conexão I2C
Fonte: www.arduinobr.com/arduino/i2c-protocolo-de-comunicacao

Vale ressaltar que os dispositivos *slaves* devem possuir endereços diferentes para que a comunicação possa ser feita corretamente. O acelerômetro MPU-6050 só pode ter dois endereços diferentes, determinados pelo fabricante pelos hexadecimais 0x68 e 0x69. Esses podem ser definidos pelo usuário através do nível recebido pela porta AD0 do acelerômetro (InvenSense, 2013a). Logo, apenas um acelerômetro a mais poderia ser adicionado ao projeto. Entretanto, outros dispositivos de modelos diferentes poderiam ser adicionados e nenhuma porta adicional do controlador precisaria ser utilizada.

A comunicação SPI, por outro lado, necessita de um pino SS (ou CS) diferente para cada dispositivo *slave* conectado. Devido à limitação de pinos no NodeMCU, já que os 9 pinos digitais já serão utilizados pelos componentes eletrônicos e LED´s, foi utilizada a comunicação I2C, caso seja necessário, para trabalhos futuros, inserir outro acelerômetro ou outros dispositivos. O módulo cartão SD utiliza o protocolo SPI, exclusivamente. Assim, a comunicação do NodeMCU com os dispositivos é feita com os pinos da Tabela 5, onde estão indicados os pinos do controlador que são específicos para receber ou enviar sinais dos componentes utilizados, definindo as restrições para a configuração de hardware:

Tabela 5 - Definição da pinagem do projeto

NodeMCU	Pino do componente externo	Componente	Restrição de Hardware
D0	+	Led amarelo	Não
D1	+	Led verde	Não
D2	CS	Módulo SD	Não
D3	SCL	Acelerômetro	Não
D4	SDA	Acelerômetro	Não
D5	SCK	Módulo SD	Sim
D6	MISO	Módulo SD	Sim
D7	MOSI	Módulo SD	Sim
D8	+	Led vermelho	Não

Os 3 LED´s servem para identificação no próprio local da situação do nível global de vibrações da máquina e possuem as cores verde, amarelo e vermelho. O valor desses é configurado em um arquivo do cartão SD, lido na inicialização do programa e serve como parâmetro de notificação. Assim, com a seleção dos componentes e pinos utilizados, temos a seguinte configuração de hardware da Figura 15, feita com o auxílio do software Fritzing:

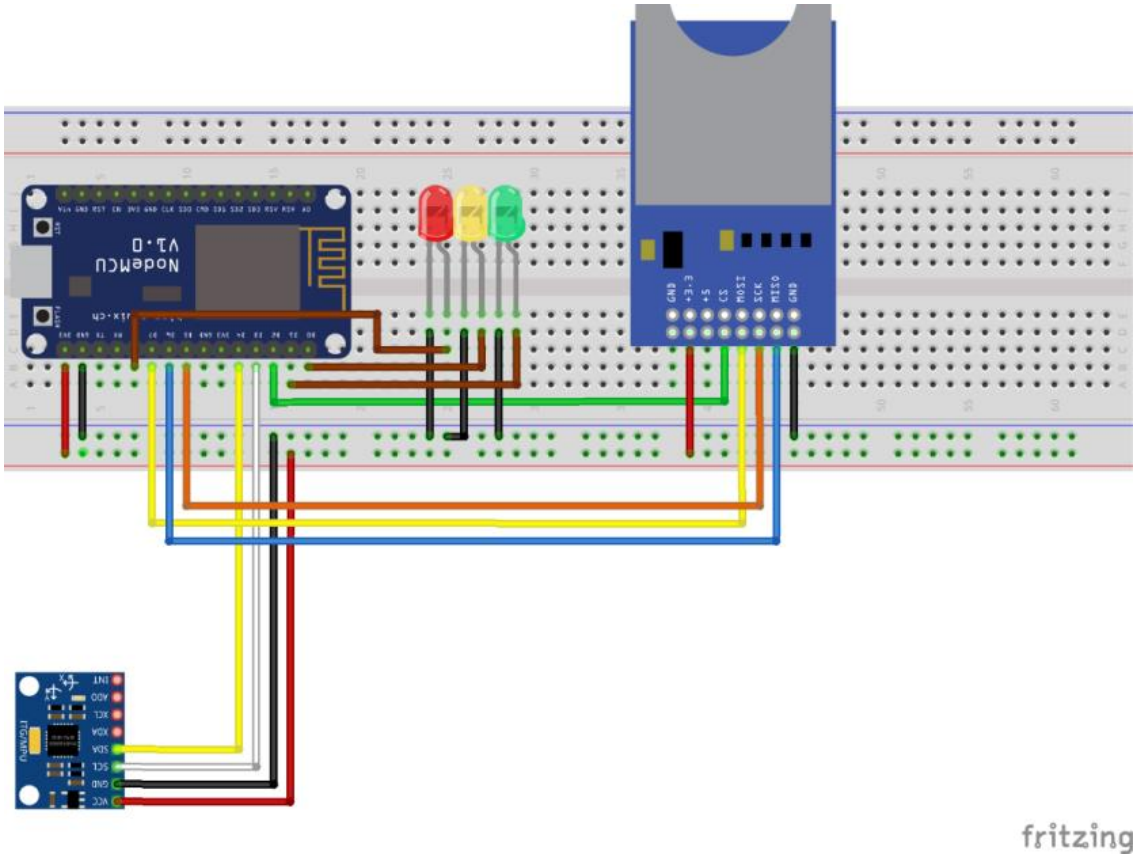


Figura 15 - Esquema eletrônico

3.3 Projeto dos encapsulamentos

O suporte do acelerômetro deve funcionar como uma base magnética, para que seja de fácil instalação, utilização e manutenção. Assim, o suporte deve conter um ímã, fixar o acelerômetro e protegê-lo de eventuais fatores externos que possam danificá-lo.

3.3.1 Base magnética

O mesmo foi fabricado em uma impressora 3D, devido ao seu tamanho, geometria e baixa densidade do material, quando comparado a materiais metálicos. O material utilizado pela impressora foi o PLA (ácido polilático). O encapsulamento é composto de uma base, onde o sensor é acomodado e o ímã é fixado, e uma tampa, que prende o sensor através de dois parafusos passantes, fixados por arruelas e porcas. O desenho técnico da base magnética está no Anexo III e o modelo está mostrado a seguir:

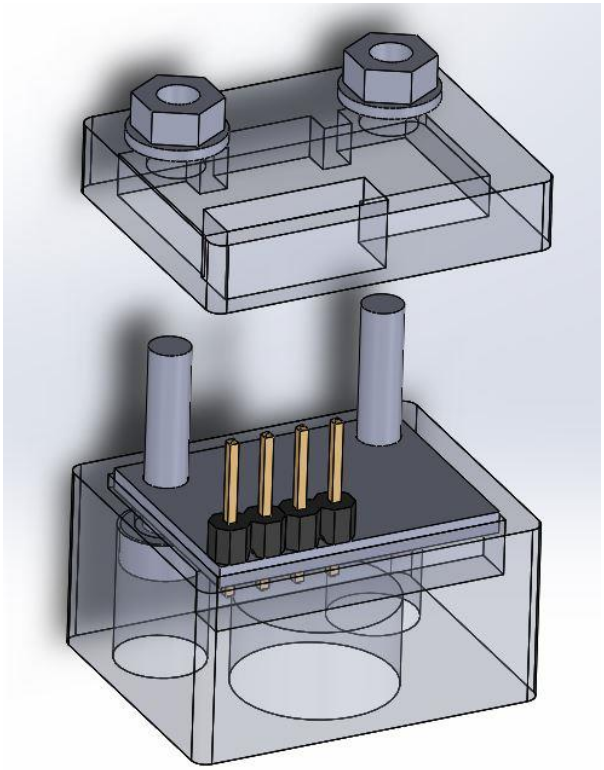


Figura 16 - Modelo da base magnética impressa com o acelerômetro MPU 6050

A tolerância dimensional da impressora 3D foi informada pelos usuários do equipamento, através de testes anteriores realizados no laboratório. Seu valor, disponível na Tabela 6, é maior que uma camada de filamento depositada, que possui valor de 1 mm, já que a contração do material influencia nas dimensões finais da peça. A tolerância de posicionamento do furo foi considerada como a metade da tolerância dimensional. Como o parafuso é passante, pode obter qualquer posição dentro do diâmetro funcional dos furos do suporte. Assim, o valor de sua tolerância de posicionamento é nulo.

Tabela 6 – Valores utilizados para o cálculo do furo funcional

	Diâmetro	Tolerância dimensional	Tolerância de posicionamento	Unidade
Furo do suporte	Di	0,35	0,18	mm
Parafuso	2,86	0,02	0,00	mm

Foi calculado o diâmetro dos furos de acordo com a Tabela 6, de forma à passagem dos parafusos pela base, sensor e tampa ser sempre possível. O cálculo do diâmetro funcional foi retirado de Marco Filho e Canabrava Filho (1996) e representada pela Equação (10):

$$D_i = D_e + 2TPO_e + 2TPO_i + Ff \quad (10)$$

Onde:

D_i = Diâmetro do furo
D_e = Diâmetro do parafuso
TPO_i = Tolerância de posicionamento do furo
TPO_e = Tolerância de posicionamento do parafuso

Assim, temos o valor de 3,59 mm para o diâmetro mínimo do furo, sem folgas funcionais. Para que não houvesse problemas de montagem, foi usada uma folga funcional de 0,31 mm. Assim, temos o valor utilizado para o diâmetro dos furos do suporte para a passagem do parafuso:

$$D_i = 3{,}9\ mm \quad (11)$$

Após a colagem do ìmã de neodímio na base com cola Super Bonder, foi obtido o resultado mostrado na Figura 17 e Figura 18:

