CENTRO UNIVERSITÁRIO DO SUL DE MINAS - UNIS
ENGENHARIA ELÉTRICA
EUDER ALVES COSTA

SISTEMA DE MONITORAMENTO REMOTO EM MOTORES DE INDUÇÃO

Varginha
2021

EUDER ALVES COSTA

SISTEMA DE MONITORAMENTO REMOTO EM MOTORES DE INDUÇÃO

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia Elétrica do Centro Universitário do Sul de Minas - UNIS, como pré-requisito para obtenção do grau de Bacharel, sob orientação do Prof. Me. Eduardo Henrique Ferroni.

Varginha

2021

EUDER ALVES COSTA

SISTEMA DE MONITORAMENTO REMOTO EM MOTORES DE INDUÇÃO

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia Elétrica do Centro Universitário do Sul de Minas, como pré-requisito para obtenção do grau de Bacharel, pela aprovação da Banca Examinadora composta pelos membros:

Aprovado em / /

Prof.

_____

Prof.

_____

Prof.

_____

OBS.

AGRADECIMENTO

Agradeço primeiramente a Deus, que tem me abençoado grandemente desde o início do curso. Agradeço aos meus amigos especificamente a Rafaela Zanateli a quem esteve ao meu lado a maior parte deste processo e que passou junto comigo fases boas e ruins e aos seus avós. A Sra. Amélia Zanateli e Sr. Jair Zanateli a quem tenho grande admiração e respeito.

Agradeço em especial a minha mãe - Amália Rodrigues. Que nunca negou apoio e ajuda quando precisei, pelo momento mais difícil da minha vida. Enfim, muitíssimo obrigado. Amo vocês.

RESUMO

Devido à sua simplicidade de aplicação, os motores de indução são equipamentos normalmente mais usado nas indústrias. No entanto, o ambiente de inserção e as condições de funcionamento podem reduzir a sua vida útil. As principais consequências da falha afetam diretamente a linha de produção e trazem prejuízos para a indústria. A vibração embora considere ruim é uma ótima aliada para os diagnósticos em uma manutenção preditiva. O presente projeto visa criar um protótipo de monitoramento remoto contínuo de máquinas rotativas. Utiliza-se um dispositivo móvel (*Smartphone*) com acesso à internet, para o monitoramento obtendo informações de vibração. Em seguida, se realizou o teste para verificar o funcionamento do protótipo em condições operacionais.

Palavras-chave: Motor de Indução, Vibração, ESP 32, MPU6050, *Smartphone.*

ABSTRACT

Due to their simplicity of application, induction motors are equipment most commonly used in industries. However, the insertion environment and operating conditions can shorten its service life. The main consequences of the failure directly affect the production line and bring losses to the industry. Vibration, although considered bad, is a great ally for diagnostics in predictive maintenance. This project aims to create a prototype for continuous remote monitoring of rotating machines. A mobile device (Smartphone) with internet access is used, no monitoring getting vibration information. Then, tests were carried out to verify the functioning of the prototype under operational conditions.

Keywords: Induction motors, vibrations, ESP 32, MPU6050, Smartphone.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS.

MTBF (TMPF)	*Mean time between failures* (Tempo Médio para Falhar).
WI-FI	*Wireless Fidelity* (Fidelidade sem fio).
IEEE	*Institute of Electrical and Electronic Engineers* (Instituto de Engenheiros Elétricos e Eletrônicos).
SSID	*Service Set Identifier* (Identificador de conjunto de serviço).
IP	*Internet Protocol* (Protocolo de Internet)
TCP	*Transmission Control Protocol (Protocolo de Controle de Transmissão)*
DNS	*Domain Name System* (Sistema de Nome de Dominio)
SMTP	*Simple Mail Transfer Protocol* (Protocolo de Transferência de Correspondência Simples)
SSH	*Secure Shell* (Cápsula Segura)
MEMS	*Sistemas Micro Eletromecânicos* (Sistemas Micro Eletromecânicos)
IOT	*Internet of Things* (Internet das Coisas)
RMS	*Root Mean Square* (Raiz Quadrada Média)
AWG	*American Wire Gauge*
URL	*Uniform Resource Locator* (Localizador Uniforme de Recursos)
I2C	*Inter-Integrated Circuit* (Circuito inter-integrado)
HTML	*HyperText Markup Language* (Linguagem de Marcação de Hipertexto)
CSS	*Cascading Style Shee*ts (Folha de Estilo)
ADC	*Analog-to-Digital Converter* (Conversor Analógico-Digital)
GND	*Graduated Neutral Density Filter* (Filtro Graduado de Densidade Neutra)
TX	*Transmit Power/Upstream Power Level (Poder de Transmissão / Nível de energia a Montante)*
RX	*Receive Power/Downstream Power Leve* (Receber energia / Nível de Energia)
GPIO	*General Purpose Input/Output* (Entrada / Saída de Uso Geral)
IDE	*Integrated Drive Electronics (Eletrônica de Acionamento Integrado)*

SUMÀRIO

1. INTRODUÇÃO.

Ao longo dos anos, o uso de motores elétricos se acelerou em diversos setores da economia, o que é vital para o desenvolvimento da indústria, do comércio e do meio rural. Porém, o uso contínuo deste equipamento causará falhas no sistema, exigindo manutenção contínua, entre as quais se destacam as manutenções preventivas, corretivas e preditivas. A manutenção pode ser definida como um conjunto de atividades e recursos aplicados aos sistemas e equipamentos para garantir que eles continuem operando em termos de disponibilidade, qualidade, prazos, custos e vida útil suficiente. Portanto, a manutenção também passou a buscar manutenção rápida e reduzir os serviços de emergência (ASSIS e NOGUEIRA, 2006).

A NBR 5462 (1994), diz que a manutenção corretiva é realizada após uma falha, com o objetivo de restaurar o projeto ao estado de desempenho das funções requeridas.

A manutenção preventiva refere-se à manutenção realizada em intervalos pré-determinados ou de acordo com padrões prescritos para reduzir a possibilidade de falha ou degradação de uma determinada operação.

A manutenção preditiva é a manutenção em que garante a qualidade de serviço exigida, a aplicação de sistemas baseados em tecnologia de análise reduz ao mínimo a manutenção preventiva e reduz a manutenção corretiva através de supervisão centralizada ou amostragem.

A análise de vibração é um método de manutenção preditiva projetado para analisar a condição do equipamento e prever sua manutenção futura e estender sua vida útil. No entanto, devido ao alto valor de aquisição do analisador de vibração, sua aplicação na manutenção é limitada e muitas vezes fica em segundo plano (JÚNIOR , 2016).

A migração para a Indústria 4.0 melhora muito a competitividade entre as empresas, o aumento da produtividade, o aumento da renda, o aumento das oportunidades de emprego e o fortalecimento dos recursos humanos, a otimização dos processos produtivos, o desenvolvimento de tecnologia exponencial e melhores clientes serviço (ARKTIS, 2016)

A implementação da Indústria 4.0 permitirá monitorar equipamentos, otimizar planos de manutenção e obter uma visão em tempo real dos riscos operacionais. (EZRA, 2018).

Com o avanço da tecnologia, tornou-se possível usar plataformas eletrônicas de código aberto para desenvolver dispositivos de baixo custo, como microcontroladores ESP32 e tecnologia MEMS, que podem ser projetados para obter informações na manutenção preditiva e suporte ao desenvolvimento de vários dispositivos.

2. MOTORES DE INDUÇÃO.

As máquinas elétricas rotativas são classificadas de acordo com suas funções, como motores elétricos ou geradores. Quando a energia elétrica da fonte de alimentação é convertida em energia mecânica para rotação, o sistema é denominado motor. Por outro lado, quando o eixo é excitado por uma fonte de energia externa, ele tem a capacidade de gerar uma determinada tensão, agindo assim como um gerador. (FITZGERALD, 2006).

A Figura 1 mostra o motor com os principais componentes independentes. Um motor de indução trifásico é composto por dois componentes principais, um estator e um rotor. O estator é a parte estacionária, e o rotor é a parte rotativa. Rolamentos, coberturas, quadros, eixos, ventiladores, enrolamentos e outros componentes completam o conjunto (CAPELLI, 2013).

Figura 1: Motor de indução.

Fonte: WEG, 2016.

01) Chassis: Estrutura que suporta os componentes internos do motor, podendo ser em ferro fundido ou alumínio com ou sem aletas;

02) Núcleo do estator: laminado de aço eletromagnético;

03) Núcleo do rotor: placa laminada com as mesmas características da placa do estator;

04)Tampa;

05) ventilador;

06) Tampa defletora;

07) Eixo responsável pela transmissão da potência gerada pelo campo magnético;

08) Enrolamento trifásico: Um conjunto de bobinas de igual valor forma um sistema trifásico balanceado que pode promover, um campo magnético giratório;

09) Caixa de junção;

10) Terminal de fiação;

11) Rolamento;

12) Gaiola de esquilo: Barra de alumínio e anel de curto-circuito, injetando alumínio como um todo sob pressão.

2.1.Principais Fontes de Defeitos em Motores Elétricos.

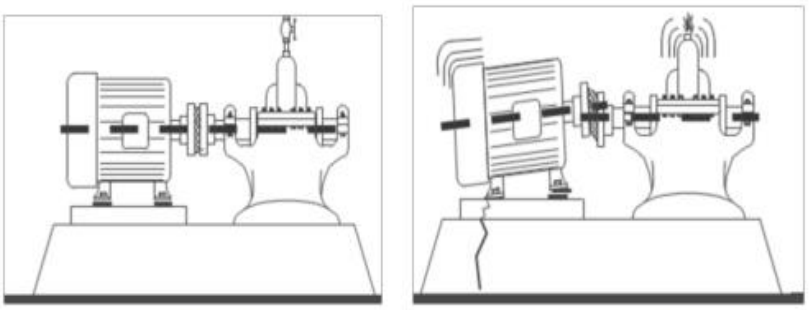
Devido a sua grande utilização, estão expostos a uma ampla variedade de ambientes e condições que os tornam sujeitos a diversos defeitos. Os fatores que afetam o comportamento dos motores podem ser agrupados em falhas de origem mecânica e falhas de origem elétrica. (SILVA, 2012).

2.2.Falhas de Origem Mecânica

2.2.1. Desalinhamento Mecânico

Conforme mostrado na Figura 2, o desalinhamento é tão comum quanto o desequilíbrio. Na montagem mecânica, normalmente existem vários tipos de eixos, rolamentos e acoplamentos com diferentes características dinâmicas. Quando o equipamento está funcionando, haverá forças de interação que eventualmente levarão à vibração. (SILVA, 2012).

Figura 2: Comparação de dois motores com mancais



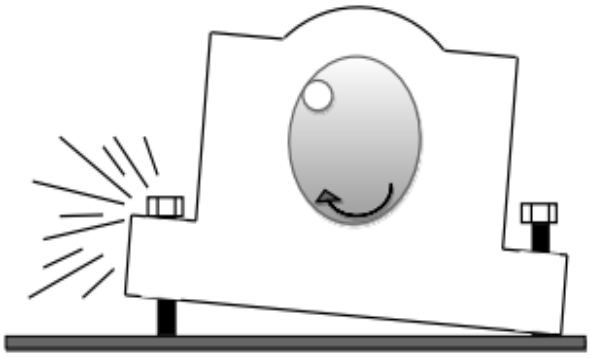
(a) Alinhado. (b) Desalinhado.

Fonte: Silva, 2012.

2.2.2. Folga Mecânica

Maioria das vezes ocasionado por parafuso solto levando assim o equipamento a vibração excessiva. A demora pela corretiva dessa falha leva a máquina ao seu estado de depredação. (MOBLEY E KEITH, 1999). Na Figura 03 representação de folga mecânica.

Figura 3: Representação da folga mecânica



Fonte: Silva, 2012.

2.2.3. Defeitos de Rolamentos

Conforme Figura 4, mostra defeito de rolamento que pode ser causado durante o processo de fabricação ou de uso. Se essas falhas não forem descobertas a tempo, podem causar o mau funcionamento da máquina ou até mesmo colocar em perigo outros componentes, tornando-a inutilizável (BEZERRA, 2004).

Figura 4: Rolamento danificado

Fonte: Ronatec, 2021.

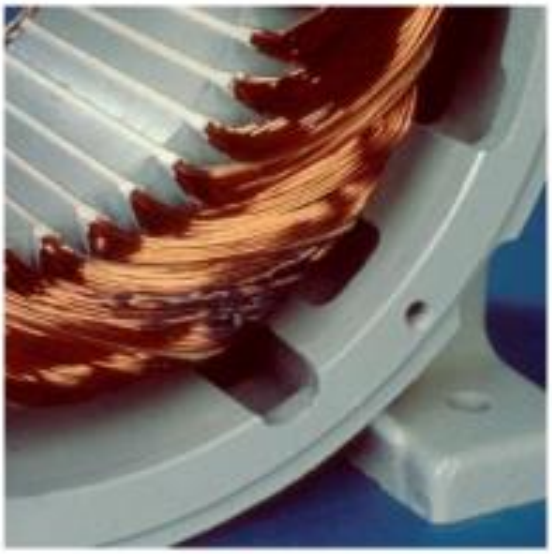
2.3. Falhas de origem elétrica

2.3.1. Curto-circuito

Na Figura 5 se encontra um exemplo de um tipo de falha de isolamento onde frequentemente as causas se dá devido a contaminação do enrolamento, desgaste, vibração ou

pico de tensão. A falha ou ineficiência do processo de impregnação pode agravar essa situação, incluindo o uso de condutores e verniz ou resina de baixa qualidade, preservação inadequada ou danos ao equipamento. Incompatibilidade do grau de resistência ao calor e tensão, e processo de cura insuficiente. (SILVA, 2012).

Figura 5: Curto-circuito entre espiras.



Fonte: Silva, 2012.

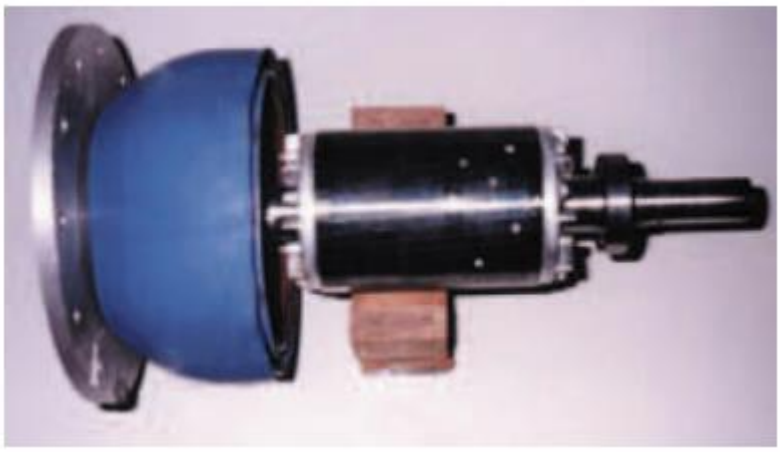
2.3.2. Desequilíbrio de Fase

O desequilíbrio de fase ou desequilíbrio de tensão é caracterizado por diferentes níveis de tensão entre as duas fases. Geralmente é identificado pela análise da corrente do motor. (Silva, 2012).

2.3.3. Barras Quebradas em Rotores

A Figura 6 encontraram falhas comuns dos rotores dos motores de indução são as barras dobradas e soldadas a frio na gaiola. Os sintomas característicos são vibração e ruído anormais. (SILVA, 2012).

Figura 6: Exemplo da simulação de um rotor com barras quebradas.



Fonte: Silva, 2012.

De acordo com a pesquisa de Thomson e Fenger (2001), os motivos da fratura e trinca de barras ou anéis de aço podem ser decorrentes de esforços:

- Térmico: causado por sobrecarga;
- Magnético: devido a vibrações, excentricidade;
- Residual: problemas de fabricação;
- Dinâmico: forças centrífugas, conjugado da carga;
- Ambiental: contaminação química ou devido à umidade;
- Mecânico: fadiga.

Nos motores de indução cerca de 40-50% das falhas se dá a problemas de rolamento, 30-40% são falhas do estator e 5-10% são falhas do rotor. (BONNETT, et al., 2008).

Uma manutenção eficaz deve ser realizada para manter o motor nas melhores condições de funcionamento, pois todos os processos de produção estão relacionados ao uso do motor de alguma forma, e as falhas afetam diretamente o custo do produto final. (GONGORA, 2016)

3. CONCEITO DE MANUTENÇÃO.

A manutenção é caracterizada por uma combinação de ações, tecnologia e gestão, incluindo supervisão, visando manter ou restaurar o projeto a um estado em que ele possa desempenhar as funções necessárias. (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 1994).

Outra definição comumente usada é o comportamento ou efeito da manutenção, ou todos os comportamentos necessários para preservar ou reparar um equipamento de forma que ele possa permanecer sob certas condições. (TAVARES, 1999, p.39)

3.1.Tipos de Manutenções.

3.1.1. Manutenção Corretiva.

Consiste em intervenções que ocorrem após a avaria do equipamento e permite que os mesmos voltem ao funcionamento (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS 5462, 1994).

Para Tsang (1995, p.3), esse tipo de manutenção costuma ser caro pelos seguintes motivos:

Alto custo de recuperação das condições de operação do equipamento em situação de crise.

- Danos secundários e riscos de segurança impostos pela falha.
- Penalidades associadas pela perda de produção.

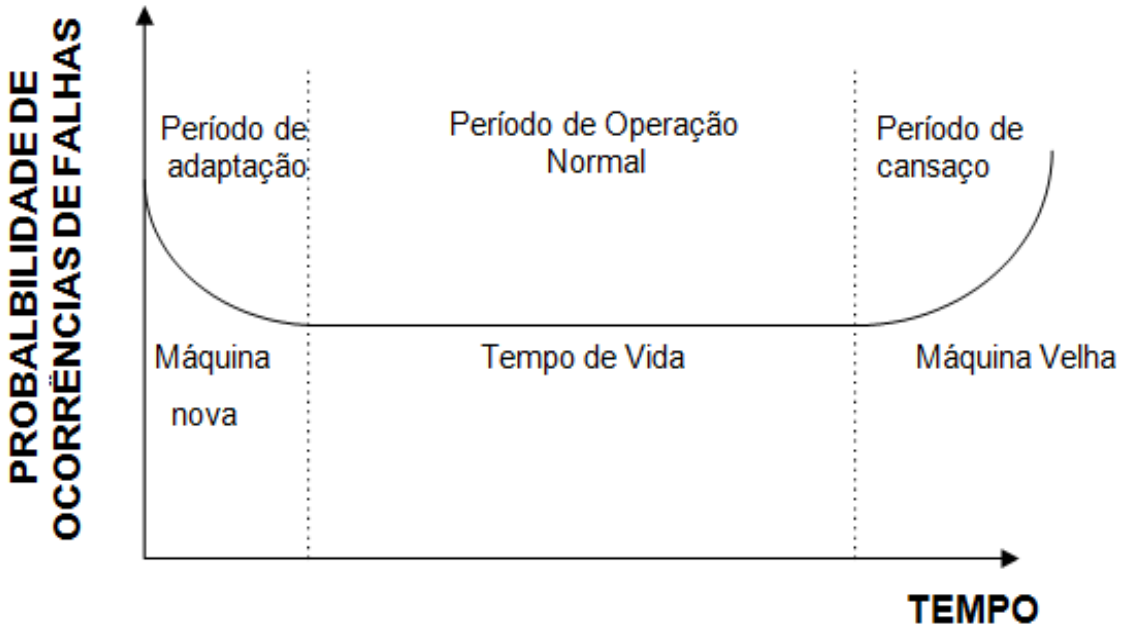
3.1.2. Manutenção Preventiva.

É realizado por meio de intervenção com base em resultados estatísticos ou informações programadas fornecidas pelo fabricante. Portanto, o principal objetivo é reduzir as falhas do equipamento para evitar o tempo de inatividade do equipamento devido a danos (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 1994).

Um conjunto de medidas preventivas baseadas no tempo ou baseadas em padrões pré-estabelecidos e medidas preventivas baseadas nas condições destinadas a reduzir a incidência de danos ou degradação do equipamento (XENOS, 1998, p. 35)

Até 1960, a base da manutenção preventiva era a curva do tempo médio até a falha, chamada de curva da banheira à curva da banheira, que era usada para expressar a probabilidade de falha de equipamentos ou sistemas durante a operação (Motta 1999) mostrada na Figura 7.

Figura 7: Curva de taxa de falha.



Fonte: adaptado de Wuttke, 2008.

No período de adaptação, há uma taxa de queda e, geralmente a falha é causada de fabricação. Falhas no período de operação tendem a ser erros operacionais. Na terceira etapa, temos um aumento no gráfico, possivelmente devido ao desgaste do material. A curva da banheira é um gráfico que representa uma falha esperada de um equipamento ao longo de um período de tempo, considerando que ele venha a falhar até o momento. (SELLITTO, 2005).

4. MANUTENÇÃO PREDITIVA.

Sua finalidade é baseada na aplicação de um conjunto de técnicas analíticas, utilizando supervisão centralizada ou amostras para garantir o controle de qualidade do serviço, de forma a minimizar manutenções preventivas e reduzir correções. (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS 5462, 1994).

O benefício da manutenção preditiva é que o tempo médio entre falhas (MTBF) pode ser estimado. Este indicador fornece uma base para determinar o tempo mais econômico para substituir uma máquina, em vez de continuar a arcar com altos custos de manutenção. Quando o custo de operação e manutenção do equipamento (MTBF) ultrapassar o custo de reposição, a máquina deverá ser trocada. (COLACOTT, 1999)

4.1.Importância da Manutenção Preditiva

A manutenção preditiva inclui atividades destinadas a prever falhas por meio do monitoramento da máquina (GONÇALVES NETO et al, 2013).

Entre as principais razões para a decisão da organização de incluir um plano de manutenção preditiva, podemos citar a proteção de ativos financeiros, o reconhecimento da administração das capacidades de melhoria, taxas de seguro mais baixas e mais provável de ser certificado pela ISO 9001 (MOBLEY, 1999). Tudo isso é possível ao atingir as vantagens da Manutenção Preditiva, que são:

- Redução dos custos de manutenção;
- Redução de falhas nas máquinas;
- Redução do tempo de parada para reparo;
- Redução no estoque de peças sobressalentes;
- Aumento da vida útil das peças;
- Aumento da produção melhoria na segurança do operador;
- Verificação das condições do equipamento novo;
- Verificação dos reparos lucro global;

Um adequado plano de manutenção pode reduzir em média de 20% a 25% nos custos diretos de manutenção. A manutenção preditiva requer instrumentos tecnicamente para realizar testes e diagnósticos de máquinas. Esses instrumentos costumam ser muito caros e requerem pessoal treinado tecnicamente para analisar os resultados. O impacto do custo, sejam instrumentos complexos ou mão de obra qualificada, muitas vezes deixa a manutenção preditiva em segunda plano. (SCHEFFER, GIRDHAR, 2004).

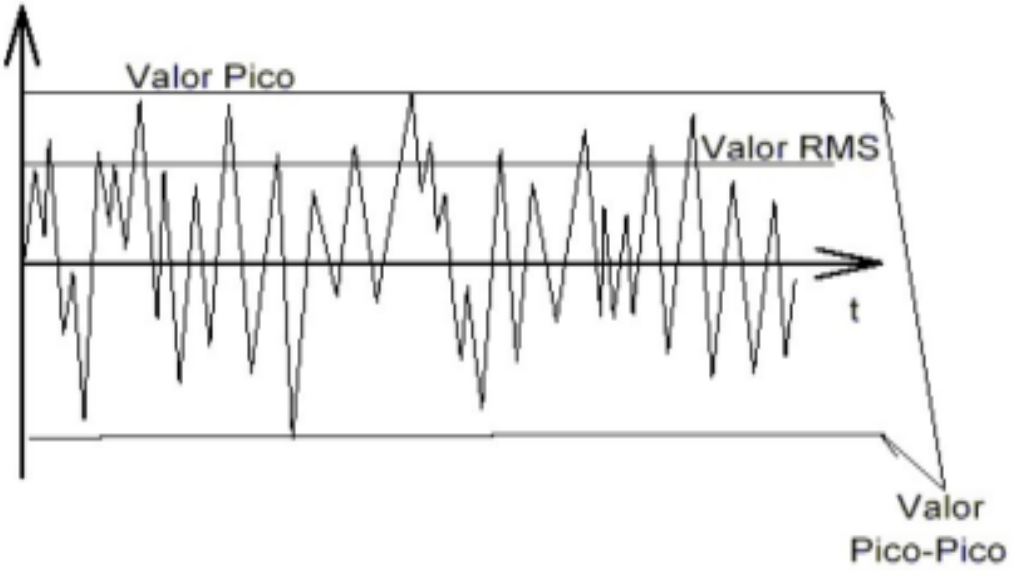
5. ANÁLISE DE VIBRAÇÕES.

Todas as máquinas em operação produzirão vibração, levando ao seu processo de degradação. Ao rastrear o progresso do nível de vibração, podem ser obtidas informações sobre a condição da máquina. Essas vibrações são transmitidas para toda a máquina como um todo e, por fim, produzem um espectro de frequência específico para o sistema, denominado "recursos". A análise pode verificar se o dispositivo está em operação normal ou se existem defeitos iniciais que podem causar falha (MARAN 2012).

O valor de pico é utilizado para identificar picos gerados por impactos, no entanto indica

apenas a ocorrência do valor de pico e não considera o histórico de tempo da onda. O valor pico a pico representa o intervalo máximo da onda, ou seja, o período mais longo e é usado para identificar falhas precoces e tardias sem ter que considerar o histórico da onda que muda com o tempo. O parâmetro na medição de nível mais relevante é o valor RMS, pois considera o histórico da forma de onda ao longo do tempo e comprova a gravidade de um sinal, sendo este, destrutivo desta vibração (GARCIA, 2005). Veja na Figura 8, representação de um sinal de vibração.

Figura 8: Sinal de vibração.



Fonte: NETO, 2012.

Existe uma correlação entre o valor de pico a pico, o valor de pico e o nível RMS de uma onda senoidal. No caso de medição de ruído, o valor efetivo é muito importante. E sua importância na pesquisa estatística, devido ao desvio padrão do processo aleatório. (RAO 2008) Veja na Equação 01 como obter os valores RMS.

$$RMS = \sqrt{\frac{1}{T}\int_0^t x(t)dt} \qquad (01)$$

O monitoramento e a análise de vibrações fazem parte do método de predição e são considerados um dos métodos mais importantes para proteção de equipamentos. Monitorar a vibração de equipamentos rotativos é muito importante, e são usados métodos e instrumentos em conjunto com o software de suporte para monitorar (KARDEC E NASCIF, 2015)

6. NÍVEL GLOBAL DE VIBRAÇÕES

O nível global de vibração é um dos métodos para avaliar o desempenho de equipamentos rotativos. Inclui a medição do nível global (valor efetivo) do sinal do transdutor. A ocorrência de algum defeito resultará em uma alteração no nível global, que pertence às características de cada máquina, a vibração faz com que este parâmetro mude. As medições do sistema são projetadas para avaliar a intensidade das vibrações e se elas estão dentro dos limites aceitáveis. (ANTONIOLLI , 1991)

Além de ser confiável e permitir ações muito antes de atingir o estágio perigoso ou mesmo catastrófico. Normalmente, este método é utilizado como uma etapa inicial, para métodos mais sofisticados, um diagnóstico mais preciso pode ser estabelecido e as irregularidades podem ser eliminadas no menor tempo possível. A desvantagem do método de nível global é que ele não pode determinar com precisão a fonte da falha que causa vibração excessiva. (NEPOMUCENO, 1989).

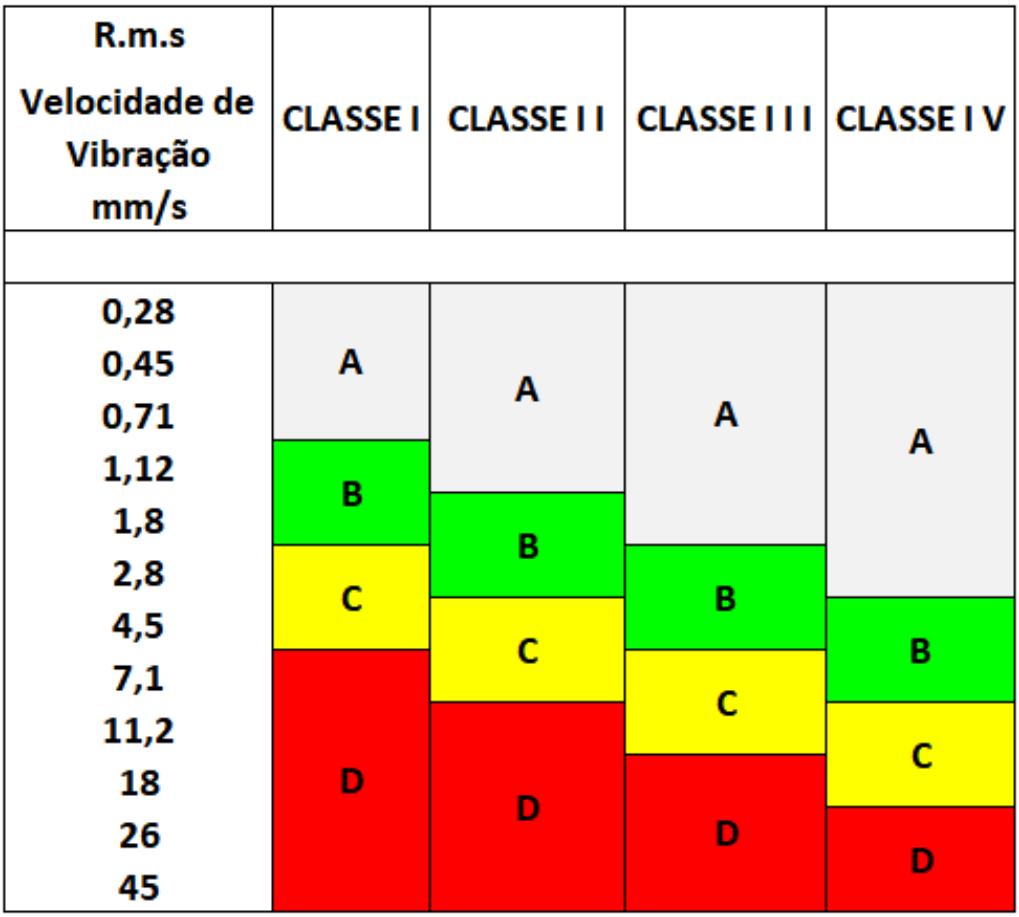
6.1.Parâmetro de Vibração Nível Global.

De acordo com a ISO 10816 - 2016 o status operacional de máquinas elétricas rotativas pode ser classificado de acordo com os níveis de vibração. O estado da máquina é definido como quatro intervalos A, B, C e D, que são:

A. Equipamento novo ou em perfeitas condições de operação;
B. Equipamento em boas condições de operação;
C. Equipamento sob condições operacionais permitidas, mas em níveis de manutenção e alerta;
D. Equipamento em condições de operação não permissíveis com níveis de vibração ultrapassam o limite permitido.

A Figura 9 apresenta os níveis de severidade conforme a norma.

Figura 9: Classificação de Vibração.



Fonte: Adaptado de *IEC Motor Service Manual*, 2021.

Classe I: Motores individuais de até 15 kW.

Classe II: Máquinas de tamanho médio de 15 a 75 kW)) sem fundações especiais ou máquinas de até 300 kW em fundações especiais.

Classe III: Movimentadores primários grandes que são relativamente rígidas em relação à direção da medição de vibração.

Classe IV: Máquinas com grandes massas rotativas e fundações pesadas que são relativamente frouxas em relação a direção da medição de vibração. (Ex.: Conjunto turbo geradores e turbinas a gás com saídas maiores que 10 MW).

O acelerômetro é um tipo de sensor usado para detectar vibrações cuja resposta é proporcional à aceleração gerada na parte inferior do sensor. Os dados do nível de vibração são geralmente obtidos pelo sensor no domínio do tempo e podem ser convertido para o domínio da frequência pela transformada rápida de Fourier (GUIDE 2007).

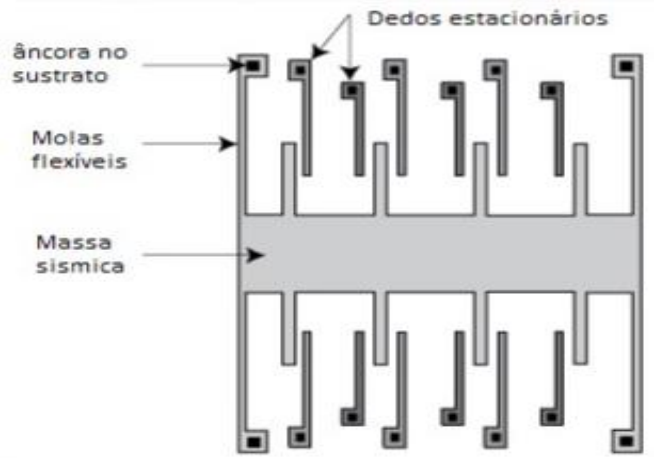
7. TRANSDUTORES DE VIBRAÇÃO

Os transdutores convertem o sinal mecânico em um sinal elétrico. Para a detecção de vibração, existem muitos tipos de transdutores, os acelerômetros são os mais usados devido à sua versatilidade, enquanto outros transdutores são limitados a aplicações específicas. (SEQUEIRA, 2013)

7.1.Transdutores Acelerômetro MEMS.

Acelerômetro MEMS (microeletromecânico) integra dispositivos eletrônicos que variam em tamanho de alguns microns a centenas de microns. Eles geralmente transmitem sinais de um domínio físico para outro, como mecânico para elétrico. (O'NEAL, 1999). A Figura 10, tem um diagrama de um acelerômetro.

Figura 10: Diagrama do Acelerômetro *MEMS*



Fonte: *Compliant Mechanisms*, 2014.