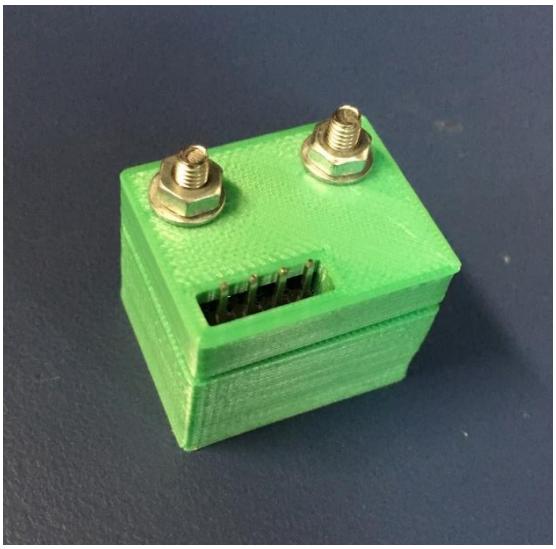


Figura 17 - Suporte do acelerômetro: topo

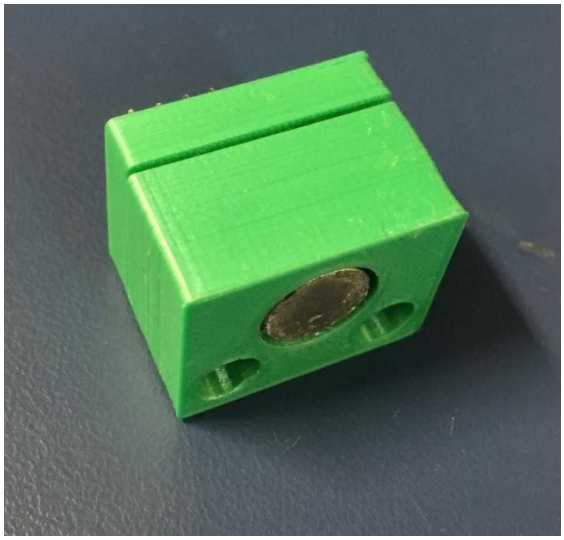


Figura 18 - Suporte do acelerômetro: base

3.3.2 Unidade de processamento

Por não necessitar de uma geometria especial, a unidade de processamento, que contém o módulo cartão SD, NodeMCU e LED´s, não necessitou de projeto de fabricação para o seu armazenamento. Assim, foi utilizada uma caixa eletrônica e os componentes citados foram fixados de forma a manterem sua funcionalidade. O interior da unidade pode ser vista na Figura 19. Para fixarmos a parte superior da caixa de modo a manter a funcionalidade das conexões, do cartão SD e visualização dos LED´s, foram feitos cortes na mesma com auxílio de uma serra e uma Dremel, além de furos para a fixação por interferência dos LED´s. O resultado está mostrado na Figura 20.

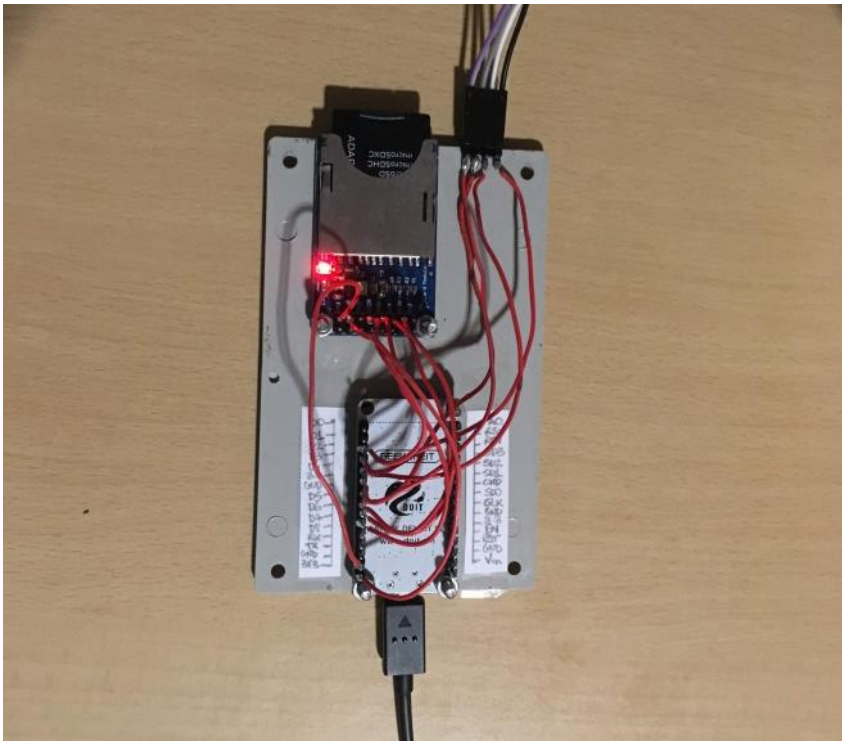


Figura 19 - Fixação dos componentes na unidade de processamento

Figura 20 - Unidade de processamento com LED´s

4 Software

Para a programação do NodeMCU, foi utilizado o software Arduino versão 1.8.3. Entretanto, a execução do código só pode ser realizada com sucesso caso sejam feitas algumas configurações nesse ambiente. Primeiramente, deve-se abrir a Arduino IDE e clicar em Arquivo > Preferências.

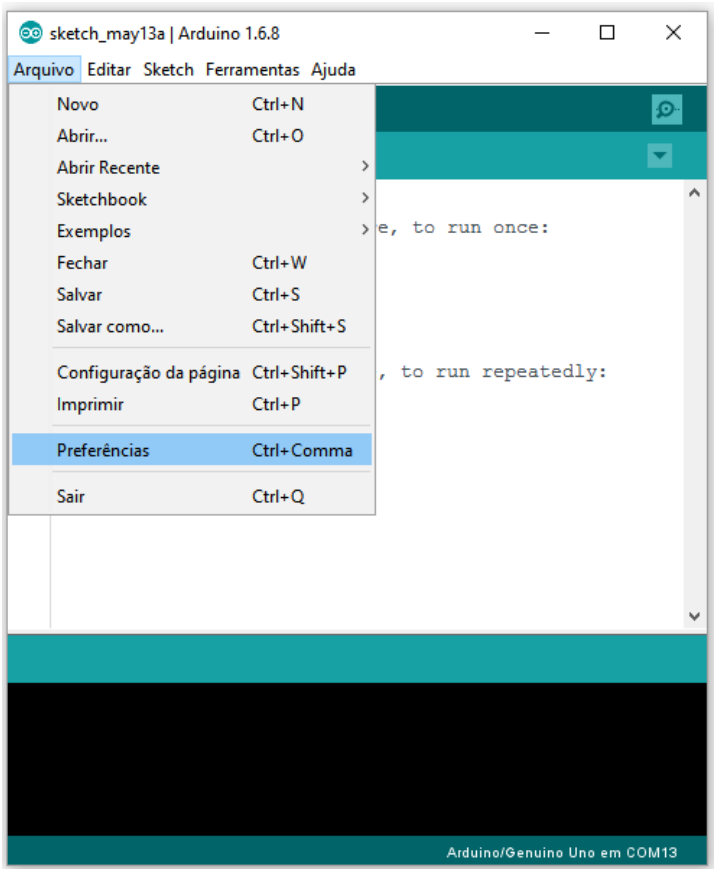


Figura 21 - Configuração da Arduino IDE: passo 1

Em seguida, deve ser digitado o *link* a seguir no campo “URLs adicionais de Gerenciadores de Placas” e inserir conforme a Figura 22:

<http://arduino.esp8266.com/stable/package_esp8266com_index.json>

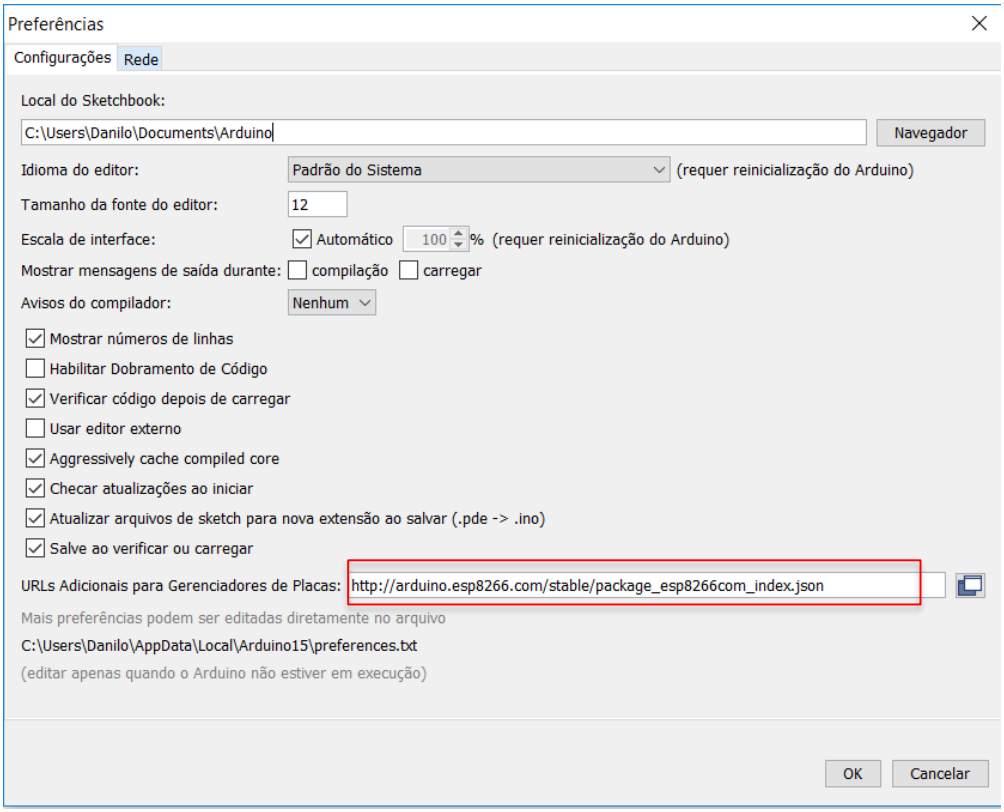


Figura 22 - Configuração da Arduino IDE: passo 2

Em seguida, deve-se clicar em “OK” para retornar à tela inicial e clicar em Ferramentas > Placa > Gerenciador de Placas:

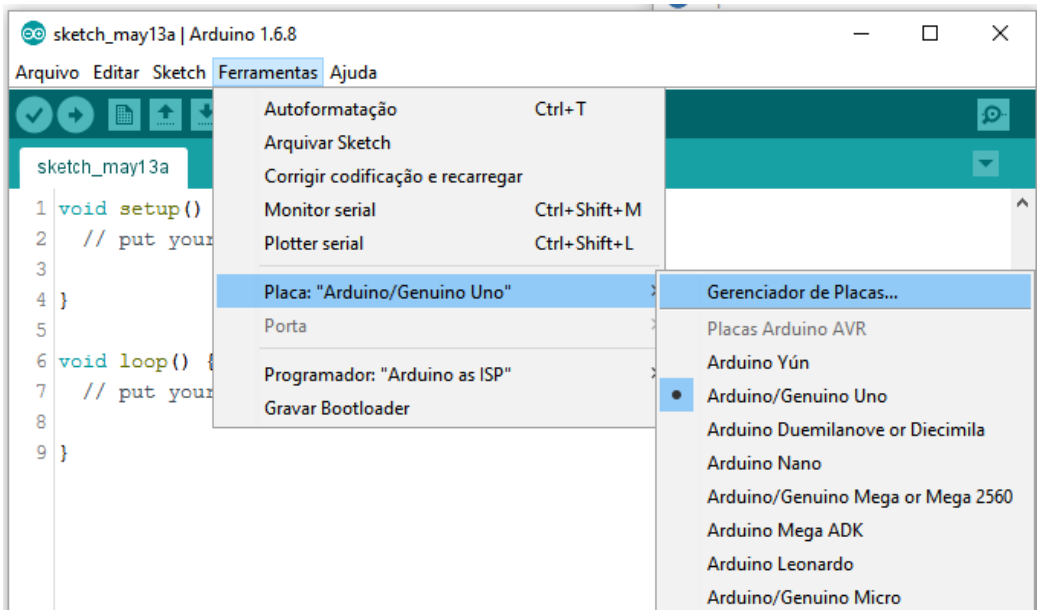


Figura 23 - Configuração da Arduino IDE: passo 3

Escolha a opção “esp8266 by ESP8266 Community” e clique em instalar para finalizar o processo. Depois desse passo, deve ser escolhida a placa adequada do NodeMCU em Ferramentas > Placa, além da respectiva porta COM na qual o dispositivo está conectado.