O acelerômetro converte energia mecânica em energia elétrica e sua fonte de informação é a aceleração do sistema. Consiste em três estruturas básicas: massa sísmica, área da mola e índice estrutural ou capacitância. Ao aplicar aceleração à massa vibratória fisicamente conectada à placa do capacitor, ela muda a distância ou a superfície entre os dedos do capacitor, alterando assim a capacitância do capacitor. Essa capacidade é proporcional à aceleração aplicada ao sistema. (TEZ; AKIN, 2013). A tensão de saída é descrita pela Equação 02.

$$V_{semse} = \frac{2C_s}{2C_s + C_P + C_{gs} + C_{gd}} . \frac{V_m}{\omega \frac{2}{n} d} . a \qquad (02)$$

Onde:

V_{semse} = Tensão de saída;

C_s = Capacitância nos dedos capacitivos;

C_P = Capacitância parasita;

C_{gs} e C_{gd} = são duas capacitâncias do transistor MOS;

V_m = Amplitude do sinal modulado;

d = Distância entre os dedos capacitivos do sensor;

ω = Frequência de ressonância mecânica do transdutor;

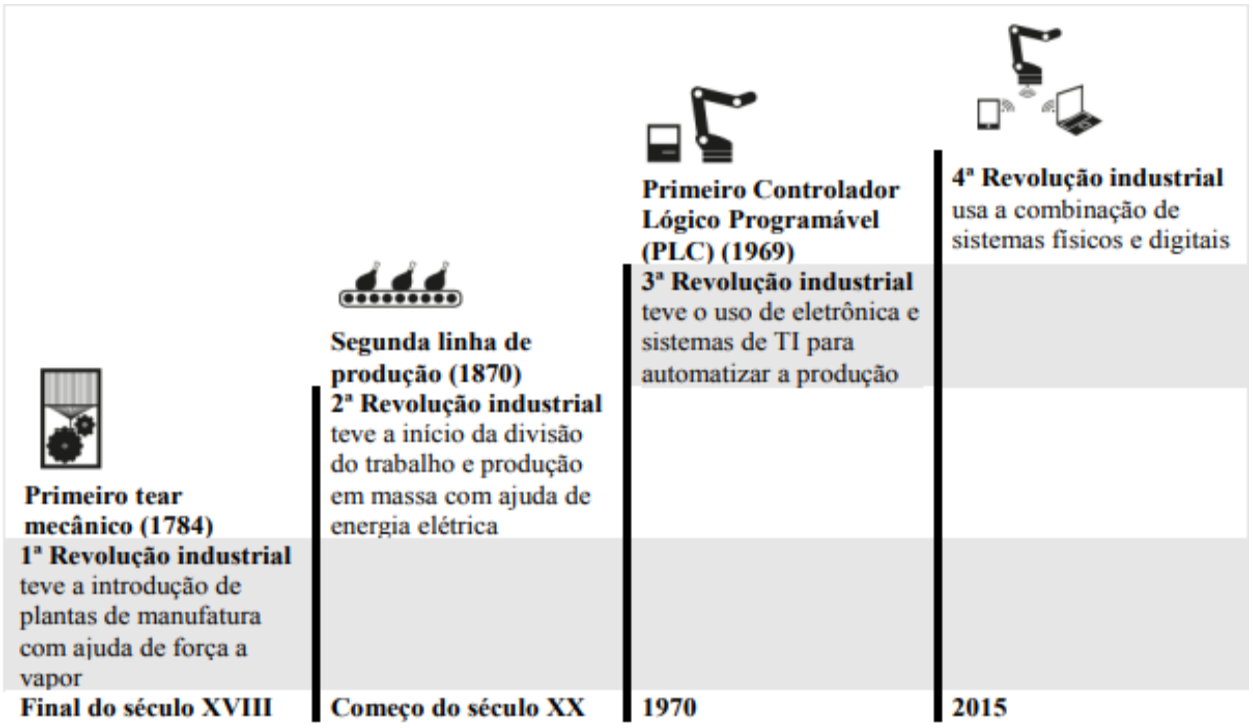
a = Aceleração a ser medida.

8. INDÚSTRIA 4.0

A Indústria 4.0 originou-se da digitalização de fábricas inteligentes. Propõe a conexão efetiva da estrutura física com a estrutura virtual, e constrói redes inovadoras, de produção mais rápida e inteligente (MAURA, 2019).

Silva, Santos Filho (2015) fala que: "A Indústria 4.0 prevê a integração entre humanos e máquinas, mesmo em locais remotos, forma uma rede em larga escala e fornece produtos e serviços de forma independente." Na Figura 11 tem uma cronologia da revolução industrial.

Figura 11: Revolução Industrial.



Fonte: Adaptado Henning, 2013.

Esse fenômeno é chamado de Quarta Revolução Industrial e faz parte da transição de um ambiente industrial para um mundo altamente interconectado, onde as informações são constantemente compartilhadas. Desta forma, novos conceitos como a Internet das Coisas foram introduzidos. (CARDOSO, 2016).

8.1.A internet das coisas (IoT) na era da Indústria 4.0

O conceito da Internet das Coisas foi mencionado pela primeira vez por Kevin Ashton, refletindo sua aplicação de identificadores de radiofrequência à cadeia de suprimentos. Mais tarde, quando foi aplicado a um novo sistema de comunicação no qual a Internet estava conectada ao mundo físico por meio de uma rede sem fio, o termo se tornou amplamente conhecido. (SOBREIRA, 2018).

Tecnologias e padrões de rede sem fio surgiram, adaptando-se a uma ampla gama de aplicações e cobertura. O destaque é o Wi-Fi (*Wireless Fidelity*), que é o nome de uma tecnologia sem fio compatível com o padrão IEEE 802.11.(SOUTO, 2005).

8.2.Rede de Comunicação Wi-Fi

A tecnologia de rede proprietária que foi pioneira no padrão IEEE 802.11 / Wi-Fi foi desenvolvida no início da década de 1990 e operava inicialmente na banda de frequência de 900MHz disponível na América do Norte. Posteriormente, essa tecnologia passou a operar na banda de frequência ISM (Ciência Industrial e Médica) de 2,4 GHz, disponível globalmente (SOUTO, 2005).

O Wi-Fi (*wireless fidelity*), é um conjunto de especificações para uma rede local sem fios, esta tecnologia permite ligar vários dispositivos sejam laptops, impressoras, televisões, entre outros, desde que estejam dentro do alcance da rede evitando assim que seja necessário passar cabos entre os equipamentos diminuindo assim os custos de interligar os dispositivos. Todos os dispositivos da rede, chamados de clientes, estão ligados a uma central (*Access Point*). Esta rede criada possui um nome que a identifica do ponto de vista do utilizador, SSID (*Service Set Identifier*). (OLIVEIRA, 2017).

8.3.Protocolo TCP/IP

A comunicação entre dispositivos requer um protocolo TC / IP, que é um conjunto de

protocolos de comunicação. TCP é o nome do protocolo de transporte da Internet e do protocolo IP. Eles têm o princípio de padronizar todas as comunicações de rede. Portanto, em sua estrutura, constatamos que o HTTP permite a navegação em páginas da web, de forma que o DNS converte a URL do navegador em um número único (IP), que é utilizado como identificador da rede local. (M.SILVA, 1992) .

8.4.Conceito de Arduíno

O Arduino é uma plataforma de serviço aberta, que permite que a tecnologia seja usada para vários fins. O objetivo de é tornar a prototipagem de projetos envolvendo eletrônica e programação mais acessível. O primeiro Arduino foi projetado pelo Professor Massimo Banzi e David Cuartielles em Ivrea, Itália, em 2005. (OLIVEIRA, 2017).

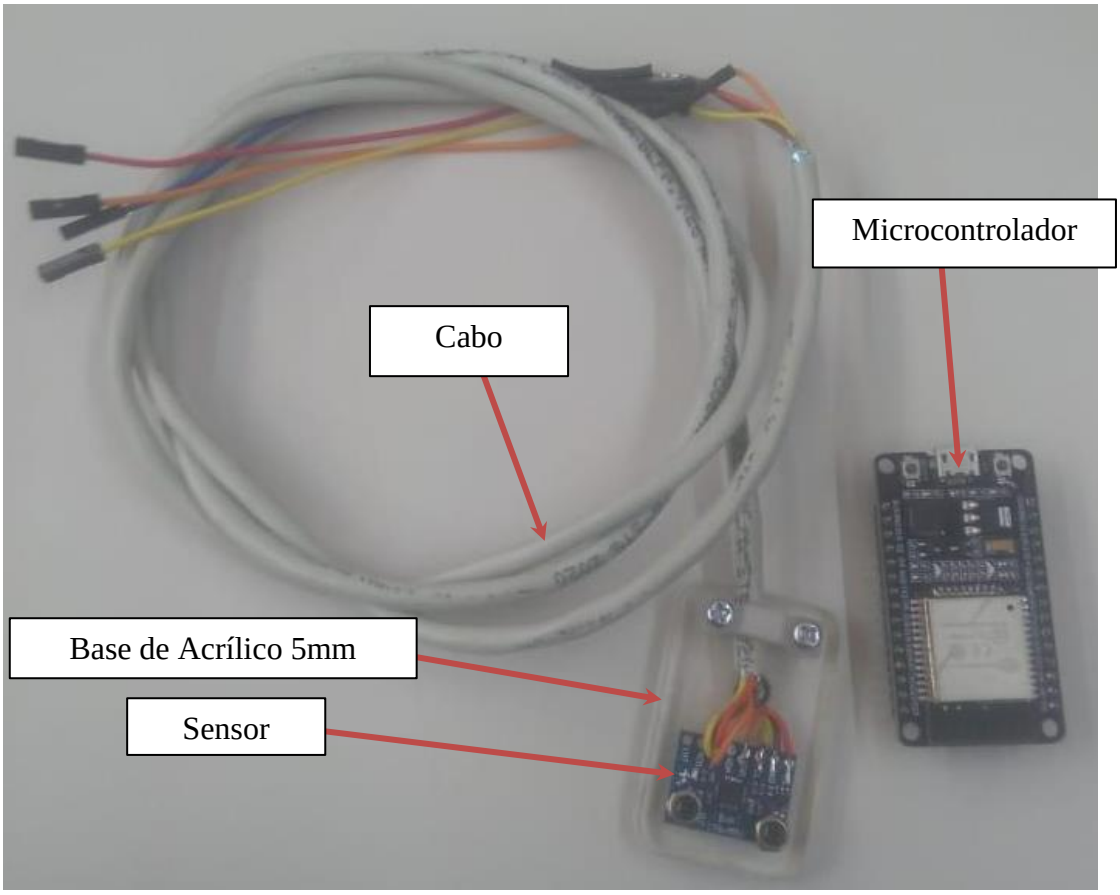
9. MATERIAIS E MÉTODO.

Este capítulo apresenta o conceito e a estrutura do desenvolvimento de um protótipo. O objetivo é obter de maneira remota as leituras de vibração utilizando dispositivos móveis. Para tanto, sensores de tecnologia MEMS são usados para ler a vibração, e um microcontrolador é usado para decodificar os dados e enviar a leitura de valores RMS para um *smartphone*.

9.1.Confecção do Protótipo.

Na confecção do protótipo, foi utilizado o acelerômetro modelo MPU 6050, no qual a informação de vibração é gerada. Um microcontrolador da família *Espressif*, ESP32 utilizado para o se fazer cálculos matemáticos e enviar os valores do sinal resultante ao *smartphone*. Na fixação do sensor MPU 6050 no motor foi necessário a confecção de um suporte. O suporte foi fabricado de um material acrílico, tendo 80 milímetro de comprimento 30 milímetro de largura e 5millimitro de espessura, também foi preciso de dois ímãs de neodímio com diâmetro de 6 mm para fixar ao motor. Na comunicação entre o microcontrolador ESP32 e o sensor MPU6050 é usado um cabo blindado de quatro vias 1 X 25 AWG. A Figura 12 mostra o protótipo confeccionado pronto para o uso.

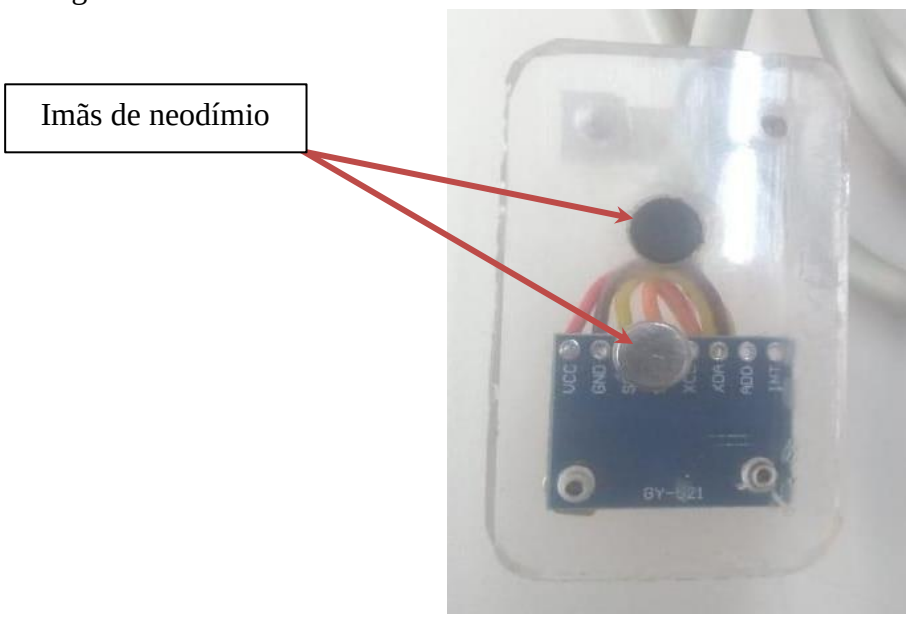
Figura 12: Protótipo



Fonte: Autor, 2021.

A Figura 13 demonstra a alocação dos imãs na base inferior do acrílico.

Figura 13: Base inferior do acrílico.



Fonte: Autor, 2021.

De acordo com o fabricante o material utilizado para a construção do chassi do motor

elétrico e o ferro fundido, partindo desse conceito para melhor fixação no chassi foi necessário instalar dois ímãs de neodímio.

9.2.Custo Protótipo.

A Figura 14 exibe o levantamento do investimento para a confecção do protótipo, o preço é disponível na data atual 29/10/021.

Figura 14: Preço do total do protótipo.

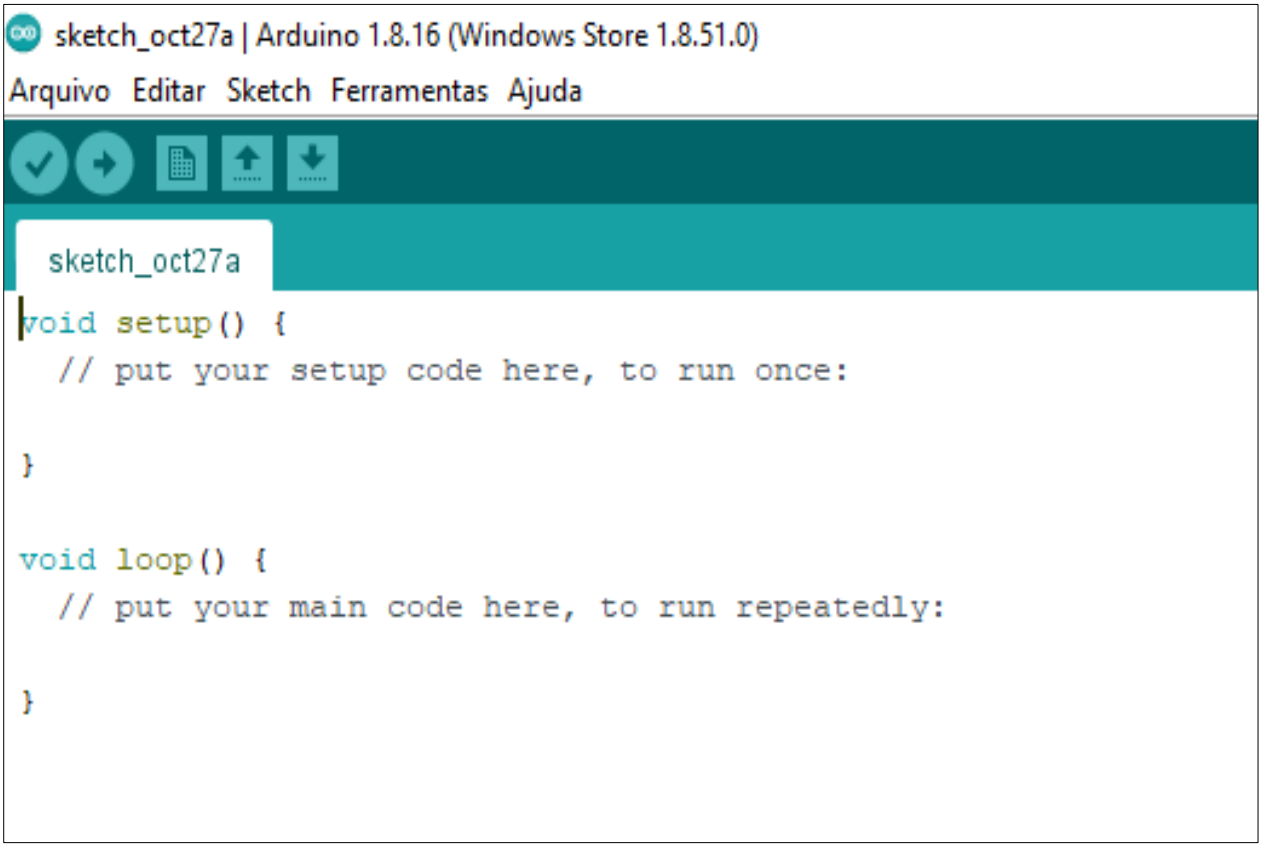
Item	Preço R$	Quant.	Total
Microcontrolador ESP32	R$ 62,00	1	R$ 62,00
Acelerômetro MPU 6050	R$ 20,00	1	R$ 20,00
Cabo blindado 4x25 awg	R$ 3,50	1M	R$ 3,50
Imã neodímio	R$ 2,50	2	R$ 5,00
Acrílico	Sem custo	"--"	"--"
	Total		**R$ 90,50**

Fonte: Autor, 2021.

9.3.Software

É um conjunto de informações que deve ser completado por um determinado mecanismo, que pode ser um computador ou qualquer outro dispositivo que requeira lógica. Este termo é usado para descrever programas, aplicativos e instruções de código para dizer o que fazer. Para o desenvolvimento do código no ESP 32, um programa precisa ser executado no computador, que se chama IDE (Integrated Development Environment). O Arduino IDE pode ser instalado em vários sistemas, como Windows, Linux, Mac e Android. Faz uso de uma linguagem de programação chamada Wiring baseada em C ++ (RAMOS et al., 2007). A Figura 15.1 mostra a interface de programação de código aberto, o IDE Arduíno.

Figura 15.1: IDE Arduíno



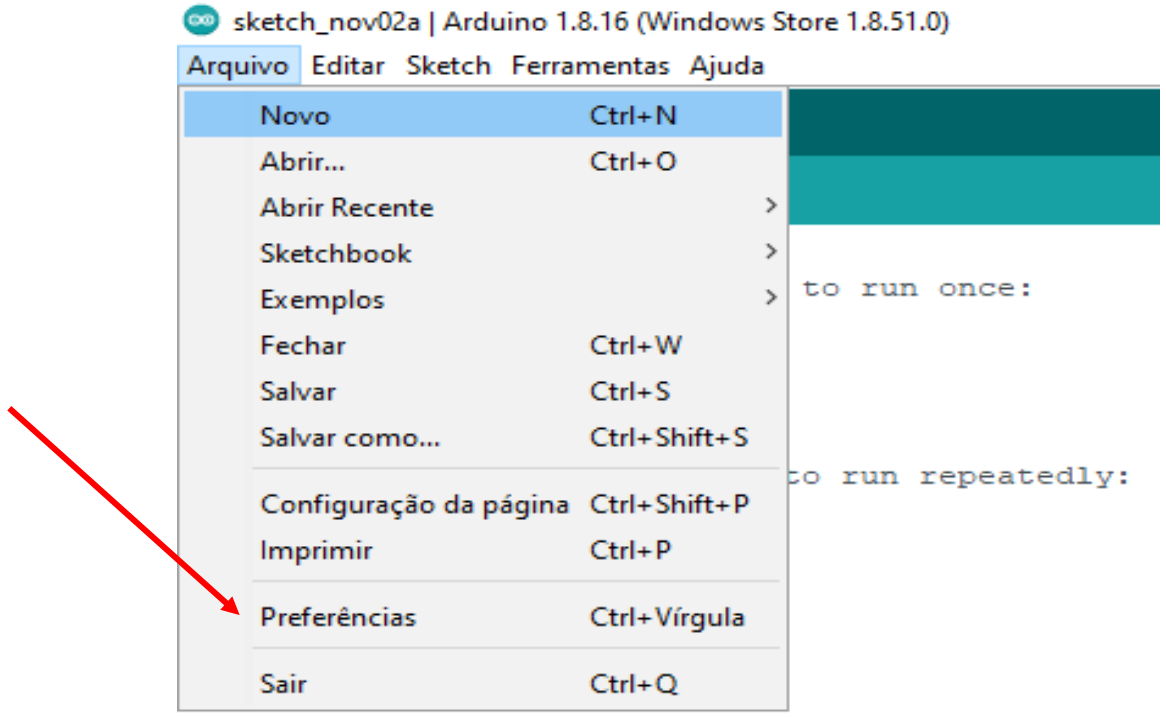
Fonte: Arduino, 2021

Por sua vez, o microcontrolador ESP32 da série *Espressif* é utilizado para receber os dados enviados pelo acelerômetro. O microcontrolador ESP32 foi escolhido devido ao seu custo e benefícios para o desenvolvimento de dispositivos IoT.

9.4.Configuração da IDE Arduíno.

Na IDE do Arduíno algumas configurações precisam ser feitas para que o código possa ser carregado no ESP32. Com a interface do Arduíno inicializada vá em arquivos>preferências, conforme a Figura 15.2.

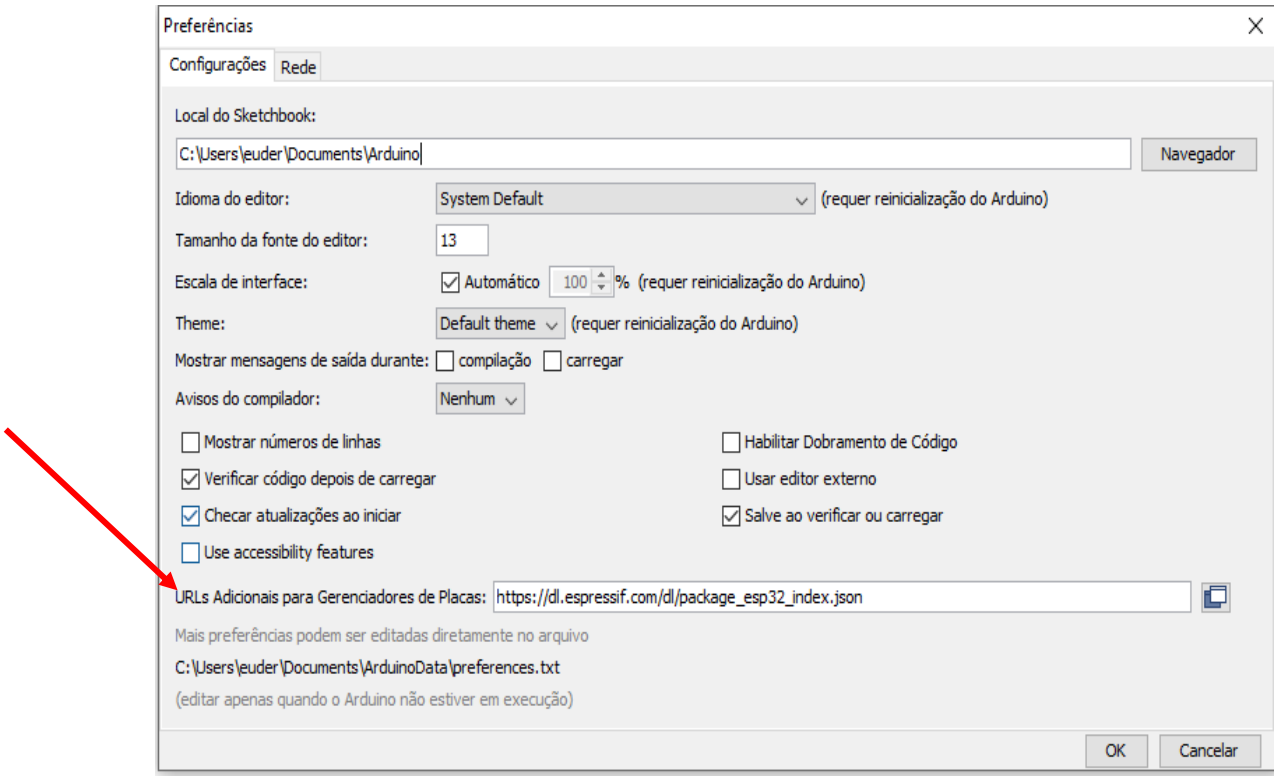
Figura 15.2: Configuração IDE Arduíno 01



Fonte: Arduíno, 2021

Em seguida digitada o *liink* (https://dl.espressif.com/dl/package_esp32_index.jsonno) no campo URL, de acordo com a Figura15.3.

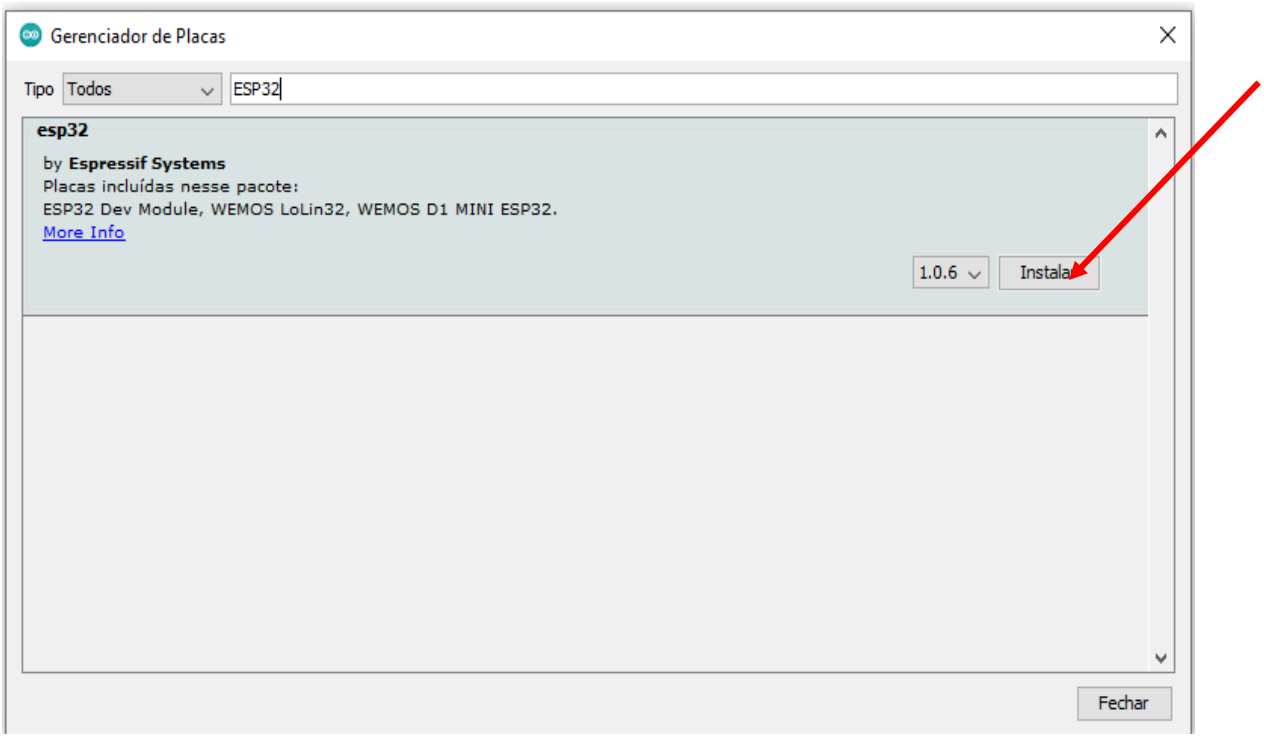
Figura 15.3: Configuração IDE Arduíno 02.



Fonte: Arduíno, 2021.

Clica-se em “OK” e retorna à tela inicial. Agora deve-se ir em Ferramentas > Placa > Gerenciador de Placas, e no campo de busca digite ESP32, conforme a Figura 15.4.

Figura: 15.4: Configuração IDE Arduíno 03.



Fonte: Arduíno, 2021.

Com a opção ESP32 selecionado clique em instalar para finalizar o processo. Depois desse passo, deve ser escolhida a placa adequada neste caso o ESP32, em Ferramentas > Placa e instalar além da respectiva porta COM na qual o dispositivo está conectado.

Depois de definir as configurações necessárias para o uso do microcontrolador, é feita a análise do código.

9.5.Análise do Código Apêndice A.

O código do sistema analisado se encontra em apêndice A, e consiste em três partes: a definição das bibliotecas, *setup* e *loop*. Na primeira linha do código importamos biblioteca Wire.h que é responsável por fazer a comunicação entre as portas I2C. Logo definimos as variáveis, conforme mostrado na Figura 16: Bibliotecas e variáveis definidas.

Figura 16: Bibliotecas e variável definidas.

```
#include<Wire.h>//Biblioteca para comunicação I2C

const int MPU=0x68;//Endereço do sensor
int16_t AcX,AcY,AcZ; //Variaveis guarda os valores lidos

float g = 9.8; // Valor da aceleração da gravidade
float resolucao = 0.000604043; // Resolução de 9.8/16224.010
```

Fonte: Arduíno, 2021.

Na função *setup* iniciamos a porta serial do Arduino com a taxa de transferência determinada, logo a comunicação I2C e o endereçamento do sensor MPU6050, e confirma se verdadeiro. Veja na Figura 17 a configuração do *setup*.

Figura 17: Configuração do *setup*.

```
void setup(){
//Inicia a comunicaçao serial (para exibir os valores lidos)
  Serial.begin(115200);
//Inicia a comunicação I2C
  Wire.begin();
//Começa a transmissao de dados para o sensor
  Wire.beginTransmission(MPU);
// registrador PWR_MGMT_1
  Wire.write(0x6B);
// Manda 0 e "acorda" o MPU 6050
  Wire.write(0);
  Wire.endTransmission(true);
```

Fonte: Arduíno, 2021.

Na função de loop, as primeiras quatro linhas são para chamar o endereço e transmissão do sensor MPU 6050. Se a porta MCU não receber nenhuma informação, ela termina (falso), caso contrário, verifica-se que o programa de execução de informações lê o valor (verdade). Na Figura 18 mostra a configuração do *loop*.

Figura 18: Configuração do *loop* parte 01.

```
void loop(){
//Começa a transmissao de dados para o sensor
  Wire.beginTransmission(MPU);
// registrador dos dados medidos
  Wire.write(0x3B);
//Caso não encontre nenhuma informção dá como encerrado.
  Wire.endTransmission(false);
// faz um "pedido" para ler 14 registradores,
que serão os registrados com os dados medidos
  Wire.requestFrom(MPU,14,true);
```

Fonte: Arduíno, 2021.

Uma vez que o sensor envia dados, ele pode obter leituras brutas sem processamento de dados, que podem ser encontradas nas variáveis AcX, AcY e AcZ. Os valores manipulado da leitura da aceleração pode ser encontrado nas variáveis gX, gY e gZ. A Figura 18.1 mostra a configuração do código no *loop*.

Figura: 18.1 Configuração do *loop* parte 2.

```
// Guarda o valor original da leitura do acelerômetro.
  AcX=Wire.read()<<8|Wire.read();
// Guarda o valor original da leitura do acelerômetro.
  AcY=Wire.read()<<8|Wire.read();
// Guarda o valor original da leitura do acelerômetro.
  AcZ=Wire.read()<<8|Wire.read();

// Valor convertido em aceleração
 float gX = resolucao*AcX;
// Valor convertido em aceleração
 float gY = resolucao*AcY;
// Valor convertido em aceleração
 float gZ = resolucao*AcZ;

//Agora escreve os valores no monitor serial
 Serial.print("AcX = "); Serial.print(gX);
 Serial.print(" | AcY = "); Serial.print(gY);
// Valor convertido com o offset no eixo Z.
 Serial.print(" | AcZ = "); Serial.println(gZ-g);


  delay(333); // Tempo de 333 millissegundos
}
```

Fonte: Arduíno, 2021.

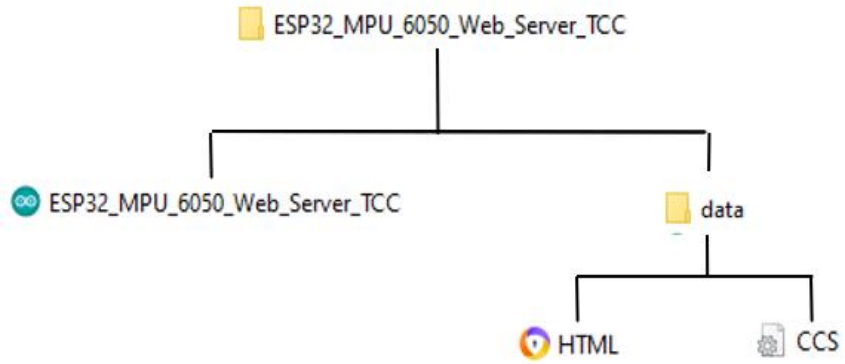
Após a conversão dos dados para a leitura de aceleração o mesmo imprime os valores na porta serial do Arduíno, após 333 milissegundos, encerra o ciclo e a operação é reiniciada.

9.6. Análise do Código Anexo I

No código analisado acima, foi escrito para executar as informações do acelerômetro e imprimi-las na porta serial IDE do Arduino, portanto o segundo código que se encontra em Anexo I foi desenvolvido para imprimir as informações no *smartphone*. Para otimizar e promover a compreensão do código, são utilizadas bibliotecas e arquivos adicionais.

Para tornar o código mais organizado e fácil de entender, três arquivos diferentes serão criados durante a construção do servidor web, a saber Arduino IDE, HTML e CSS. A Figura 19 mostra a aparência da estrutura de pastas. (RUI SANTOS, 2021). Na Figura 19 exibe como deve ficar a organização da pasta de arquivos.

Figura 19: Estrutura da Pasta de Arquivos.



Fonte: Adaptado Rui Santos, 2021.

- Arquivo Arduíno é onde se desenvolve toda lógica de programação;
- Arquivo HTML serve para integrar certos atributos de formatação e especificações de layout para o conteúdo Web, veja no anexo II desenvolvimento do código;
- Arquivo CSS é usado para estilizar, onde se dá toda estética da página, código encontra-se em anexo III.