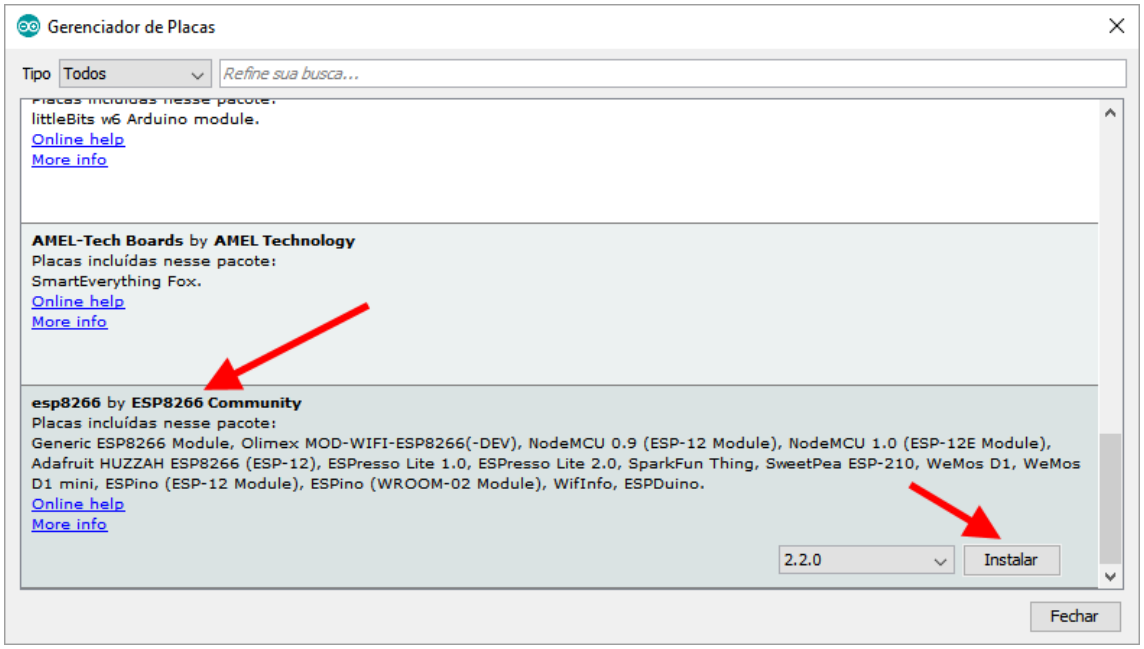


Figura 24 - Configuração da Arduino IDE: passo 4

Definidas as configurações necessárias, seguimos para a análise do código do sistema, que se encontra no Anexo II e consiste de três etapas: definição de bibliotecas e variáveis globais, inicialização (*setup*) e *loop*. A primeira consiste na importação das bibliotecas usadas pelo ESP8266 (módulo Wi-Fi embarcado no NodeMCU), cartão SD (comunicação SPI e funções do cartão SD) e acelerômetro MPU-6050 (comunicação SPI).

```
8  //BIBLIOTECAS
9  #include <SD.h> //Load SD card library
10 #include<SPI.h> //Load SPI Library
11 #include <Wire.h>    // imports the wire library for talking over I2C
12 #include <ESP8266WiFi.h> //Para mandar email
```

Figura 25 - Seção das bibliotecas no código

A segunda consiste na definição das variáveis globais, como variáveis auxiliares do código, variáveis das funções de e-mail e variáveis que remetem a endereços de

memória dos componentes e pinos. Em seguida, temos as funções auxiliares utilizadas no programa, descritas a seguir na

Tabela 7. Essas funções foram criadas com o propósito de deixar o código mais fácil de ser compreendido e de forma a simplificar a resolução de problemas na programação. Vale ressaltar que funções já existentes das bibliotecas da Figura 25 não foram mencionadas como funções auxiliares.

Tabela 7 - Funções auxiliares do programa

Função auxiliar	Descrição
I2C_Write	Escreve o dado no endereço de memória especificado do dispositivo *slave* pelo protocolo I2C
MPU6050_Init	Inicialização do acelerômetro e configuração de parâmetros operacionais, como a faixa de medição de acelerações
Read_RawValue	Leitura das acelerações em bits
sendEmail	Envia o email com base na cor (nível de vibrações) atingida
eRcv	Se o email ficar preso em alguma etapa de envio por 40 segundos, o envio é cancelado
readSDsettings	Lê os parâmetros de cores do cartão SD e utiliza a função applySetting
applySetting	Aplica os parâmetros do cartão SD nas variáveis do código
toFloat	Conversão de float para string utilizada na função applySetting

Uma funcionalidade essencial do protótipo, contida na função *sendEmail*, é o alerta por e-mail caso a máquina atinja um determinado nível de vibração. Para fazermos a comunicação do NodeMCU por e-mail, precisamos:

- Comunicação ativa com a rede wi-fi.
- Alterar a configuração da conta do Gmail do remetente para que o microcontrolador possa utilizá-lo. Essa etapa é necessária porque, por definição padrão, o Google utiliza métodos mais complexos de verificação além de endereço de e-mail e senha, somente. Para fazê-lo, acesse a página de sua conta do Google, clique em “apps com acesso à conta” e depois ative o botão de aplicativos menos seguros (Figura 26 e Figura 27, respectivamente).

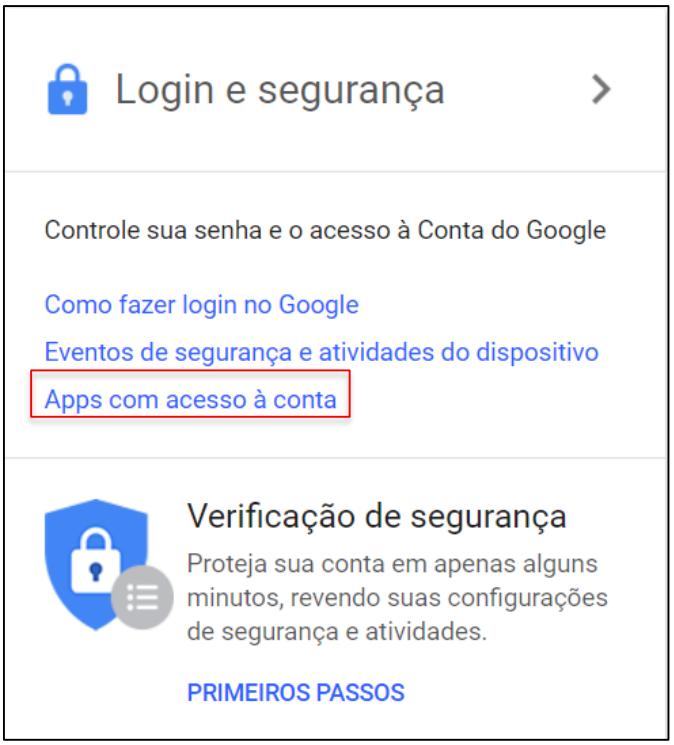


Figura 26 - Configuração do Gmail: passo 1

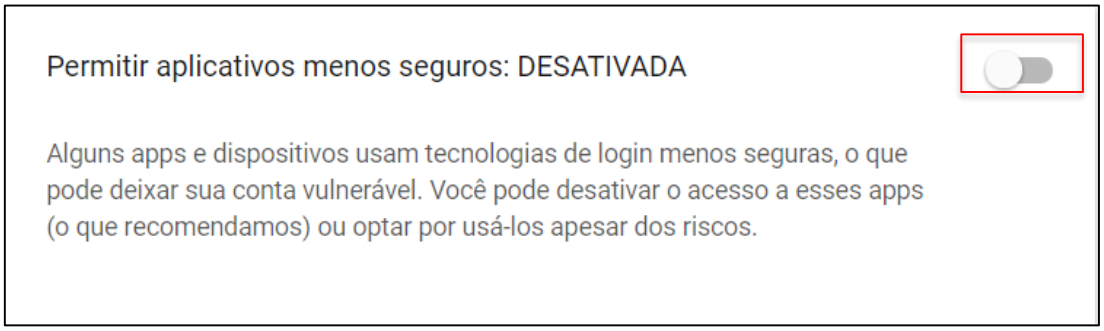


Figura 27 - Configuração do Gmail: passo 2

- Utilizar o site https://www.base64encode.org/ para converter o endereço de email e senha do remetente para a codificação Base64.
- O endereço de e-mail do destinatário.