Dentro da plataforma de desenvolvimento IDE Arduíno, o código segue a mesma estrutura desenvolvida anteriormente onde é separado em três etapas:

1º. Definição da biblioteca e variáveis globais;

A biblioteca fornece funções adicionais para esboçar, por exemplo, usar *hardware* ou manipular dados, o que facilita no desenvolvimento de códigos. Para desenvolvimento do código

foi preciso o uso de seis bibliotecas:

- WiFi.h - Possui funções para realizar tarefas como configuração de acelerômetros, leitura de dados de aceleração, giroscópios e temperatura
- AsyncTCP.h - Uma biblioteca TCP totalmente assíncrona projetada para habilitar um ambiente de rede com várias conexões para o ESP32 do *Espressif*.
- ESPAsyncWebServer.h - Fornece uma maneira fácil de construir um servidor web assíncrono.
- Adafruit MPU6050.h - Responsável por otimizar a escrita do código MPU6050.
- Adafruit Sensor.h - Coleta dados de sensores, analise os dados e tome as medidas adequadas ou envie os dados do sensor para outro sistema para processamento.
- SPIFFS.h - Permite o acesso à memória flash como se fosse um sistema de arquivos normal, como o de um computador. Na Figura 20.1 Demonstra as bibliotecas na IDE Arduíno.

Figura 20.1: Configuração bibliotecas.

```
#include <Arduino.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Arduino_JSON.h>
#include "SPIFFS.h"
```

Fonte: Adaptado Rui Santos, 2021.

Todas as bibliotecas mencionadas são de uso público e se consegue encontrar facilmente no site da GitHub.

Após a definição das bibliotecas partimos para as variáveis e criação de objetos, conforme mostrado na Figura 20.2.

Figura 20.2: Configuração de variáveis e objetos.

```
// Substitua por suas credenciais de rede
const char* NomedaRede = "Euder";
const char* Senha = "24682468";

// Criar objeto AsyncWebServer na porta 80
AsyncWebServer server(80);

// Crie uma fonte de eventos em / events
AsyncEventSource events("/events");

// Variável Json para reter as leituras do sensor
JSONVar readings;

// variáveis de temporizador
unsigned long ultimoTempo = 0;
unsigned long ultimoTempoTemperatura = 0;
unsigned long ultimoTempoAc = 0;

unsigned long tempoEsperaTemperatura = 1000;
unsigned long tempoEsperaAcelerometro = 500;


// Cria um objedo no sensor
Adafruit_MPU6050 mpu;
sensors_event_t a, g, temp;

//variaveis
float X, Y, Z;
float RMSx, RMSy, RMSz;
float temperatura;
float offsetZ =  9.80;
```

Fonte: Adaptado Rui Santos, 2021.

Ainda no início do programa foram criadas cinco funções onde três serão chamadas na função *setup*, e duas na função *loop*. Veja na Figura 20.3 as três primeiras funções a ser chamadas no *setup*.

Figura 20.3: Configuração das funções chamada no *setup*.

1º

```
// Função do sensor MPU6050
void initMPU(){
//Se não encontrar o sensor imprimi falha de sensor
  if (!mpu.begin()) {
    Serial.println("Folha do sensor MPU6050 ");
    while (1) {
      delay(10);
    }
  }
// Sensor encontrado
  Serial.println("MPU6050 Encontrado!");
}
```

2º

```
// Função de acesso a memória flash
void initSPIFFS() {
  if (!SPIFFS.begin()) {
    Serial.println("Ocorreu um erro ao montar SPIFFS");
  }
  Serial.println("SPIFFS montado com sucesso");
}
```

3º

```
// Função de inicialização do Wi-Fi
void initWiFi() {
  WiFi.mode(WIFI_STA);
//Inicia a conecção com a rede local
  WiFi.begin(NomedaRede, Senha);
  Serial.println("");
  Serial.print("Conectando ao Wi-Fi...");
// Caso não encontre entra no laço e arguade 1 segundo depois retorna.
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(1000);
  }
//Encontrado imprime o IP na serial da IDE.
  Serial.println("");
  Serial.println(WiFi.localIP());
}
```

Fonte: Adaptado Rui Santos, 2021.

Na Figura 20.4 São mostrado as outras duas funções chamada no *loop*.

Figura 20.4 Configuração das funções chamado no *loop*.

4º

```
// Função o cáculo de RMS
  String getAccReadings() {
  mpu.getEvent(&a,&g,&temp);
// Leitura dos eixos em RMS
  RMSx = a.acceleration.x*0.707;
  RMSy = a.acceleration.y*0.707;
// Offset no eixo Z desconsiderando a força da gravidade.
  RMSz = ((a.acceleration.z - offsetZ)*0.707) ;

  readings["RMSx"] = String(RMSx);
  readings["RMSy"] = String(RMSy);
  readings["RMSz"] = String(RMSz);

  String accString = JSON.stringify (readings);
  return accString;
}
```

5º

```
String getTemperature(){
  mpu.getEvent(&a,&g,&temp);
  temperatura = temp.temperature;
  return String(temperatura);
}
```

Fonte Adaptado Rui Santos, 2021.

2º. *Setup*;

Depois que a placa é ligada ou reinicializada, a função de *setup* é executada apenas uma vez. Ele pode definir variáveis, inicialização de objeto, definição de status de porta do microcontrolador (*INPUT* OU *OUTPUT*), biblioteca de inicialização, etc. A bibliotecas e funções iniciada são:

- Serial.begin - Responsável pela comunicação com o computador através do monitor serial do IDE.
- Server.begin - Diga ao servidor para começar a escutar as conexões de entrada.
- initWiFi(); Começa iniciação com a função de rede Wi-fi
- initSPIFFS(); Faz busca da memória flash.
- initMPU(); Inicia o sensor acelerômetro.

A comunicação com o HTML e CSS também e chamada na função setup. Confira na Figura 21.

Figura 21: Configuração do *setup*.

```
void setup() {
// Inicia a serial da IDE
  Serial.begin(115200);
// Inicia o Wi-fi
  initWiFi();
// Innicia a comunicação com a memoria flash.
  initSPIFFS();
// Inisica o sensor MPU 6050
  initMPU();

 //Manipular servidor web
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/index.html", "text/html");
  });
  server.serveStatic("/", SPIFFS, "/");
//Manipular servidor web Eventos
  events.onConnect([](AsyncEventSourceClient *client){
    if(client->lastId()){
      Serial.printf("Client reconnected! Last message ID that it got is: %u\n", client->lastId());
    }
    // send event with message "hello!", id current millis
    // and set reconnect delay to 1 second
    client->send("Olá!", NULL, millis(), 10000);
  });
  server.addHandler(&events);

  server.begin();
}
```

Fonte: Adaptado Rui Santos, 2021.

3º. Loop

Repita continuamente enquanto a placa estiver ligada, permitindo que seu programa mude e responda a essas mudanças. Na Figuras 22 exibe a estrutura do código *loop*.

Figura 22: Configuração função *loop*.

```
void loop() {

if ((millis() - ultimoTempoAc) > tempoEsperaAcelerometro) {
    // Envie eventos para o servidor web com as leituras do sensor
    events.send(getAccReadings().c_str(),"accelerometer_readings",millis());
    ultimoTempoAc = millis();
    Serial.println(getAccReadings());
  }
  if ((millis() - ultimoTempoTemperatura) > tempoEsperaTemperatura) {
    // Envie eventos para o servidor web com as leituras do sensor
    events.send(getTemperature().c_str(),"temperature_reading",millis());
    ultimoTempoTemperatura = millis();
  }
}
```

Fonte: Adaptado Rui Santos, 2021.

Na função *loob* se encontra duas leituras sendo enviada á dispositivos móveis. Leitura de aceleração e temperatura, a cada 500 milissegundos é envaido os valores de acelerômetro e a cada 1000 milissegundos da temperatura. O uso do método *send* () passa a ter como argumento o conteúdo que deseja enviar, que neste caso são os dados mencionado. Na Figura 23 mostra estrutura de comunicação entre os hardwares.

Figura 23: Estrutura de comunicação entre hardwares.

Fonte: Autor, 2021.

Entre os dispositivos MPU 6050 e o microcontrolador ESP32 a comunicação dos dados se faz pela serial I2C. Logo a comunicação da microcontrolador com dispositivos móveis se dá por meio de rede Wi-Fi , que faz o uso do protocolo TC/IP para o tráfego de dados.

10. MICROCONTROLADOR *ESPRESSIF*

Conhecido como micro computado de um único chip, o microcontrolador vem revolucionando em projetos de sistemas eletrônicos digitais, sua grande versatilidade e baixo custo, proporciona seu uso no meio industrial.

A *Espressif* é especializada na criação de chips com alto nível de integração, projetando soluções inteligentes que podem ser facilmente fabricadas e implementadas. Com escritórios na China, República Tcheca, Índia, Cingapura e Brasil, a *Espressif Systems* é uma empresa multinacional de semicondutores fundada em 2008, focada no desenvolvimento de soluções de IoT de última geração para Wi-Fi e *Bluetooth*. Criaram as populares séries de chips, módulos e placas de desenvolvimento ESP8266, ESP32, ESP32-S e ESP32-C com o compromisso de oferecer soluções seguras, robustas e com baixo consumo de energia. (*Espressif Systems*, 2021). Na Figura 24 é mostrado um modelo de microcontrolador ESP 32.

Figura 24: Microcontrolador ESP32

Fonte: Filipeflop, 2021.

10.1. Microcontrolador ESP 32

De acordo com a *Espressif Systems* (2021), o ESP32 é um microcontrolador que possui conexões *Bluetooth* e Wi-Fi integradas em sua placa. Isso torna o microcontrolador mais fácil para projetos de IoT tornando possível a troca de informações constante com a rede. Na Figura 25 é exibido as características de um ESP32.

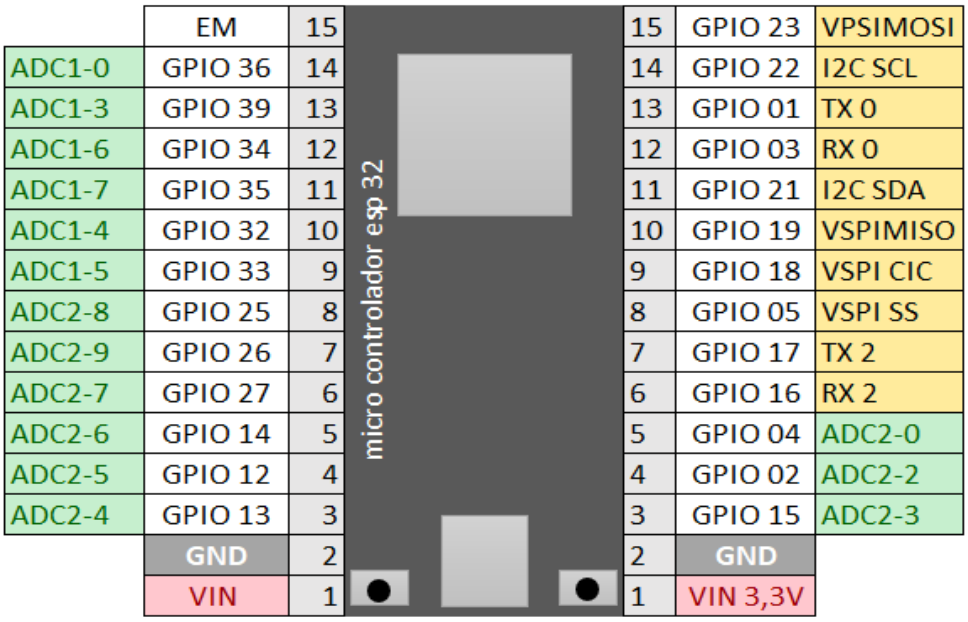
Figura 25: Características do microcontrolador ESP 32.

Alimentação:	2,2V ~ 3,3V DC
Corrente de Consumo:	Média de 80mA
Temperatura de Operação:	-40°C ~ +85°C
Processador:	Xtensa® Dual-Core 32-bit LX6
Processador secundário:	ULP (*Ultra Low Power coprocessor*) 8MHz com consumo de 150uA.
Frequência de Operação:	80MHz ~ 240MHz
Memoria FLASH:	4MB
Memoria RAM:	520KB
Memoria ROM:	448KB
GPIOs (Entradas e Saídas):	34 GPIOS de 3.3V e 12mA.
Conversores ADC (Analógico para Digital):	18 ADC com 12-bit de resolução (4096 bits)
Conversores DAC (Digital para Analógico):	2 ADC com 8-bit de resolução (256 bits)
WiFi:	2,4 GHz,
Bluetooth:	*Bluetooth Low Energy* v4.2 (BLE)
Criptografia:	AES, RSA, SHA e ECC
Segurança:	WPA/WPA2/WPA2-Enterprise/WPS
Protocolos de Rede:	IPv4, IPv6, SSL, TCP/UDP/HTTP/FTP/MQTT
Temporizadores:	4 Timers de 64-bit.
Interfaces de Módulos:	Cartão SD, UART, SPI, SDIO, I2C, LED PWM, Motor PWM, I2S, I2C, IR

Fonte: *Espressif*, 2021.

Na Figura 26 é exibido o mapeamento dos barramentos do ESP32.

Figura 26: Descrição dos barramentos do microcontrolador ESP32



Fonte: Autor, 2021.

As portas GPIOs (*General Purpose Input/Output*) da placa pode fornecer até 12 mA de corrente, são usados como entradas e saídas digitais. O microcontrolador possibilita usar até 16 barramentos como conversor analógico digital. A placa possui portas de 5V e 3,3V, e comunicação serial, TX e RX, etc. *ESPRESSIF SYSTEMS*, 2021)

10.2. Sensor Acelerômetro MPU 6050.

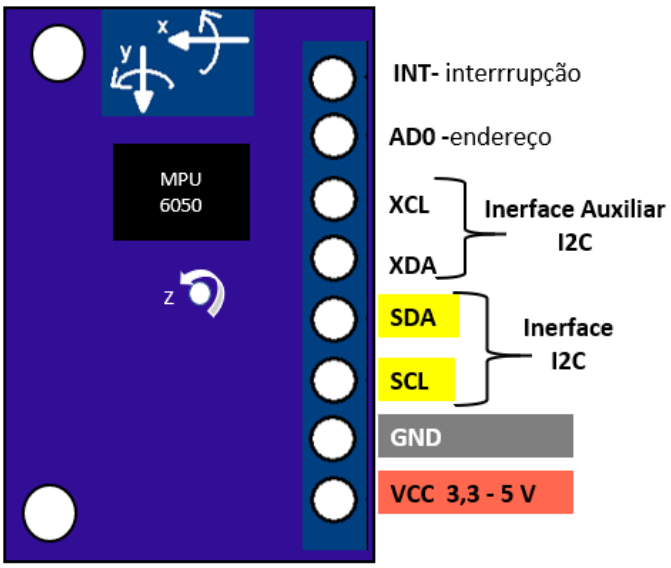
O módulo do sensor MPU 6050 é um dispositivo de rastreamento de movimento de 6 eixos completos. O acelerômetro de três eixos processa o movimento digital em um pacote pequeno. Além disso, possui recurso adicional de sensor de temperatura no chip. Possui uma interface de comunicação I2C para comunicação com os microcontroladores. Veja na figura 27 um modelo de sensor utilizado no protótipo MPU 6050

Figura 27: Sensor MPU 6050

Fonte: Autocore Robótica, 2021.

O MPU 6050 utiliza a tecnologia MEMs (*Micro Electro Mechanical Systems*) e sua unidade de medida é g a força de gravidade e mede a aceleração em quatro faixas de escala completa programáveis sendo seu conversor analógico digital de 16 bits. Na figura 28 se encontra uma representação do acelerômetro com seus barramentos e orientação dos seus eixos X, Y e Z.

Figura 28: Representação acelerômetro MPU 6050.



Fonte: Autor, 2021.

De acordo com Oliveira, 2017. A comunicação I2C é basicamente um circuito integrado para aplicações, os sensores usados no sistema de controle utilizam o protocolo de comunicação do barramento para conectar dispositivos com apenas dois barramentos.

A conexão com EPS32 é muito simples, usando apenas os barramentos analógicos (SDA) e (SCL) e fonte de alimentação, e a tensão da fonte de alimentação pode ser variada entre 3 a 5V. É possível usar os barramentos XDA e XCL para conectar outros dispositivos I2C ao módulo. O endereço I2C pode ser selecionado por meio do pino AD0, o endereço de baixo nível é 0x68 e o endereço de alto nível (3,3 V) é 0x69.

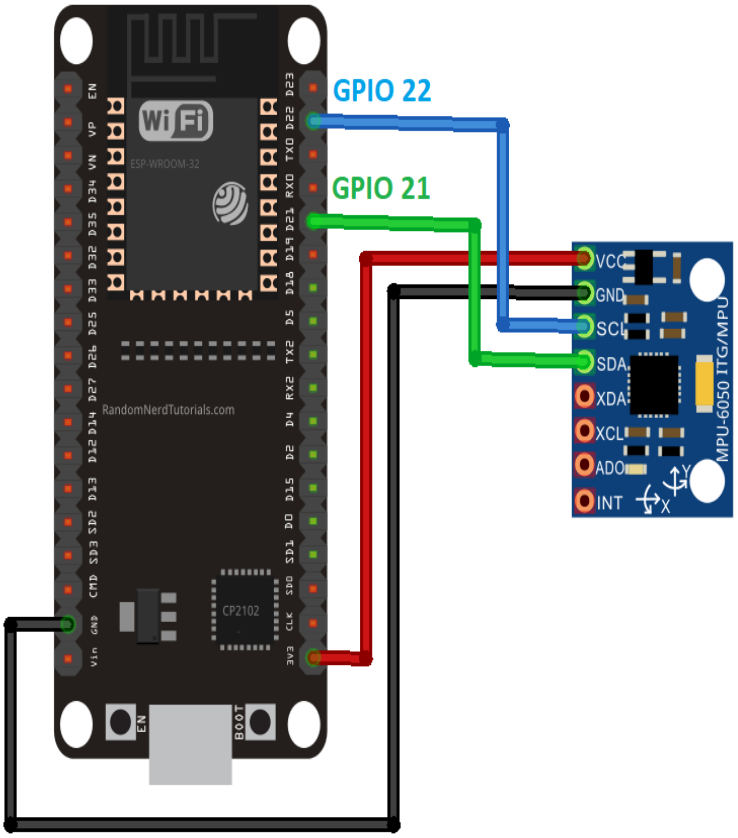
11. RESULTADOS.

Após a conclusão do protótipo, experimentos foram conduzidos para confirmar seu funcionamento, incluindo testes preliminares em uma superfície plana e testes em um motor elétrico de 1.5 CV, e o valor efetivo (RMS) foi calculado.

11.1.Ligação do ESP32 com o Acelerômetro MPU 6050

Para se fazer a conexão entre o ESP32 e o acelerômetro deve se conectar o barramento GND e VCC 3,3 V do microcontrolador com o GND e VCC do acelerômetro MPU6050. Após a conexão do barramento de alimentação se deve segui interligando os barramentos SDA e SCL do acelerômetro com os barramentos GPIO 21 e 22 do microcontrolador, conforme mostrado na Figura 29.

Figura 28: Ligação do ESP32 com acelerômetro MPU6050.



Fonte: Random, 2021.

11.2. Tratamento e Processamento de Sinais de Entradas.

Cada sinal é quantificado por expressões matemáticas e variáveis de entrada do sistema de leitura. O sistema é descrito como um conjunto de elementos que interagem entre si para realizar uma tarefa específica, geralmente composta por uma função, razão pela qual o sistema opera. Variáveis armazenam informações processadas por elementos interativos que trocam informações para implementar objetos do sistema. A conexão permite a comunicação entre os elementos. (LA VEGA, 2019).

Usando o IDE do Arduino, pode-se monitorar os dados enviados pela porta serial. De acordo com a relação de calibração selecionada, o valor obtido é armazenado junto com o valor de aceleração real (FREESCALE, 2015). A relação de escala para a calibração se dá conforme a figura 29.

Figura 29: Relação de escala.

2g	16,384
4g	8,192
8g	4,096
16g	2,048

Fonte: InvenSense, 2012.

Neste projeto, a escala selecionada para o coeficiente de calibração é de 2g. Após definir o ajuste de calibração, devemos converter os dados recebidos em bits para aceleração (m/s^2), conforme descrito nas Equações 03 e 04. A Equação 03 define a resolução do valor da gravidade sobre o valor de calibração.

$$Resolução = \frac{9{,}81}{calibração} \quad (03)$$

Em seguida, é definido a aceleração, a resolução encontrada é multiplicada pelo valor fixo do valor lido durante a leitura.

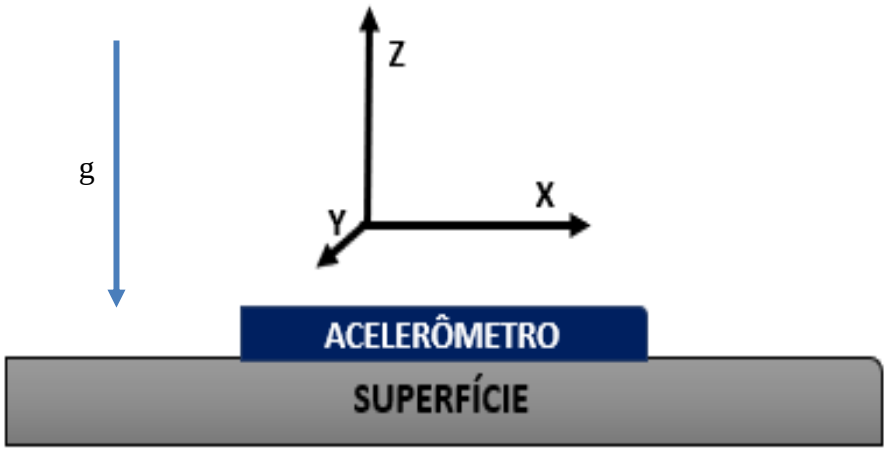
$$a = Resolução\ x\ Valor\ Lido \quad (04)$$

Feita todo o procedimento de escolha, tratamento de dados, cálculo da resolução e a aceleração, os valores são testados sobre uma superfície plana.

11.3. Teste do Acelerômetro em ma Superfície Plana.

Para a obtenção do experimento de leitura do sinal do motor, o acelerômetro será testado sobre uma superfície plana em repouso para confirmação dos seus valores. O posicionamento, é mostrado na Figura 30.

Figura 30: Acelerômetro sobre superfície.



Fonte: Autor, 2021.

Com o acelerômetro posicionado sobre a superfície plana, os valores encontrados foram de 16138,75, assim é aplicando a Equação (3), para encontrar a resolução temos.

$$Resolução = \frac{9{,}81}{16138{,}75} = 0{,}000607853$$

Com os valores da resolução definidos, partimos em aplicar Equação (4) para encontrar a aceleração.

$$a = 0{,}000607853 \; x \; Valor \; Lido$$

Considerando apenas a gravidade, observa-se que há uma força de gravidade atuando no eixo Z, que se encontra em um ângulo de 90º graus em relação à superfície, a atuação da força da gravidade pode ser vista na Figura 31.

Figura 31: Força da gravidade nos eixos X,Y e Z.


```
{"X":"0.13","Y":"-0.13","Z":"9.70"}
{"X":"0.09","Y":"-0.16","Z":"9.53"}
{"X":"0.15","Y":"-0.16","Z":"9.66"}
{"X":"0.15","Y":"-0.11","Z":"9.68"}
{"X":"0.07","Y":"-0.11","Z":"9.64"}
{"X":"0.13","Y":"-0.19","Z":"9.51"}
{"X":"0.13","Y":"-0.19","Z":"9.59"}
```


Fonte Autor, 2021.

Como mencionado, pode notar a gravidade agindo no eixo Z, onde os eixos X e Y devem permanecer em 0. Devido à alta sensibilidade, o sensor pode capturar pequenas alterações mesmo em estado estático. A leitura dos valores negativos se dá por decorrer de um sensor linear.

Para análises futuras, a força da aceleração gravitacional será matematicamente ignorada.

11.4. Teste em um Motor de 1.5 CV

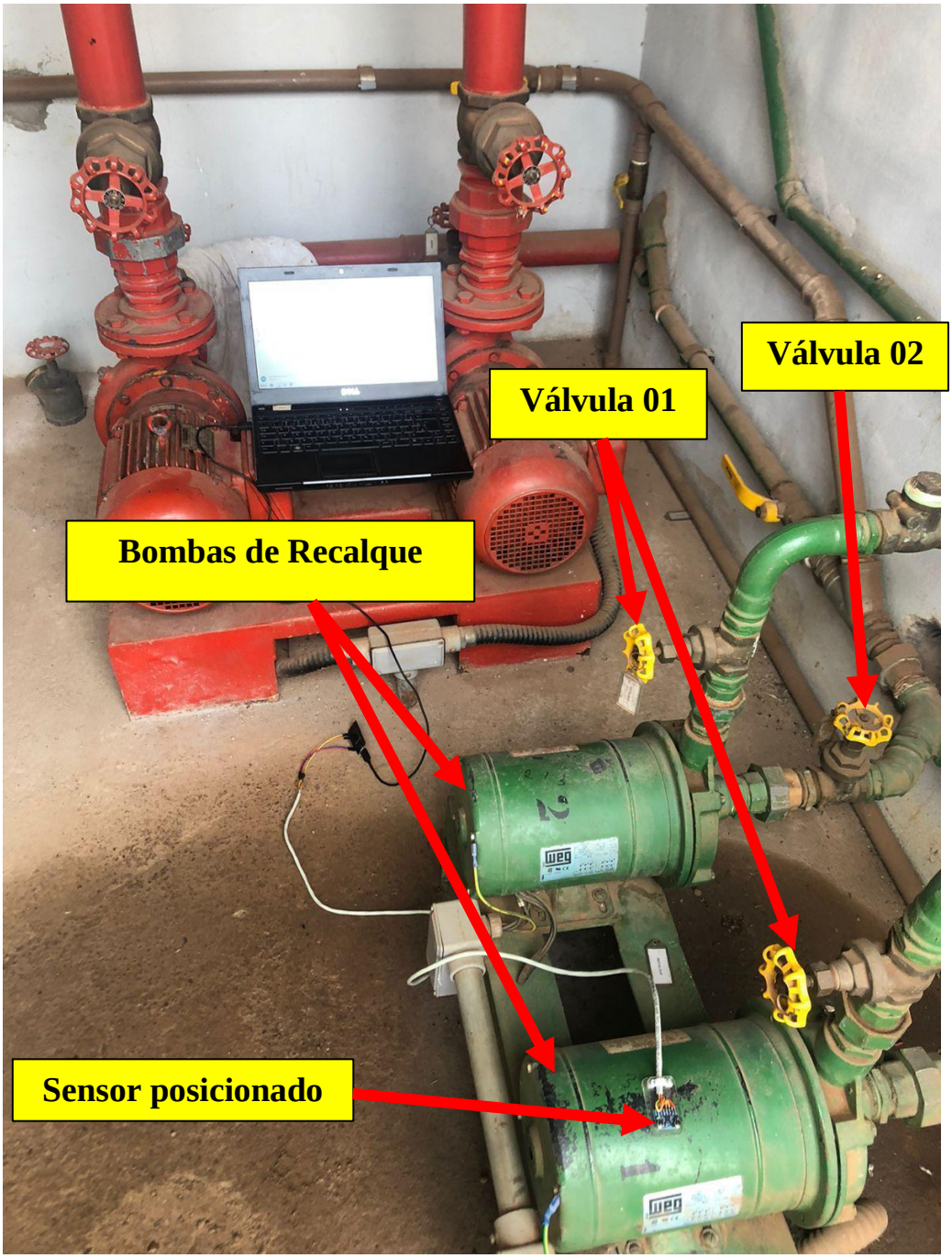
O motor analisado é um motor WEG trifásico de 1.5 CV com velocidade de 3.450 rpm, tensão de 380 V, frequência de 60 Hz e fator de potência de 1.20, conforme exibido na Figura 32 .

Figura 32: Placa motor de indução trifásico 1.5CV.

Fonte: Autor, 2021.

O motor atua como uma bomba de recalque no sistema, bombeando fluido de uma posição inferior para uma posição superior. O tubo do sistema de descarga está localizado diretamente após a bomba de sucção e está pronto para transportar fluido para a parte superior, conforme mostrado na Figura 33.

Figura 33: Sala de Máquinas

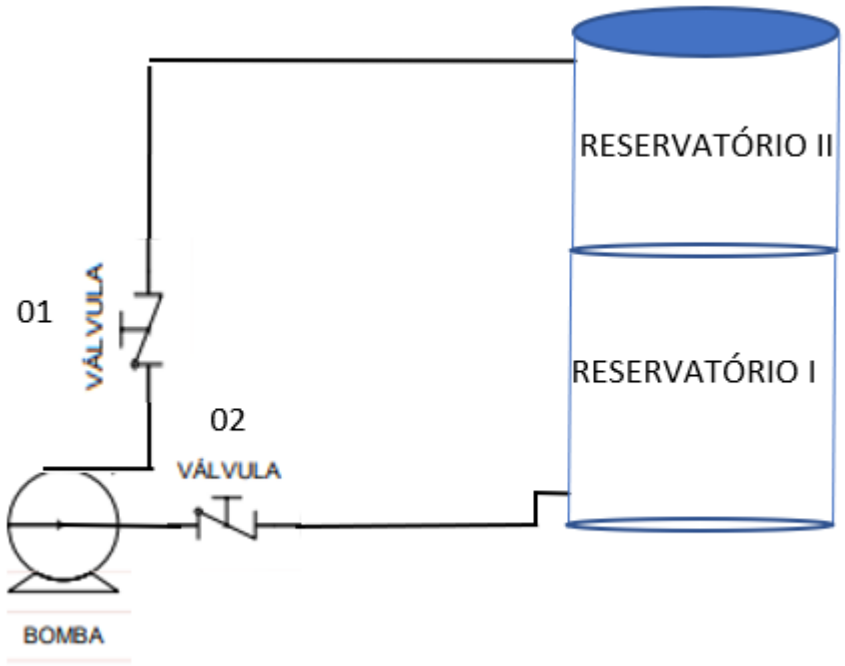


Fonte: Autor, 2021.

Conforme mostrado na figura 33, o sistema contém duas válvulas, descritas como válvula 01 e válvula 02. As válvulas são responsáveis pela passagem do fluido, onde a válvula 02 determina a chegada do fluido até a bomba, e a válvula 01 por permitir passagem do fluido até a caixa d´água.

Na Figura 34 se se têm uma representação do funcionamento do motor elétrico aplicado a um sistema de recalque.

Figura 34: Sistema de recalque.



Fonte: Autor, 2021.

O teste coletará dados em duas condições, primeiro a válvula 02 estará na condição aberta que logo após feita a coleta dos dados a válvula será fechada simulando uma condição de anormalidade em seu funcionamento.

11.5.RMS (*Root Mean Square*)

A partir deste momento iremos trabalhar com o processamento de sinais em RMS. O valor RMS (*Root Mean Square*) que em português se traduz como Raiz Quadrada Média aplicado a um sistema de vibração é uma onda periódica que faz a mensuração do nível de sinal mais relevante (GARCIA, 2005). Para encontrar o valor RMS da aceleração se utilizou a equação (01).

$$RMS = \sqrt{\frac{1}{2\pi}\int_0^{2\pi}(a.sin(wt))^2 \; dwt} = \quad (01)$$

$$RMS = \frac{a}{\sqrt{2}} \quad (01)$$

De acordo com a pesquisa de Budik et al. (2016), ao se obter o valor RMS de referência, o valor da máquina que representa o estado do dispositivo pode ser comparado periodicamente, e o limite de alarme de cada dispositivo pode ser definido estatisticamente. Este método envolve fazer

medições RMS suficientes para realizar uma análise estatística das variáveis fornecidas na Equação 05.

$$D_i = |RMS_i - RMS_{i-1}| \quad (05)$$

Com isso se pode realizar o limite de alarme utilizando a equação (06).

$$L = \overline{RMS} + 2{,}66D \quad (06)$$

Dito isso além da ISO 10816, que especifica as condições e procedimentos gerais para medir e avaliar a vibração, a análise de sinais específicos da máquina é mais adequada para definir limites de alarme. Analisando graficamente o comportamento da válvula 01 ao ser aberta, podemos definir esses sinais como parâmetros para o comportamento normal do dispositivo.

11.6. Funcionamento do Motor com a Válvula Aberta.

Na primeira análise dos dados, a válvula 02 está em um estado aberto, o sensor está posicionado sobre o motor, o motor é colocado em estado de operação. Consulte a Figura 35 para obter informações sobre como o sensor é posicionado no motor.

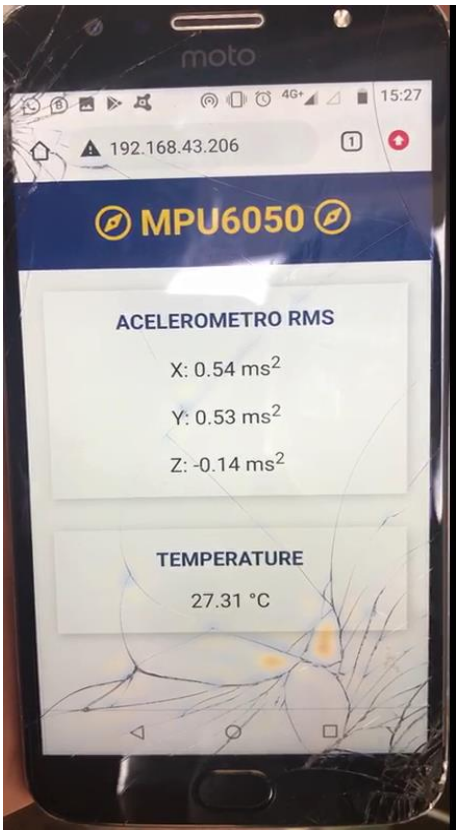
Figura 35: Posicionamento do sensor ao motor.

Fonte: Autor, 2021.

A vibração gerada pelo motor em funcionamento é coletada pelo sensor e enviada ao

microcontrolador, sendo então esses valores exibidos na interface do *smartphone*. O sensor também possibilita se fazer a leitura de temperatura ambiente. Sendo assim temos as seguistes leitura na interface do *smartphone*, leitura de vibração e temperatura, conforme Figura 36.