Essas informações devem ser alteradas em cada projeto, para que as variáveis do Wi-Fi correspondam à configuração local e os e-mails correspondam aos contatos de interesse. A Figura 28 mostra a seção do código na qual deve ser feita essa alteração. As variáveis a serem alteradas se encontram nas linhas 27 a 32.

```
22 //VARIÁVEIS DO EMAIL
23 byte sendEmail();
24 byte eRcv(WiFiClientSecure client);
25 void efail(WiFiClientSecure client);
26
27 const char* SSID = "FAMILIA SOUSA"; // wifi ssid
28 const char* PASS = " "; // ssid password
29 const char* user_base64 = "aWNheHRvQGdtYWlsLmNvbQ=="; //conta do gmail codificada em base64
30 const char* user_password_base64 = " "; //senha do gmail codificada em base64
31 const char* from_email = "MAIL From:<icaxto@gmail.com>"; //email de quem envia
32 const char* to_email = "RCPT TO:<danilocsousa@poli.ufrj.br>"; //email de quem recebe
```

Figura 28 - Variáveis que devem ser alteradas no código para cada projeto

Outra função essencial é a de leitura de medições, *Read_RawValue.* Nela ocorre a comunicação com o MPU-6050, quando é solicitado o valor da medição em bits presente em um endereço de memória determinado do sensor, através da função *Wire.requestFrom().* Os endereços de memória correspondentes à cada eixo de medição estão mostrados na Tabela 8:

Tabela 8 – Endereços de memória dos dados de aceleração

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT[7:0]							

(INVENSENSE, 2013b)

Vale ressaltar que os 16 bits de dados de cada medição ficam separados em dois endereços de memória consecutivos, ou seja, 8 bits em cada. Para uni-los em uma mesma variável do código, devemos posicionar o os bits mais significativos à frente dos menos significativos para, assim, realizar a união. Essa nova variável é, então, convertida para um valor decimal inteiro.

A função *setup* é a segunda etapa do programa e o local onde são inseridas estruturas de código que devem ocorrer apenas uma vez, antes de iniciar-se a função *loop*. Nela, se iniciam as comunicações com os componentes, são definidas as condições de operação do acelerômetro pela função *MPU6050_Init*, a conexão com a internet pela função *WiFi.begin*, a definição de pinos digitais de saída para os LED´s e a leitura dos parâmetros de RMS no cartão SD que definem os limites amarelo e vermelho. Essa leitura é feita pela função *readSDsettings* e a forma de implementação no Arduino IDE pode ser vista na Figura 29.

```
266 void setup(){
267   Serial.begin(115200); //turn on serial monitor
268   Wire.begin(sda, scl);   //MPU
269   MPU6050_Init();
270   pinMode(10, OUTPUT); //Must declare 10 an output for SD
271   SD.begin(chipSelect); //Initialize the SD card reader
272   pinMode(5, OUTPUT);
273   pinMode(15, OUTPUT);
274   pinMode(16, OUTPUT);
275   readSDSettings();
276
277   // WI-FI
278   Serial.print("Connecting To ");
279   WiFi.begin(SSID, PASS);
280   while (WiFi.status() != WL_CONNECTED)
281   {
282     delay(500);
283     Serial.print(".");
284   }
285   Serial.println("");
286
287   Serial.println("WiFi Connected");
288   Serial.print("IP address: ");
289   Serial.println(WiFi.localIP());
290   delay(2000);
291
292 }
```

Figura 29 - Seção da função *setup* no código

Logo em seguida temos a função *loop*, que é a estrutura de código que se repete até que o dispositivo seja desligado ou resetado. Seu funcionamento ocorre da seguinte forma:

1. Definição das variáveis locais
2. Medição da aceleração em cada eixo
3. Média das acelerações
4. Novas medições, agora decrescidas da média anterior
5. Dados são enviados a um cartão SD
6. Média RMS
7. LED referente à faixa de vibrações é aceso
8. E-mail é enviado caso nível ultrapasse o limite estabelecido, porém isso ocorre apenas uma vez

Para a medição da aceleração, temos que converter os dados recebidos pelo acelerômetro em bits para m/s^2. Para isso, usamos a Equação (12) para cada eixo, utilizando o fator de conversão determinado pelo fabricante na para a faixa de medições escolhida de ±4g, mostrado anteriormente na Tabela 4, onde Ax é a medição no SI no eixo X e AccelX é a medição em bits:

$$Ax = \frac{9{,}81\ AccelX}{fator\ de\ conversão} \tag{12}$$

O fator de conversão é de 8192 LSB/g, mostrado anteriormente na Tabela 4.

Em seguida, fazemos a média ao adicionarmos cada medição relativa a um eixo em uma variável (somaX, por exemplo) e dividindo pelo número de pontos após o loop de medições terminar. Como a média RMS deve ser feita sem a medição da parcela da gravidade, são feitas novas medições decrescidas da média anterior. Isso é feito porque o programa não possui memória suficiente para armazenar todos os dados, fazer a média e subtraí-la de cada dado. A justificativa para a utilização da média do sinal anterior se dá pelo fato de que a média das acelerações é constante, utilizando a premissa de que o sensor vibra em um ponto médio constante ao ser fixo na máquina:

$$x_{médio} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{13}$$

Com essa premissa, fazemos a derivada segunda da posição para termos a aceleração média do sinal, onde a parcela constante da equação (14) é a parcela da gravidade a ser medida pelo sensor:

$$a_{média} = \frac{\sum_{i=1}^{n} \ddot{x}_i}{n} + g.cos(\theta) \tag{14}$$

A derivada do somatório pode ser escrita de acordo com a equação (15):

$$a_{média} = \frac{d^2}{dt^2}\left(\frac{\sum_{i=1}^{n} x_i}{n}\right) + g.cos(\theta) \tag{15}$$

Como a parcela entre parênteses na equação acima é a posição média do sensor, dada como constante segundo nossa premissa, a derivada segunda desse componente é nula e a aceleração média é dada conforme a equação (16):

$$a_{média} = g.cos(\theta) \tag{16}$$

Dessa forma, é possível fazer a média RMS do sinal, já que a parcela constante das acelerações foi retirada ao diminuirmos a média das acelerações de cada medição. Com isso, o código analisa se o RMS ultrapassou os níveis estabelecidos no arquivo “settings.txt” do cartão SD. Caso isso ocorra, a função *sendEmail* é acionada com a respectiva cor definida para aquela faixa de RMS. O LED referente à essa cor é aceso, informando visualmente a situação atual da máquina. Ao término desse processo, a função *loop* termina, e outro ciclo de medições e análises começa.

5. Resultados

O protótipo foi instalado em uma bancada de testes do LAVI, que consiste de um motor elétrico, um conjunto mecânico acoplado por engrenagens, um inversor de frequência e um sensor óptico conectado a um osciloscópio, conforme mostrado na Figura 30. O objetivo do teste consistiu em definir os limites de alarme por uma análise estatística com as engrenagens desacopladas e, em seguida, acoplar as engrenagens, de forma a aumentar o nível de vibrações, forçando o protótipo a alterar a cor do LED de indicação e enviar um e-mail de alerta. As medições foram feitas com uma rotação de 9 Hz.

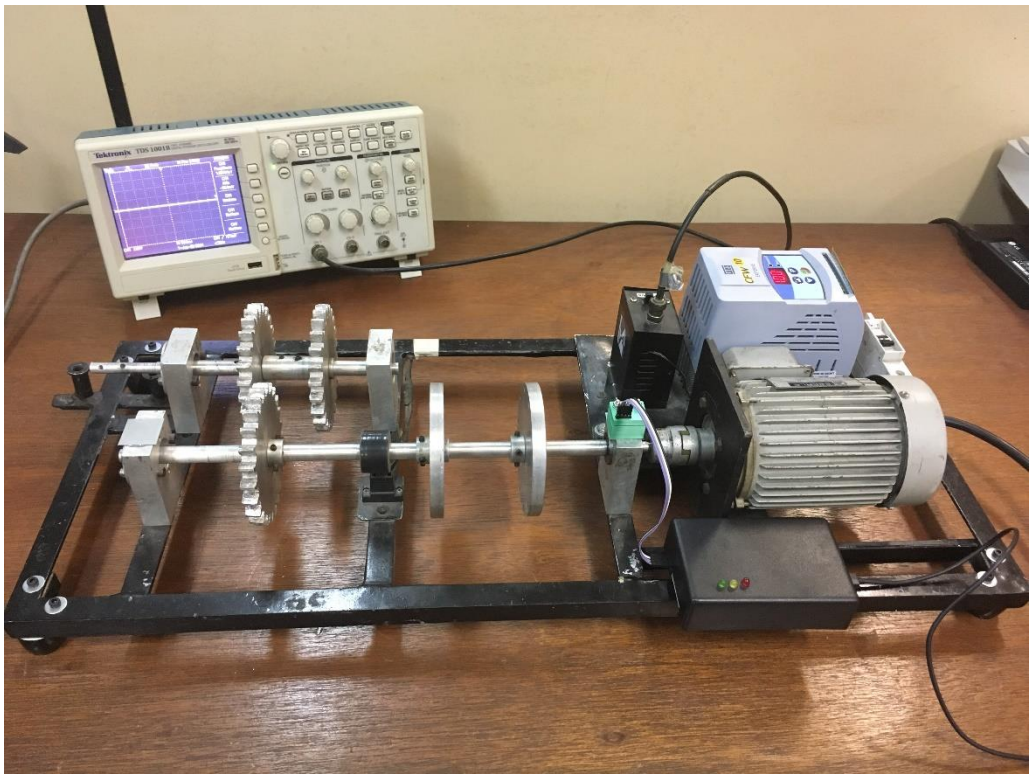


Figura 30 - Limites de alarme definidos com as engrenagens desacopladas

O limite de alarme amarelo foi definido de acordo com Budik *et al* (2016), que propõe um limite de alarme a uma distância de 3 desvios padrões da média. Já o limite de alarme vermelho, se encontra a uma distância de 6 desvios padrões. Como os dados de RMS gravados no cartão SD indicaram que o maior nível de vibrações se encontrava no eixo Y (direção axial), os limites foram estabelecidos com base nos dados de dessa orientação. O resultado está mostrado na Figura 31.