Figura 36: Dados coletados pelo acelerômetro exibidos na *interface do smartphone.*



Fonte: Autor, 2021

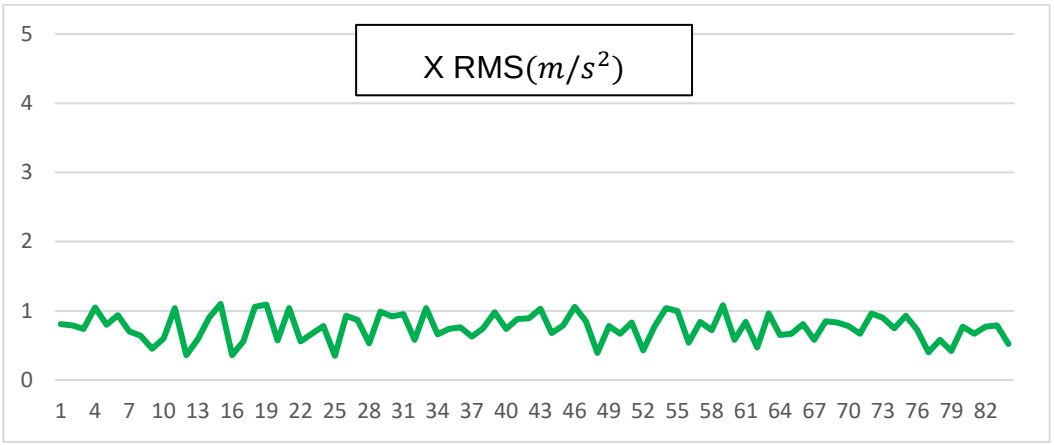
Conforme a Figura 36, os valores obtidos foram:

- X = 0,54 m/s^2
- Y = 0,53 m/s^2
- Z = 0,14 m/s^2

O motor é monitorado em tempo real na frequência de 21 Hz, sendo os dados atualizados a cada 500 milissegundos O valor exibido indica que o motor está em operação normal (válvula 01 aberta).

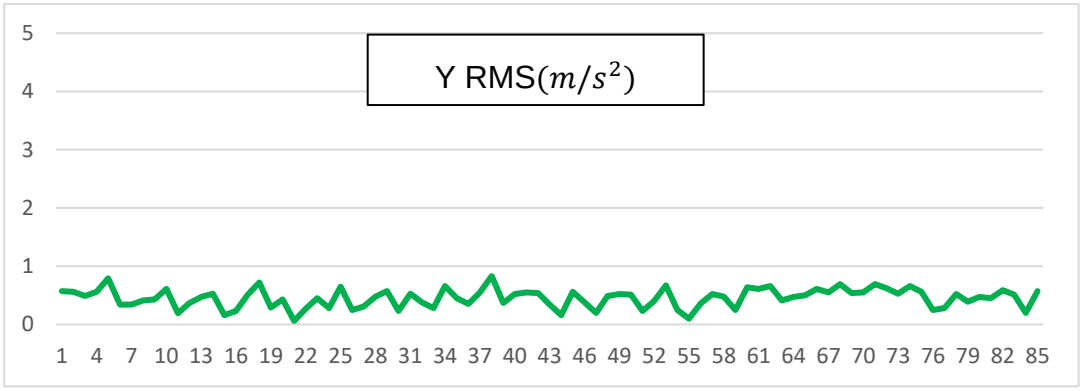
Além da análise na interface do *smartphone*, 85 amostras foram coletadas para análise gráfica. Nas Figuras 37, 38 e 39, os dados podem ser analisados graficamente em valores absolutos.

Figura 37: Sinal coletado no eixo X, com a válvula aberta.



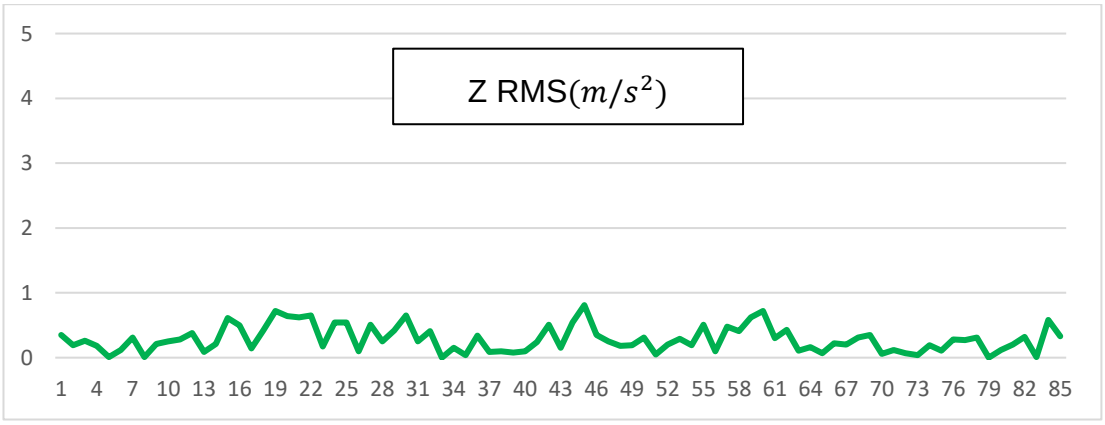
Fonte Autor. 2021.

Figura 38: Sinal coletado no eixo Y, com a válvula aberta.



Fonte: Autor, 2021.

Figura 39: Sinal coletado no eixo Z, com a válvula aberta.



Fonte: Autor, 2021.

Com os dados de estado normal de operação do motor encontrado é possível definir o limite de alarme para cada eixo X, Y e Z de acordo com Budik et al (2016), que propõe determinar o limite de alarme conforme a Equação (05) e (06).

$$D_x = |RMS_{85} - RMS_{85-1}| = 0{,}15 \quad (5)$$

$$D_y = |RMS_{85} - RMS_{85-1}| = 0{,}37 \quad (5)$$

$$D_z = |RMS_{85} - RMS_{85-1}| = 0{,}25 \quad (5)$$

$$Lx = \overline{RMS}\, x + 2{,}66Dx = 1{,}157117647 \quad (6)$$

$$Ly = \overline{RMS}y + 2{,}66\, Dy = 1{,}437141176 \quad (6)$$

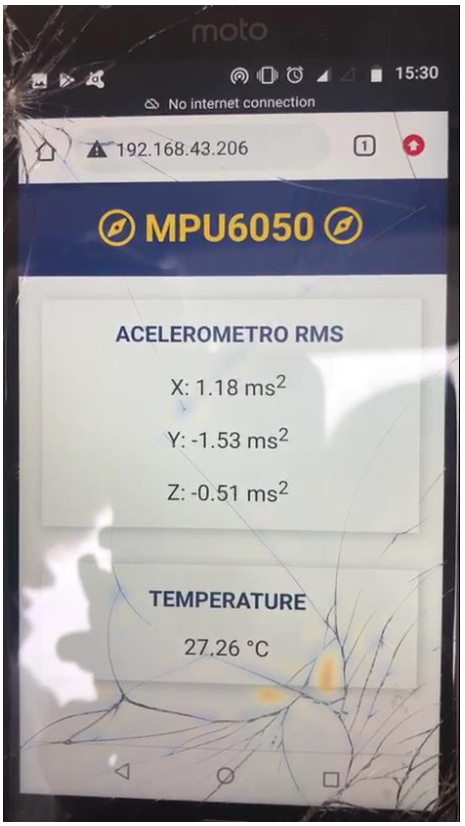
$$Lz = \overline{RMS}z + 2{,}66Dz = 0{,}949117647 \quad (6)$$

Valores de limites de alarme definidos, parte-se para se fazer outra análise com a bomba de recalque na condição de estado anormal, ou seja, válvula 02 fechada.

11.7. Funcionamento do Motor Com a Válvula Fechada.

Na cena em que a válvula 02 é fechada e o motor em funcionamento, pode se ouvir aumento de “zunido” vindo do motor, onde o mesmo foi colocado em seu estado anormal de funcionamento. Perante a esta condição foi realizada o monitoramento dos valores pela a interface do *smartphone*. Na Figura 40 tem-se os valores em exibição.

Figura 40: Valores exibido na interface do *smartphone.*



Fonte: Autor, 2021.

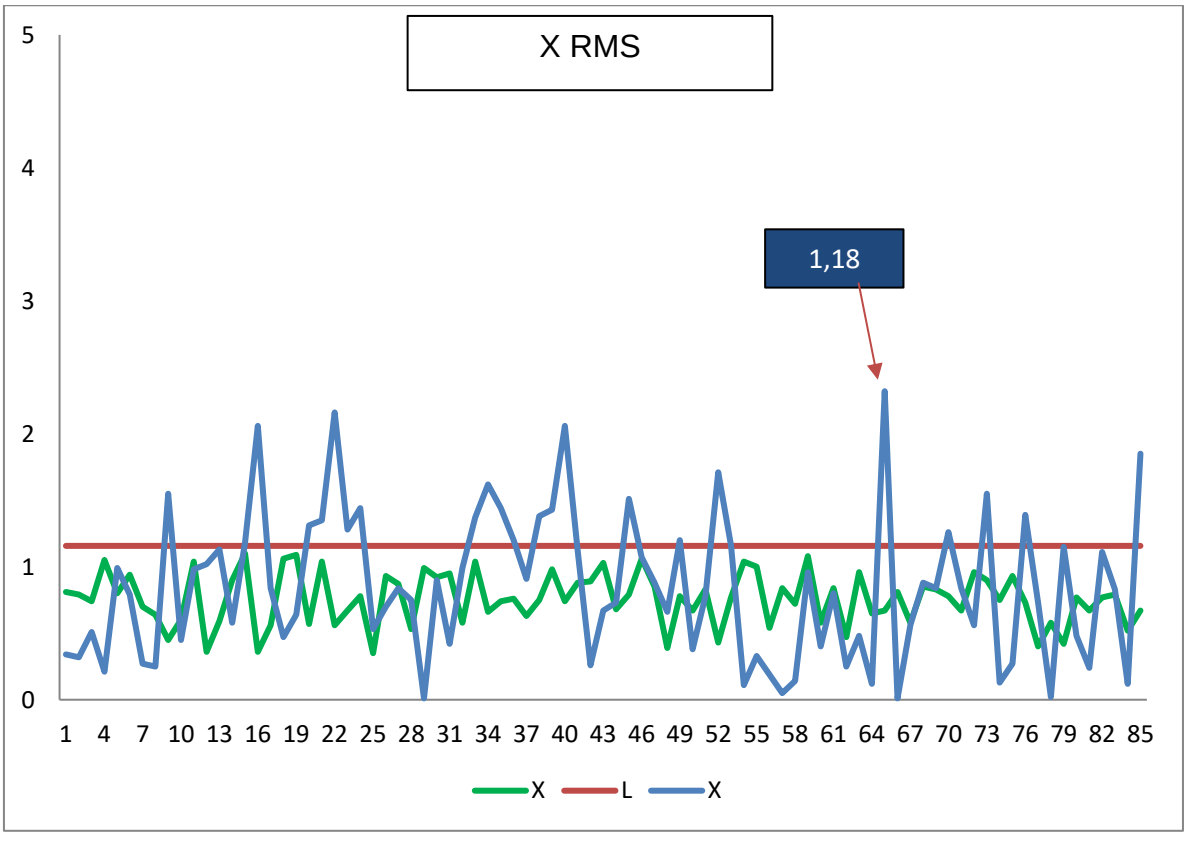
Conforme mostrado na Figura 40, os valores obtidos foram:

- X = 1.18 m/s^2
- Y = -1.53 m/s^2
- Z = -0,51 m/s^2

Inserindo a bomba em estado de trabalho anormal, se pode notar pela interface do *smartphone* que á um aumento dos valores indicando mudança no comportamento dos dados ao fechar a válvula 02. Assim como realizada no procedimento com a bomba em estado normal, foram feitas coletadas de 85 amostra com a bomba em estado anormal, no intuito de se fazer a comparação gráfica.

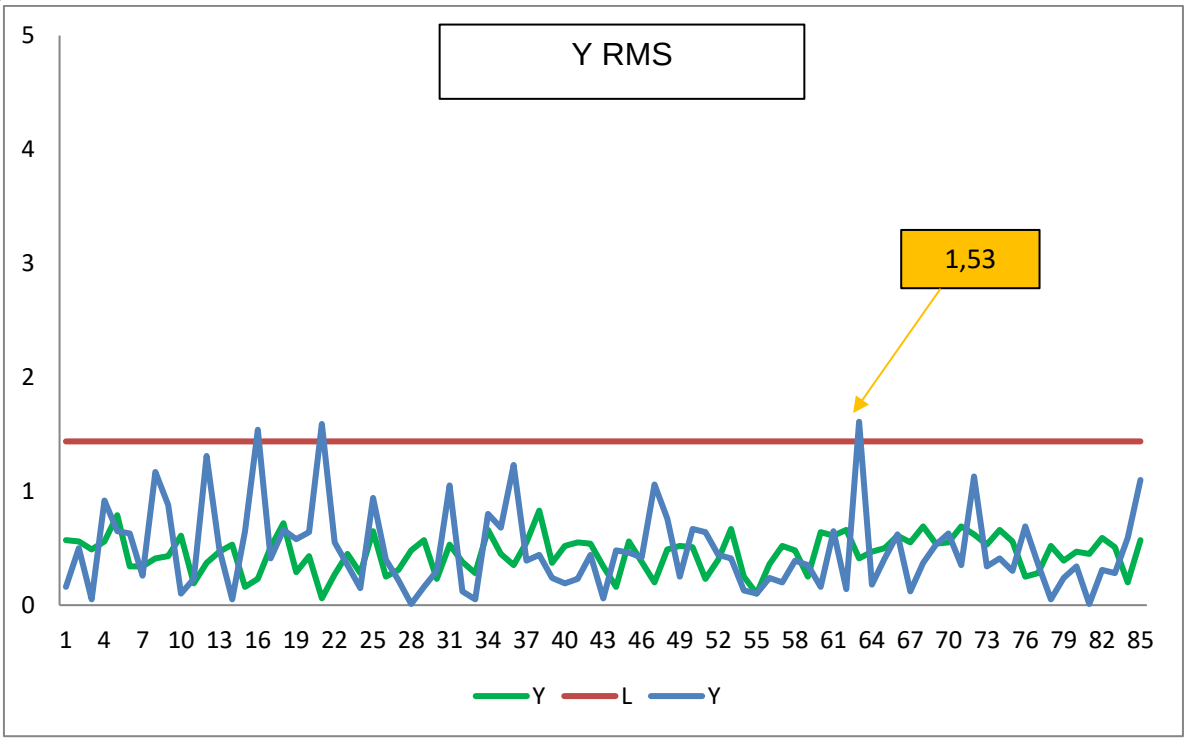
Usando o limite definido, o mesmo é inserido no gráfico junto com outros dados sob condições operacionais normais (válvula aberta) e condições operacionais anormais (válvula fechada). As Figuras 41, 42 e 43, mostra o comportamento do sinal da bomba em condições normais (verde) e anormais (azul). O limite de alarme estabelecido encontra-se em (laranja).

Figura 41: Sinal coletado no eixo X, com a válvula fechada.



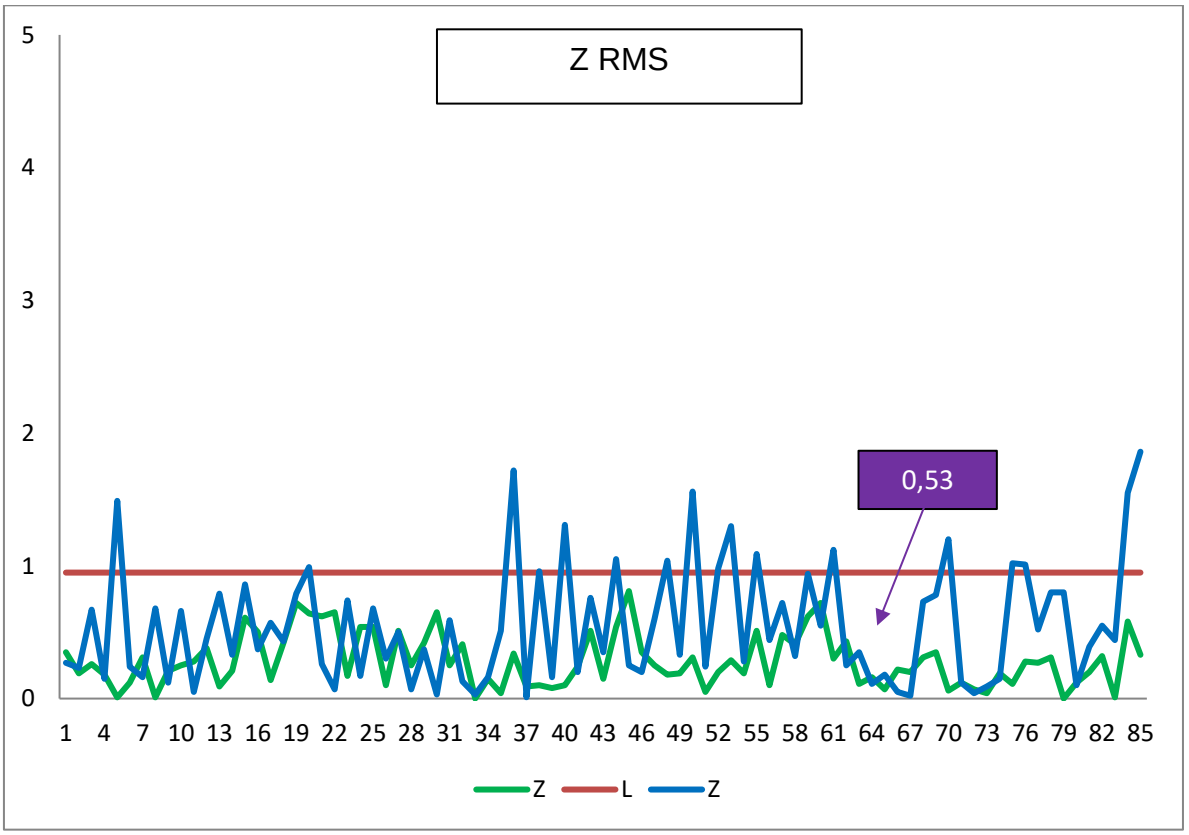
Fonte: Autor

Figura 42: Sinal coletado no eixo Y, com a válvula fechada.