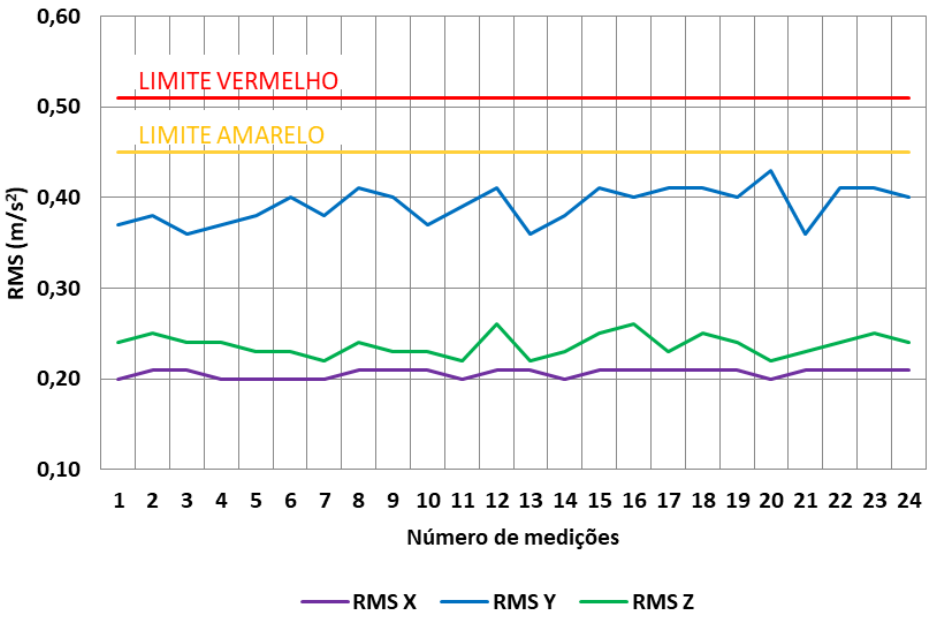


Figura 31 - Níveis de vibração para as engrenagens desacopladas

Além dos dados de RMS, o código do projeto também salva o último sinal medido no cartão de memória, mostrado na Figura 32, conforme os parâmetros da Tabela 4. Também é possível fazer uma FFT do sinal com o código disponível no Anexo I, na linguagem Python. Foi feita a FFT média de dois sinais medidos em instantes diferentes, de forma a diminuir o ruído do gráfico. O resultado está mostrado na Figura 33.

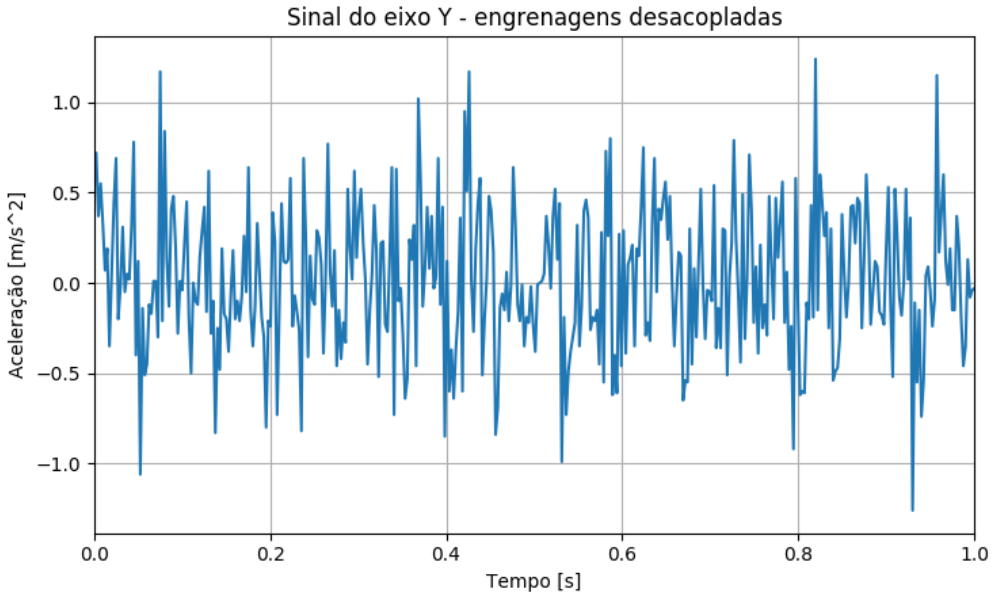


Figura 32 - Sinal armazenado no cartão SD

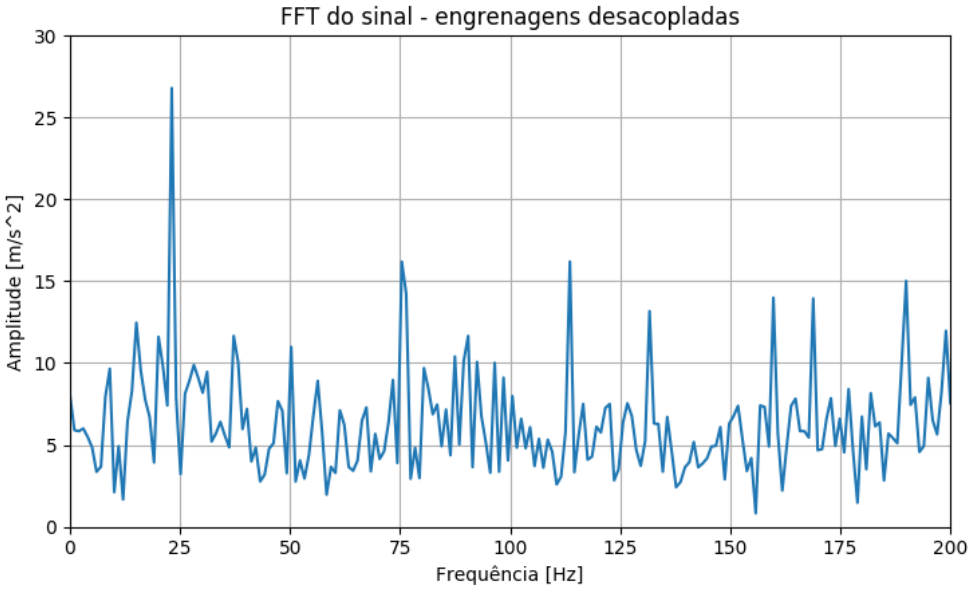


Figura 33 - FFT do eixo Y (direção axial)

Em seguida, foram acopladas as engrenagens da bancada, com o objetivo de introduzir mais vibração no sistema, conforme mostrado na Figura 34. Os dados de RMS gravados estão mostrados na Figura 35.

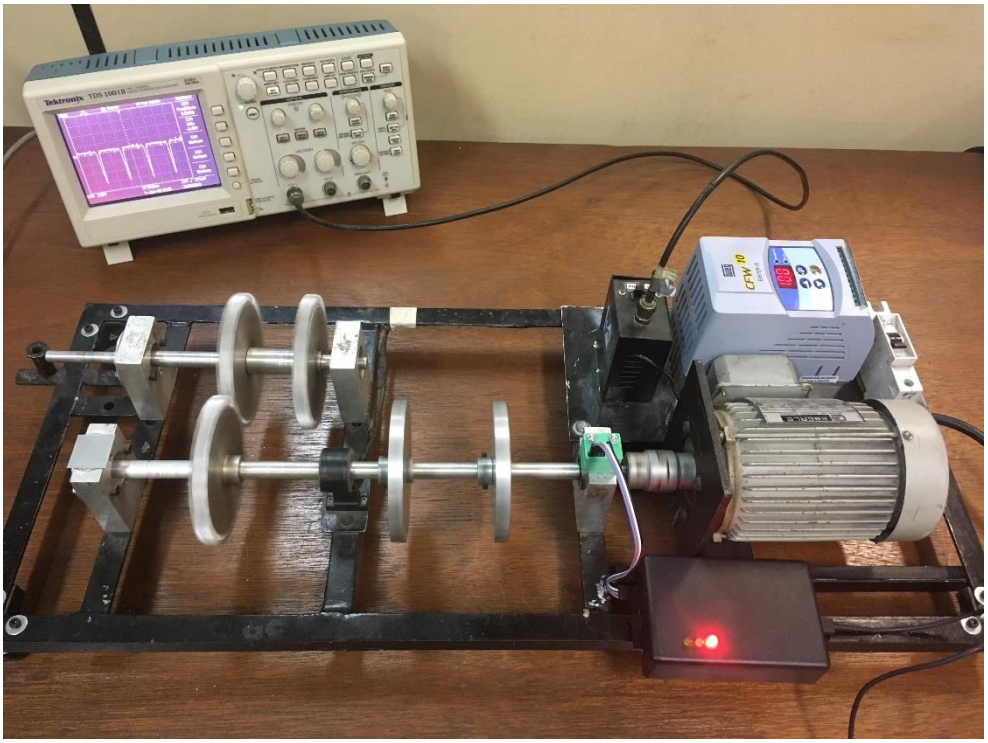


Figura 34 - Teste realizado com as engrenagens acopladas

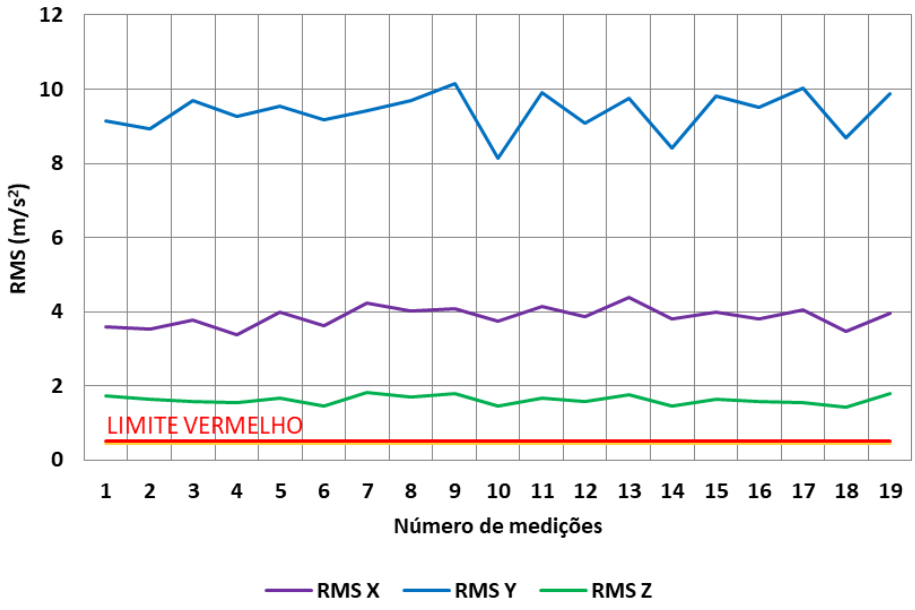


Figura 35 - Níveis de vibração para as engrenagens acopladas

Analisando a Figura 35, vemos que o limite vermelho de RMS definido anteriormente foi ultrapassado em todas as 19 medições. Nessa condição, o protótipo passou a indicar o estado da máquina ao acender o LED vermelho após cada análise, além de enviar um e-mail de alerta, mostrado na Figura 36. Vale ressaltar que apenas um e-mail é enviado após cada limite ser ultrapassado, com o objetivo de não lotar a caixa de entrada do usuário.

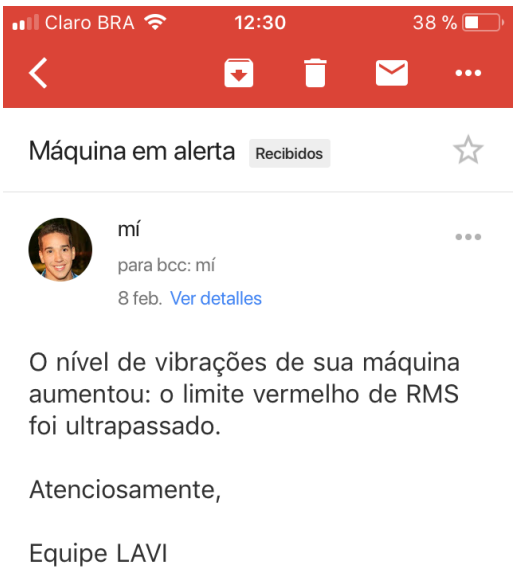


Figura 36 - E-mail de alerta

Além disso, foi verificado o tamanho médio em kilobytes de cada linha de RMS gravada no cartão SD, através da visualização do tamanho do arquivo “RMS.txt” e da contagem de linhas em diferentes instantes. Com isso, temos uma estimativa da capacidade máxima de linhas escritas por esse programa e, sabendo que este código escreve uma linha a cada 7 segundos, podemos estimar o tempo máximo que o protótipo pode funcionar sem que haja intervenção do usuário para limpar o arquivo de registro de RMS:

Figura 37 - Tamanho do arquivo por linha

Dessa forma, sabendo que o cartão SD tem espaço para 7932936 KB de dados e que cada linha consome em média 0,0161 KB de memória, através da Figura 37, o cartão SD preencheria completamente sua capacidade de memória após 111 anos de utilização contínua do protótipo.

6. Conclusão e trabalhos futuros

O projeto de monitoramento remoto de máquinas rotativas foi realizado com sucesso, já que atendeu aos objetivos de processamento de dados, monitoramento contínuo e notificação do usuário. A teoria utilizada para a configuração do nível de alerta do usuário é de aplicação viável para o projeto, necessitando apenas de medições iniciais para uma definição estatística das medições de vibrações da máquina rotativa a ser analisada. Uma análise em frequência das acelerações também pode ser realizada pelo usuário do protótipo, já que este registra as medições em um cartão SD. Outro fator de relevância é a capacidade de memória do cartão SD para sua utilização, que se mostrou suficiente para a aplicação proposta, já que o mesmo só teria sua memória cheia após 111 anos de utilização contínua.

Como trabalhos futuros, podem ser adicionados componentes como displays LCD pela comunicação I2C, para visualização no local do status da máquina e erros do protótipo, de modo a melhorar sua utilização. Uma melhoria no código também pode ser feita, para que todas as configurações iniciais sejam definidas no cartão. Além disso, pode ser feito um estudo de implementação de comunicação 3G para o protótipo, de modo a monitorar máquinas de difícil acesso ou localizações remotas. Um estudo de mercado também pode ser feito, de modo a analisar a viabilidade de transformar o protótipo deste trabalho em um produto comercial.

Referências bibliográficas

ADA, L., “Introducing the Raspberry Pi Zero”, 2017. Disponível em: < https://cdn-learn.adafruit.com/downloads/pdf/introducing-the-raspberry-pi-zero.pdf >. Acesso em: 12 dez. 2017.

ALBARBAR, A. *et al*, “Performance evaluation of MEMS accelerometers”, *Journal of the International Measurement Confederation,* vol 42 , no. 5 , pp. 790-795, 2008.

ALVAREZ, O.E., *Manual de manutenção planejada*, 1ª ed, João Pessoa, Universitária UFPB, 1988.

ARDUINO, “Arduino Uno Rev3”. Disponível em: https://store.arduino.cc/usa/arduino-uno-rev3. Acesso em: 12 dez. 2017.

BUDIK T., JANKOVYCH R., HAMMER M. “Operational limits in vibration diagnostics”. In: Jabłoński R., Brezina T. (eds) *Advanced Mechatronics Solutions. Advances in Intelligent Systems and Computing*, Vol. 393 pp. 13-18, Cham, 2016

CNI, “Indicadores CNI ISSN 2317-7330”, abril de 2016. Disponível em: < http://www.fiemt.com.br/arquivos/2282_30_05_-_sondagem_especial_industria_4.0.pdf > Acesso em: 15 dez. 2017.

ESPRESSIF, “ESP8266EX Datasheet”, Versão 5.7, 2017. Disponível em: < http://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf >. Acesso em: 12 dez. 2017.

MARCO FILHO F. d., CANABRAVA FILHO J. S., Apostila de Metrologia, Rio de Janeiro: Sub-Reitoria de Ensino de Graduação e Corpo Discente/SR-1, 1996.

GRANEY B. P., STARRY K., “Rolling Element Bearing Analysis”. *Materials Evaluation*, Vol. 70, No. 1, pp. 78-85, Jan 2012.

INVENSENSE, “MPU-6000 and MPU-6050 Product Specification Revision 3.4”, Agosto de 2013. Disponível em: < https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf >. Acesso em: 12 dez. 2017

INVENSENSE, “MPU-6000/MPU-6050 Register Map and Descriptions”, Agosto de 2013. Disponível em: < https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf >. Acesso em: 24 jan. 2018

NXP, “I²C-bus Specification and user manual, Version 6.0”, abril de 2014. Disponível em: < https://www.nxp.com/docs/en/user-guide/UM10204.pdf >. Acesso em: 02 jan. 2018.

SCHEFFER, C., “Pump Condition Monitoring Through Vibration Analysis”. *Pumps: Maintenance, Design, and Reliability Conference*, 2008.

THANAGASUNDRAM, S., SCHLINDWEIN F. S., “Comparison of integrated micro-electrical-mechanical system and piezoelectric accelerometers for machine condition monitoring”. *Journal of Mechanical Engineering Science*, Vol. 220, Part C, pp. 1135-1146, Abr 2006.

MAIS, J., SKF, “Spectrum Analysis – The Key Features of Analysing Spectra” Maio de 2002. Disponível em: < http://www.skf.com/binary/tcm:12-113997/CM5118%20EN%20Spectrum%20Analysis.pdf >. Acesso em: 20 dez. 2017

Anexo I – Código da FFT em Python

```
"""
Created on 2017

Importando dados para o Python e fazendo a FFT do sinal

@author: Danilo
"""
#BIBLIOTECAS UTILIZADAS
import matplotlib.pylab as plt
import numpy as np
import scipy.fftpack

#ABRINDO O ARQUIVO E IMPORTANDO DADOS:
arq = open('sinal1.txt', 'r')                # mudar o nome do
arquivo aqui
texto = arq.readlines()
ndados=0
sinal1=[]

for linha in texto :
    sinal1.append(linha)
    ndados=ndados+1                          # número de dados
arq.close()


#TEMPO
f=500               #[Hz]
T=1/f               #[s]
tempo=T*ndados
t=np.linspace(0,tempo,ndados)

#PLOTANDO SINAL
plt.figure()
plt.plot(t,sinal1)
plt.title("Sinal 1")
plt.grid()
plt.xlabel("Tempo [s]")
plt.ylabel("Aceleração [m/s^2]")

#FFT DO SINAL
fft= scipy.fftpack.fft(sinal1)
amp_fft=np.abs(fft)
f=np.linspace(0,f/2,ndados/2)

#GRÁFICO DA FFT
plt.figure()
plt.plot(f,amp_fft[0:ndados/2])
plt.title("FFT do sinal 1")
plt.grid()
plt.xlabel("Frequência [Hz]")
plt.ylabel("Amplitude")
```

Anexo II – Código do projeto

```
/*
 *  PROJETO FINAL: DANILO CASTOR DE SOUSA
 *  DEPARTAMENTO DE ENGENHARIA MECÂNICA - UFRJ
 *  danilocsousa@poli.ufrj.br
 *  01/2018
*/

//BIBLIOTECAS
#include <SD.h> //Load SD card library
#include<SPI.h> //Load SPI Library
#include <Wire.h>    // imports the wire library for talking over
I2C
#include <ESP8266WiFi.h> //Para mandar email

//VARIÁVEIS DO CÓDIGO
int Npontos = 200;
int contadorRED=0;
int contadorYEL=0;
File settingsFile;
float RMSamarelo = 0.5; // valor só é utlizado caso não seja
possível ler settings.txt
float RMSvermelho = 0.8; // valor só é utlizado caso não seja
possível ler settings.txt

//VARIÁVEIS DO EMAIL
byte sendEmail();
byte eRcv(WiFiClientSecure client);
void efail(WiFiClientSecure client);

//const char* SSID = "FAMILIA SOUSA"; // wifi ssid
//const char* PASS = "standup18"; // ssid password
const char* SSID = "wLAVI1";
const char* PASS = "lavi-i130";
const char* user_base64 = "aWNheHRvQGdtYWlsLmNvbQ=="; //your gmail
account in base64 - https://www.base64encode.org/
const char* user_password_base64 = "YmVsaXNzaW1heA=="; //your gmail
password in base64
const char* from_email = "MAIL From:<icaxto@gmail.com>"; //dummy
from email
const char* to_email = "RCPT TO:<danilocsousa@poli.ufrj.br>";
//your gmail account to sent to

//VARIÁVEIS DO MPU6050
// MPU6050 Slave Device Address
const uint8_t MPU6050SlaveAddress = 0x68;

// Select SDA and SCL pins for I2C communication
const uint8_t scl = D3;
const uint8_t sda = D4;

// sensitivity scale factor respective to full scale setting
provided in datasheet
//const uint16_t AccelScaleFactor = 16384; // +- 2g
const uint16_t AccelScaleFactor = 8192; // +- 4g
//const uint16_t AccelScaleFactor = 4096; // +- 8g
```

```
//const uint16_t AccelScaleFactor = 2048; // +- 16g

const uint16_t GyroScaleFactor = 131;

// MPU6050 few configuration register addresses
const uint8_t MPU6050_REGISTER_SMPLRT_DIV   =  0x19;
const uint8_t MPU6050_REGISTER_USER_CTRL    =  0x6A;
const uint8_t MPU6050_REGISTER_PWR_MGMT_1   =  0x6B;
const uint8_t MPU6050_REGISTER_PWR_MGMT_2   =  0x6C;
const uint8_t MPU6050_REGISTER_CONFIG       =  0x1A;
const uint8_t MPU6050_REGISTER_GYRO_CONFIG  =  0x1B;
const uint8_t MPU6050_REGISTER_ACCEL_CONFIG =  0x1C;
const uint8_t MPU6050_REGISTER_FIFO_EN      =  0x23;
const uint8_t MPU6050_REGISTER_INT_ENABLE   =  0x38;
const uint8_t MPU6050_REGISTER_ACCEL_XOUT_H =  0x3B;
const uint8_t MPU6050_REGISTER_SIGNAL_PATH_RESET  = 0x68;

int16_t AccelX, AccelY, AccelZ, Temperature;

//VARIÁVEIS DO CARTAO SD
int chipSelect = 4; //chipSelect pin for the SD card Reader
File ACCEL;
File RMSFile;

// -------------------------------- FUNÇÃO DO MPU
void I2C_Write(uint8_t deviceAddress, uint8_t regAddress, uint8_t
data){
  Wire.beginTransmission(deviceAddress);
  Wire.write(regAddress);
  Wire.write(data);
  Wire.endTransmission();
}

// -------------------------------- FUNÇÃO DO MPU: configure
MPU6050
void MPU6050_Init(){
  delay(150);
  I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_SMPLRT_DIV,
0x07);
  I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_PWR_MGMT_1,
0x01);
  I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_PWR_MGMT_2,
0x00);
  I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_CONFIG, 0x00);
  I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_GYRO_CONFIG,
0x00);//set +/-250 degree/second full scale

  //I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_ACCEL_CONFIG,
0x00);// set +/- 2g full scale
  I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_ACCEL_CONFIG,
0x08);// set +/- 4g full scale
  //I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_ACCEL_CONFIG,
0x10);// set +/- 8g full scale
  //I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_ACCEL_CONFIG,
0x18);// set +/- 16g full scale

  I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_FIFO_EN, 0x00);
```

```
  I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_INT_ENABLE,
0x01);
  I2C_Write(MPU6050SlaveAddress,
MPU6050_REGISTER_SIGNAL_PATH_RESET, 0x00);
  I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_USER_CTRL, 0x00);
}

// -------------------------------- FUNÇÃO DO MPU: read all 14
register
void Read_RawValue(uint8_t deviceAddress, uint8_t regAddress){
  Wire.beginTransmission(deviceAddress);
  Wire.write(regAddress);
  Wire.endTransmission();
  Wire.requestFrom(deviceAddress, (uint8_t)8);
  AccelX = ~((((int16_t)Wire.read()<<8) | Wire.read())-1);
  AccelY = ~((((int16_t)Wire.read()<<8) | Wire.read())-1);
  AccelZ = ~((((int16_t)Wire.read()<<8) | Wire.read())-1);
  Temperature = (((int16_t)Wire.read()<<8) | Wire.read());
  //GyroX = ~((((int16_t)Wire.read()<<8) | Wire.read())-1);
  //GyroY = ~((((int16_t)Wire.read()<<8) | Wire.read())-1);
  //GyroZ = ~((((int16_t)Wire.read()<<8) | Wire.read())-1);
}

// -------------------------------- FUNÇÃO DO EMAIL: ENVIAR EMAIL
byte sendEmail(String Cor) //sending email function
{
  WiFiClientSecure client;

  if (client.connect("smtp.gmail.com", 465) == 1) {
    Serial.println(F("connected"));
  } else {
    Serial.println(F("connection failed"));
    return 0;
  }
  if (!eRcv(client)) return 0;
  Serial.println(F("--- Sending EHLO")); client.println("EHLO
1.2.3.4"); if (!eRcv(client)) return 0;
  Serial.println(F("--- Sending login")); client.println("AUTH
LOGIN"); if (!eRcv(client)) return 0;
  Serial.println(F("--- Sending User base64"));
client.println(user_base64); if (!eRcv(client)) return 0;
  Serial.println(F("--- Sending Password base64"));
client.println(user_password_base64); if (!eRcv(client)) return 0;
  Serial.println(F("--- Sending From"));
client.println(from_email); if (!eRcv(client)) return 0;
  Serial.println(F("--- Sending To")); client.println(to_email); if
(!eRcv(client)) return 0;
  Serial.println(F("--- Sending DATA")); client.println(F("DATA"));
if (!eRcv(client)) return 0;
  client.println(F("Subject: Máquina em alerta\r\n"));
  String msg1 = "O nível de vibrações de sua máquina aumentou: o
limite ";
  String msg2 = " de RMS foi ultrapassado.\n";
  String msg3 = msg1 + Cor + msg2;
  client.println(msg3);
  client.println(F("Atenciosamente,\n"));
  client.println(F("Equipe LAVI"));
  client.println(F("."));
```

```
  if (!eRcv(client)) return 0;
  Serial.println(F("--- Sending QUIT"));
  client.println(F("QUIT"));
  if (!eRcv(client)) return 0;
  client.stop();
  Serial.println(F("disconnected"));
  Serial.println(F("Email enviado!"));
  return 1;
}

// -------------------------------- FUNÇÃO DO EMAIL
byte eRcv(WiFiClientSecure client)
{
  byte respCode;
  byte thisByte;
  int loopCount = 0;

  while (!client.available()) {
    delay(1);
    loopCount++;
    // if nothing received for 40 seconds, timeout
    if (loopCount > 40000) {
      client.stop();
      Serial.println(F("\r\nTimeout"));
      return 0;
    }
  }
  respCode = client.peek();
  while (client.available())
  {
    thisByte = client.read();
    Serial.write(thisByte);
  }
  if (respCode >= '4')
  {
    //  efail();
    return 0;
  }
  return 1;
}

// -------------------------------- FUNÇÃO DE LEITURA DOS
PARÂMETROS DE RMS
 void readSDSettings(){
   char character;
   String settingName;
   String settingValue;
   settingsFile = SD.open("settings.txt");
   if (settingsFile) {
     while (settingsFile.available()) {
       character = settingsFile.read();
       while((settingsFile.available()) && (character != '[')){
        character = settingsFile.read();
       }
       character = settingsFile.read();
       while((settingsFile.available()) && (character != '=')){
         settingName = settingName + character;
         character = settingsFile.read();
```

```
        }
        character = settingsFile.read();
        while((settingsFile.available()) && (character != ']')){
          settingValue = settingValue + character;
          character = settingsFile.read();
        }
        if(character == ']'){

        //Debuuging Printing
        Serial.print("Name:");
        Serial.println(settingName);
        Serial.print("Value :");
        Serial.println(settingValue);

        // Apply the value to the parameter
        applySetting(settingName,settingValue);
        // Reset Strings
        settingName = "";
        settingValue = "";
        }
      }
      // close the file:
      settingsFile.close();
    } else {
    // if the file didn't open, print an error:
    Serial.println("error opening settings.txt");
    }
 }

// -------------------------------- FUNÇÃO DE APLICAÇÃO DOS
PARÂMETROS DE RMS
 void applySetting(String settingName, String settingValue) {
   if(settingName == "RMSamarelo") {
     RMSamarelo=toFloat(settingValue);
     Serial.println(RMSamarelo);
   }
   if(settingName == "RMSvermelho") {
     RMSvermelho=toFloat(settingValue);
     Serial.println(RMSvermelho);
   }
 }

// -------------------------------- FUNÇÃO DE CONVERSÃO: STRING
PARA FLOAT
 float toFloat(String settingValue){
   char floatbuf[settingValue.length()+1];
   settingValue.toCharArray(floatbuf, sizeof(floatbuf));
   float f = atof(floatbuf);
   return f;
 }


// -------------------------------- SETUP -------------------------
----- //

void setup(){
  Serial.begin(115200); //turn on serial monitor
  Wire.begin(sda, scl);   //MPU
```

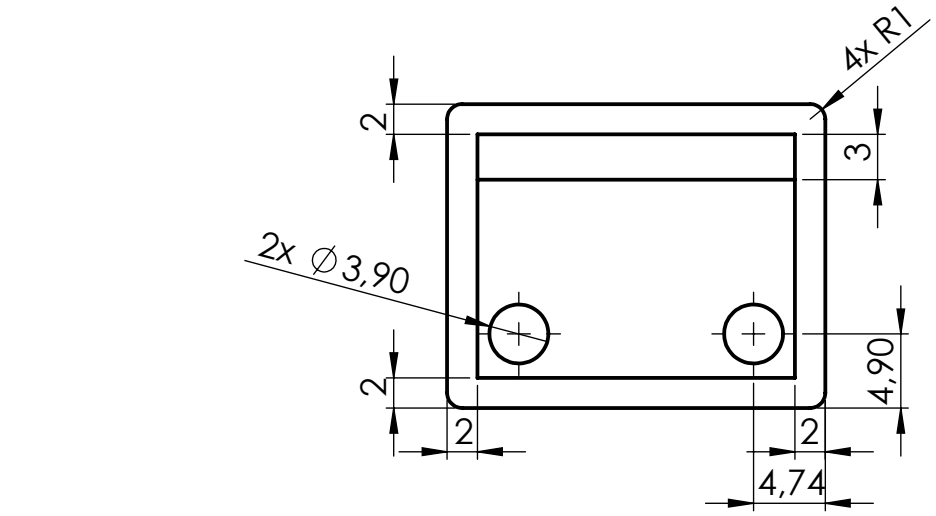
```
  MPU6050_Init();
  pinMode(10, OUTPUT); //Must declare 10 an output and reserve in
order for the SD library to work
  SD.begin(chipSelect); //Initialize the SD card reader
  pinMode(5, OUTPUT);
  pinMode(15, OUTPUT);
  pinMode(16, OUTPUT);
  readSDSettings();

  // WI-FI
  Serial.print("Connecting To ");
  WiFi.begin(SSID, PASS);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");

  Serial.println("WiFi Connected");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  delay(2000);

}

// ------------------------------- LOOP ---------------------------
---- //

void loop() {

  float valX = 0;
  float somaX = 0;
  float mediaX = 0;
  float soma_quadX = 0;
  float rmsX = 0;

  float valY = 0;
  float somaY = 0;
  float mediaY = 0;
  float soma_quadY = 0;
  float rmsY = 0;

  float valZ = 0;
  float somaZ = 0;
  float mediaZ = 0;
  float soma_quadZ = 0;
  float rmsZ = 0;

  // DADOS PARA A MÉDIA
  Serial.println("MEDIÇÕES");
  for(int k=0;k<Npontos;k++){
    double Ax, Ay, Az;
    Read_RawValue(MPU6050SlaveAddress,
MPU6050_REGISTER_ACCEL_XOUT_H);

    //divide each with their sensitivity scale factor
    Ax = (double)AccelX*9.81/AccelScaleFactor;
```

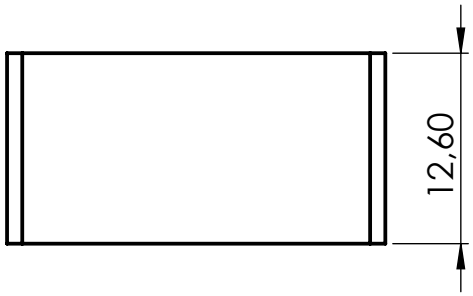
```
    Ay = (double)AccelY*9.81/AccelScaleFactor;
    Az = (double)AccelZ*9.81/AccelScaleFactor;

    Serial.println(Ax);
    somaX = somaX + Ax;
    somaY = somaY + Ay;
    somaZ = somaZ + Az;
    delay(5);
  }

  // MÉDIA
  Serial.println("MÉDIA");
  mediaX = somaX/Npontos;
  mediaY = somaY/Npontos;
  mediaZ = somaZ/Npontos;
  Serial.println(mediaX);

  // VALOR - MEDIA E FAZ SOMA DOS QUADRADOS
  somaX=0; //só para fazer o teste de próxima media = 0
  somaY=0; //só para fazer o teste de próxima media = 0
  somaZ=0; //só para fazer o teste de próxima media = 0

  Serial.println("VALOR - MEDIA:");
  for(int k=0;k<Npontos;k++){
    double Ax, Ay, Az;
    Read_RawValue(MPU6050SlaveAddress,
MPU6050_REGISTER_ACCEL_XOUT_H);

    //divide each with their sensitivity scale factor
    Ax = (double)AccelX*9.81/AccelScaleFactor;
    Ay = (double)AccelY*9.81/AccelScaleFactor;
    Az = (double)AccelZ*9.81/AccelScaleFactor;

    ACCEL = SD.open("ACCEL.txt", FILE_WRITE);
    valX = Ax - mediaX;
    valY = Ay - mediaY;
    valZ = Az - mediaZ;
    if (ACCEL) {
      ACCEL.print(valX);
      ACCEL.print(" ");
      ACCEL.print(valY);
      ACCEL.print(" ");
      ACCEL.println(valZ);
      Serial.println(valX);
    }
    ACCEL.close();
    somaX = somaX + valX;
    somaY = somaY + valY;
    somaZ = somaZ + valZ;
    soma_quadX = soma_quadX + pow(valX, 2);
    soma_quadY = soma_quadY + pow(valY, 2);
    soma_quadZ = soma_quadZ + pow(valZ, 2);
    delay(5);
  }

  //MÉDIA PARA CONFERIR SE DEU ZERO
  Serial.println("MÉDIA PARA CONFERIR SE DEU ZERO:");
  mediaX = somaX/Npontos;
```

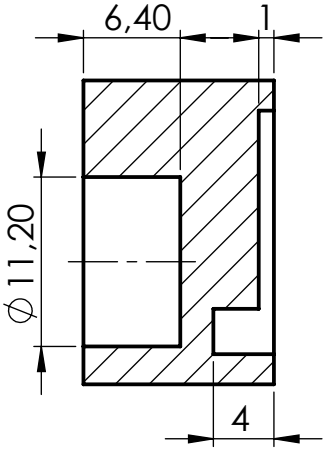
```
  mediaY = somaY/Npontos;
  mediaZ = somaZ/Npontos;
  Serial.println(mediaX);
  Serial.println(mediaY);
  Serial.println(mediaZ);

  //RMS
  Serial.println("RMS");
  soma_quadX = soma_quadX/Npontos;
  soma_quadY = soma_quadY/Npontos;
  soma_quadZ = soma_quadZ/Npontos;
  rmsX = pow(soma_quadX, 0.5);
  rmsY = pow(soma_quadY, 0.5);
  rmsZ = pow(soma_quadZ, 0.5);
  RMSFile = SD.open("RMS.txt", FILE_WRITE);
  if (RMSFile) {
    RMSFile.print(rmsX);
    RMSFile.print(" ");
    RMSFile.print(rmsY);
    RMSFile.print(" ");
    RMSFile.println(rmsZ);
    Serial.println(rmsX);
    Serial.println(rmsY);
    Serial.println(rmsZ);
  }
  RMSFile.close();

  //AÇÕES: ENVIAR EMAIL CASO NÍVEL AUMENTE
  if (rmsX<RMSamarelo and rmsY<RMSamarelo and rmsZ<RMSamarelo){
    digitalWrite(5, HIGH);
  }
  if ((rmsX>RMSamarelo and rmsX<RMSvermelho)||(rmsY>RMSamarelo and
rmsY<RMSvermelho) || (rmsZ>RMSamarelo and rmsZ<RMSvermelho)){
    digitalWrite(16, HIGH);
    if (contadorYEL==0){
      contadorYEL = 1;
      sendEmail("amarelo");
    }
  }
  if (rmsX>RMSvermelho || rmsY>RMSvermelho || rmsZ>RMSvermelho){
    digitalWrite(15, HIGH);
    if (contadorRED==0){
      contadorRED = 1;
      sendEmail("vermelho");
    }
  }

  //APAGANDO O ARQUIVO ACCEL
  delay(4000);
  SD.remove("ACCEL.txt");
  Serial.println("Nova medição!");

  //LED'S VOLTAM A FICAR APAGADOS DURANTE AS ANÁLISES
  digitalWrite(5, LOW);
  digitalWrite(15, LOW);
  digitalWrite(16, LOW);
}
```

Anexo III – Desenho técnico do suporte

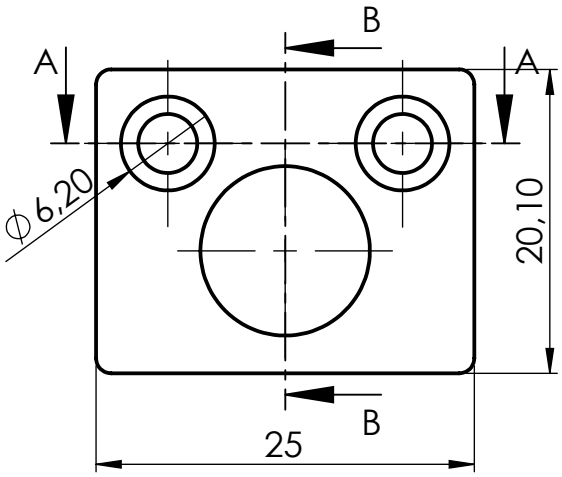
Nesta seção, serão apresentados os componentes que compõem o suporte do acelerômetro, formando a base magnética.

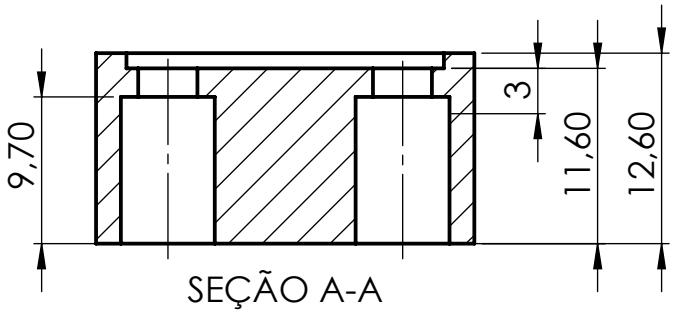






SEÇÃO B-B
ESCALA 2 : 1

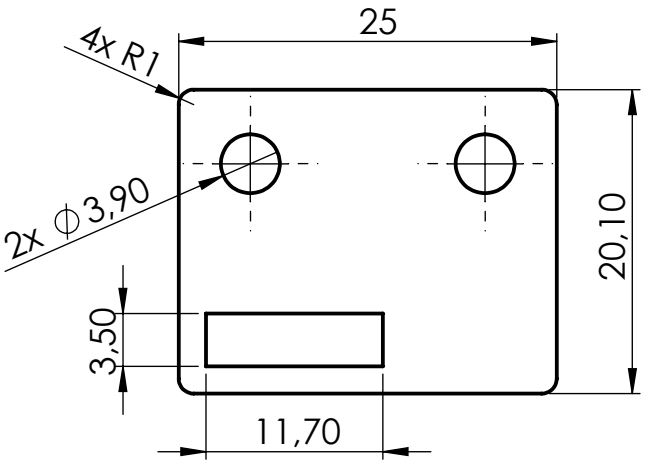


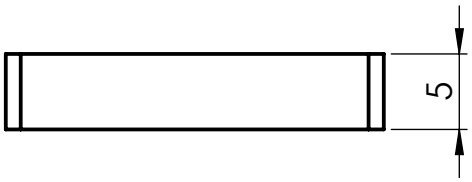


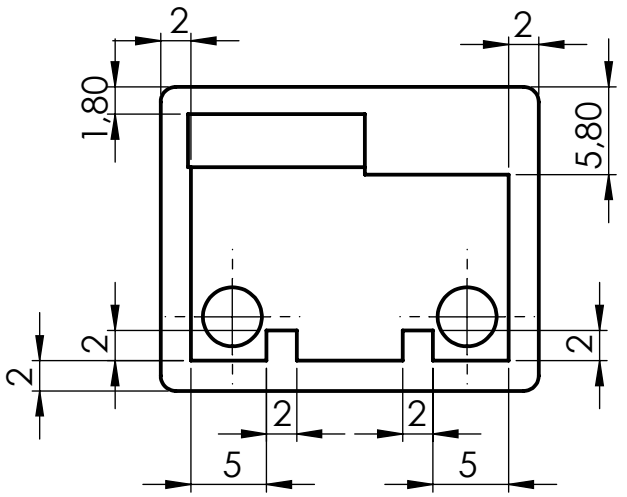
SEÇÃO A-A

Peça	Denominação e observações	Quant.	Material
1	Base	1	PLA

Danilo Castor de Sousa			1° Diedro
Projeto Final - Engenharia Mecânica	Data: 03/01/2018	POLI	Escala: 2:1
Universidade Federal do Rio de Janeiro	Laboratório de Acústica e Vibrações	UFRJ	Unidade: mm







Peça	Denominação e observações	Quant.	Material e dimensões
2	Tampa	1	PLA

Danilo Castor de Sousa			1° Diedro
Projeto Final - Engenharia Mecânica	Data: 03/01/2018	POLI	Escala: 2:1
Universidade Federal do Rio de Janeiro	Laboratório de Acústica e Vibrações	UFRJ	Unidade: mm