Fonte, Autor 2021.

Figura 43: Sinal coletado no eixo Z, com a válvula fechada.



Fonte: Autor, 2021.

Vale ressaltar que na Figura 40 os valores mostrados na interface *do smartphone*, são similares aos das Figuras 41, 42 e 43 onde os mesmo se encontra destacados no gráfico, o que mostra a veracidade da leitura.

Analisando os gráficos 41, 42 e 43, pode-se observar que nos eixos X, Y e Z o limite de alarme em (laranja) é excedido pelo sinal em(azul). Sinal este de operação do motor em estado anormal. Sendo assim obtendo os valores de operação do motor, no estado normal de operação, é possível determinar o limite de alarme, e fazer a tomada de decisão com a leitura dos valores exibido no *smartphone*.

.

12. CONCLUSÃO.

O protótipo para o monitoramento remoto continuo do motor de indução foi implementado com sucesso, destacando-se a utilização de *smartphones* para visualização dos valores de vibração atendendo aos objetivos do processamento de dados. Através da utilização de limites de alarmes, foi possível detectar condições anormais no funcionamento do motor, facilitando assim a tomada de decisões sobre o estado do equipamento.

A aplicação da tecnologia móvel na gestão da manutenção é considerada positiva, pois as melhorias resultantes trazem qualidade, confiabilidade e agilidade às intervenções de manutenção, o que é um passo importante para a inserção das empresas na Indústria 4.0.

Como trabalhos futuros pode ser realizado estudos da implementação da lógica Fuzzy facilitando a tomada de decisão sobre o equipamento sem a necessidade de uma mão de obra especializada. Também há margem de melhora no código, se buscando a integração do hardware com plataforma de monitoramento, por exemplo: ScadaBR. Os estudos foram realizados sobre o domínio do tempo sendo assim é possível se fazer a aplicação para o domínio da frequência e a análise dos dados. Como visto há alguns ruídos no sensor onde a aplicação de um filtro pode vir a reduzi-lo, exemplo: Filtro de Kalman.

13. SUGESTÕES DE TRABALHOS FUTUROS

Algumas sugestões para trabalhos futuros são apresentadas a seguir:

- Implementação de um filtro de Kalman.
- Implementação da lógica nebulosa (Fuzzy).
- Implementação domínio da frequência (Transformada de Fourier).
- Implementação em plataforma de monitoramento (ScadaBR).

14. REFERÊNCIAS

NBR, ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 1994 **Confiabilidade e mantenabilidade**. Disponível em: <https://cupdf.com/document/nbr-5462.html>. Acesso em 28 junh. 2021.

ASSIS, F.; NOGUEIRA, J.C. **Aspectos da manutenção dos equipamentos científicos da Universidade de Brasília**. Dissertação apresentada na Faculdade de Economia, Administração, Contabilidade e Ciência da Informação e Documentação, Universidade de Brasília. 2006

JUNIOR, J.J.M.G**. Redução de Custo na Manutenção de Ativos no Ambiente Industrial**. UNIVERSIDADE ESTADUAL PAULISTA, 2016.

JÚNIOR, **Geraldo Motta Azevedo et al. Estudo sobre a manutenção preditiva em motores trifásicos através da análise de vibrações**. Projectus, Rio de Janeiro, v. 1, n. 4, p. 70-83, dez. 2016. Disponível em:<http://apl.unisuam.edu.br/revistas/index.php/projectus/article/view/1642>. Acesso em: 28 abr. 2021.

ARKTIS. (19 de Janeiro de 2016). ***Arktis- Industry 4.0: Everything you need to know/Entrepreneurial Insights***. Disponível em: <http://arktis.com.br/a-quartarevolucao-da-industria/>. Acesso em 30 jun. 2021

EZRA, ***Oren. Achieving Manufacturing Excellence with Predictive Maintenance and Machine Learning. In: Industry*** 4.0 Insights, [S.I.], 27 mar. 2018. Disponível em: https://blog.seebo.com/predictivemaintenance-machine-learning/. Acesso em: 26 out. 2021.

FITZGERALD, Arthur E. MÁQUINAS ELÉTRICAS**: Com introdução à eletrônica de potência.** 6. ed. Bookman, 2006. 648 p.

CAPELLI, A**. Automação Industrial: controle do movimento e processos contínuos. 3. ed. São Paulo: Érica**, 2013.

SILVA, V.A.D. **Detecção de falhas em motores elétricos através das máquinas de**

vetores de suporte. 2012 . Dissertação Universidade Estadual de Campinas.

BEZERRA, Roberto de Araújo**, Detecção de Falhas em Rolamentos por Análise de Vibração,** Campinas, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2004. 1xx p. Tese (Doutorado).

THOMSON, W.T.; FENGER, M. ***Current singature analysis to detect induction motor faults. IEEE Industry Applications*** Magazine.vol.7, issue4, pp. 26-34,2001.

BONNETT, Austin H.; YUNG, ***Chuck. Increased efficiency versus increased reliability. IEEE Industry Applications*** Magazine, v. 14, p. 29–36, Jan 2008.

GONGORA, W. S. et al. ***Neural approach to fault detection in three-phase induction motors. IEEE Latin America Transactions***, New York, v. 14, n. 3, p. 1279-1288, 2016

TAVARES, L. A. **– Administração Moderna da Manutenção**. By Novo Polo Publicações e Assessoria Ltda. Rio de Janeiro, 1999

TSANG. A. H. C**.** ***Condition-based Maintenance: Tools and Decision Making. In: Journal of Quality in Maintenance Engineering***, v. 1, n. 3, p. 3-17. ISSN 1355-2511. MCB University Press. 1995.

XENOS, H. G. – **Gerenciando a manutenção produtiva. Belo Horizonte**. Editora de Desenvolvimento Gerencial, 1998, 302 p.

SELLITTO, M. **Formulação estratégica da manutenção industrial com base na confiabilidade dos equipamentos**. Produção, S. Paulo, v. 15, n. 1, p. 044-059, 2005

FINCH, B.J. e GILBERT, J.P. (1986). ***Developing maintenance craft labor efficiency through an integrated planning and control system***: A prescriptive model 1986.

COLLACOTT, R. A., ***Vibration Monitoring and Diagnosis – Techniques for Costeffective Plant Maintenance***, 2 ed. New York, John Wiley& Sons, 1999

GONÇALVES NETO, W. et al. **Manutenção preditiva através de análises em equipamentos rotativos monitorado por sensores de vibração**. Revista de Controle e Automação, Campinas, v. 1, n. 1, p. 1-8, 2013.

MOBLEY, R. e KEITH, R.K. ***Vibration Fundamentals. Butterworth-Heinemann***, 1999.

GIRDHAR, P.; SCHEFFER, C. ***Practical Machinery Vibration Analysis and Predictive Maintenance.Newnes,*** Oxford, 2004. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9780750662758500011>. Acesso em: 13 set. 2021

MARAN, M**. Manutenção baseada em condição aplicada a um sistema de ar condicionado como requisito para sustentabilidade de edifício de escritórios**. São Paulo: EPUSP, 2012. 22 p. (Boletim Técnico da Escola Politécnica da USP, Departamento de Engenharia de Construção Civil, BT/PCC/576). Disponível em: <http://www.pcc.usp.br/files/text/publications/BT_00576.pdf.> Acesso em: 13 Jun. 2015.

JUNIOR, J.J.M.G. **Redução de Custo na Manutenção de Ativos no Ambiente Industrial**. UNIVERSIDADE ESTADUAL PAULISTA, 2016.

GARCIA, Maurício Sanches**. Análise de defeitos em sistemas mecânicos rotativos a partir da monitoração de vibrações**. Tese de Doutorado em Engenharia Mecânica, Universidade Federal do Rio de Janeiro, 2005.

RAO, ***Singiresu S. Mechanical Vibrations***. 4.ed. São Paulo: Pearson Prentice Hall, 2008

KARDEC, Alan; NASCIF, **Julio. Manutenção função estratégica**. Rio de Janeiro, Qualitymark, 2015.

ANTONIOLLI E. A. **Estudo Comparativo De Técnicas De Medição E Análise De Vibrações Para A Manutenção Preditiva Em Mancais De Rolamentos**. Disponível em: <https://nedip.ufsc.br/uploads/file/dissertacao_edilar.pdf>. Acesso em 14/set. 2021.

NEPOMUCENO, Lauro Xavier. **Técnicas de Manutenção Preditiva**. São Paulo: Edgard Blucher Ltda., 1989. 501 p.

GUIDE for ***Induction Machinery Maintenance Testing and Failure Analysis***. IEEE Std 1415-2006, p. c1–58, April 2007.

ISO 10816-1/ISO. Abstract to ISO 10816-1. ISO, 1995. Disponível em: <https://www.iso.org/standard/18866.html> . Acesso em: 25 out. 2021.

RIPPER, G.P. **Padronização Primária Em Metrologia De Vibrações**. 2005. Disponível em: <https://pt.scribd.com/document/327754698/Padronizacao-Primaria-Em-Metrologia-de-Vibracoes>. Acesso em: 24 out. 2021.

SEQUEIRA, C. **Sensores para medições de vibrações mecânicas-acelerômetros**. Revista Manutenção, Porto, v. 116, p. 4-6, 2013.

O'NEAL, C.B. "***Challenges in the packaging of MEMS," Advanced Packaging Materials: Processes***, Properties and Interfaces, 1999. Proceedings. International Symposium on , vol., no., pp.41,47, 14-17 Mar 1999. Disponível em:<https://ieeexplore.ieee.org/document/757284> . Acesso em: 15 jan 2021

TEZ, S.; AKIN, T. ***Fabrication of a sandwich type three axis capacitive MEMS accelerometer.*** Sensors, 2013 IEEE , vol., no., pp.1,4, 3-6 Nov. 2013. Disponível em: 96 <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6688598&isnumber=6688115>. Acesso em: 08 julh 2021.

MAURA, S**. Os caminhos para a transformação das empresas no futuro. In: Indústria 4.0**. [S.I.], 12 out. 2019. Disponível em: https://www.industria40.ind.br/artigo/18906-industria-40-os-caminhospara-a-transformacao-das-empresas-para-o-futuro. Acesso em: 13 nov. 2019.

SILVA, R. M. DA; SANTOS FILHO, D. J.; MIYAGI, P. E**. Modelagem de Sistema de Controle da Indústria 4.0 Baseada em Holon,** Agente, Rede de Petri e Arquitetura Orientada a Serviços. XII Simpósio Brasileiro de Automação Inteligente. Natal. 2015.

CARDOSO, M.O.INDÚSTRIA 4.0: **a quarta revolução industrial**, UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ, 2016. Disponivel em:

<http://riut.utfpr.edu.br/jspui/bitstream/1/17086/1/CT_CEAUT_2015_08.pdf>. Acesso 20 out. 2021.

J.V.SOBREIRA **Desafios para a Manutenção na perspetiva da Indústria** 4.0, INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA, 2018.

SOUTO, L.M. **Especificação E Implementação De Componentes Para Modelar Redes Locais Sem Fio** Ad Hoc Padrão IEEE 802.11, UFCG, 2005

OLIVEIRA, M.J.G. **Desenvolvimento de uma Plataforma para Internet das Coisas Baseada em Wi-Fi**, FEUP 2017.

SILVA, R.M. **Desenvolvimento de Serviço em Redes TCP/IP** - Projeto de um Servidor de Mode. USP, 1992.

RAMOS, J. et al. **Iniciativa para robótica pedagógica aberta e de baixo custo para inclusão social e digital no brasil**. Simpósio Brasileiro de Automação Inteligente (SBAI), 2007. Citado na página 27.

ESPRESSIF. ***Committed to technological innovation for the benefit of both our society and the planet***. Disponível em: <https://www.espressif.com/en/products/socs/esp32>. acesso em 26 de maio de 2021.

VEGA , L.A. **Alexandre Santos de. Apostila de Teoria para Fundamentos de Processamento Digital de Sinais**. 2019. Disponível em: <https://pt.scribd.com/document/377255801/Apostila-de-Teoria-para-Processamento-Digital-de-Sinais-Versao-A2018M04D16 >. Acesso em: 28 set. 2021.

RUI SANTOS**.** ***ESP32 Web Server with MPU-6050 Accelerometer and Gyroscope (3D object representation). Random Nerd Tutorials***, 21/01/2021. Disponível em: <https://randomnerdtutorials.com/esp32-mpu-6050-web-server>. Acesso em: 06 out.2021.

FREESCALE. ***MMA8451Q, 3-Axis, 14-bit/8-bit Digital Accelerometer***. Disponível em: . Acesso em setembro de 2015.

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

SILVA, E.P. **A Transição Da Manutenção Industrial Para O Modelo Do Novo Paradigma Da Indústria 4.0**, UNIP, 2018.

APÊNDICE A – Código do Projeto Acelerômetro MPU6050, ESP32.

```
/*
PROJETO FINAL: EUDER ALVES COSTA
* BACHARELADO DE ENGENHARIA ELETRICA - UNIS/MG
* euder.costa@alunos.unis.edu.br
* 08/2020
*/
#include<Wire.h>//Biblioteca para comunicação I2C
const int MPU=0x68;//Endereço do sensor
int16_t AcX,AcY,AcZ; //Variaveis guarda os valores medidos
float g = 9.8;// Definição do valor da gravidadee
float resolucao = 0.000604043; // valor da resolução de calibração

void setup(){
  Serial.begin(115200); //Inicia a comunicaçao serial (para exibir os valores lidos)
  Wire.begin(); //Inicia a comunicação I2C
  Wire.beginTransmission(MPU); //Começa a transmissao de dados para o sensor
  Wire.write(0x6B); // registrador PWR_MGMT_1
  Wire.write(0); // Manda 0 e "acorda" o MPU 6050
  Wire.endTransmission(true);
}

void loop(){
  Wire.beginTransmission(MPU); //Começa a transmissao de dados para o sensor
  Wire.write(0x3B); // registrador dos dados medidos (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU,14,true); // faz um "pedido" para ler 14 registradores, que serão os registrados com os dados medidos

  AcX=Wire.read()<<8|Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcY=Wire.read()<<8|Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcZ=Wire.read()<<8|Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)

 float gX = resolucao*AcX;
 float gY = resolucao*AcY;
 float gZ = resolucao*AcZ;

  //Agora escreve os valores no monitor serial
 Serial.print("AcX = "); Serial.print(gX);
 Serial.print(" | AcY = "); Serial.print(gY);
 Serial.print(" | AcZ = "); Serial.println(gZ-g);

 // Serial.print("     "); Serial.println(setXGyroOffsetTC);
  delay(333);
}
```

ANEXO I – Código do projeto Acelerômetro MPU6050, ESP32 e WebServ.

```
/*

Fonte: Adaptado Rui Santos, 2021.
 Detalhes completos do projeto em https://RandomNerdTutorials.com/esp32-mpu-6050-web-server/
*/

#include <Arduino.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Arduino_JSON.h>
#include "SPIFFS.h"

// Substitua por suas credenciais de rede
const char* NomedaRede = "Euder";
const char* Senha = "24682468";

// Criar objeto AsyncWebServer na porta 80
AsyncWebServer server(80);

// Crie uma fonte de eventos em / events
AsyncEventSource events("/events");

// Variável Json para reter as leituras do sensor
JSONVar readings;

// variáveis de temporizador
unsigned long ultimoTempo = 0;
unsigned long ultimoTempoTemperatura = 0;
unsigned long ultimoTempoAc = 0;

unsigned long tempoEsperaTemperatura = 1000;
unsigned long tempoEsperaAcelerometro = 500;


// Cria um objedo no sensor
Adafruit_MPU6050 mpu;
sensors_event_t a, g, temp;

//variaveis
float RMSx, RMSy, RMSz;
float temperatura;
```

```
float offsetZ =  9.80;


// Função do sensor MPU6050
void initMPU(){
//Se não encontrar o sensor imprimi falha de sensor
  if (!mpu.begin()) {
    Serial.println("Folha do sensor MPU6050 ");
    while (1) {
      delay(10);
    }
  }
// Sensor encontrado
  Serial.println("MPU6050 Encontrado!");
}

// Função de acesso a memória flash
void initSPIFFS() {
  if (!SPIFFS.begin()) {
    Serial.println("Ocorreu um erro ao montar SPIFFS");
  }
  Serial.println("SPIFFS montado com sucesso");
}

// Função de inicialização do Wi-Fi
void initWiFi() {
  WiFi.mode(WIFI_STA);
//Inicia a conecção com a rede local
  WiFi.begin(NomedaRede, Senha);
  Serial.println("");
  Serial.print("Conectando ao Wi-Fi...");
// Caso não encontre entra no laço e arguade 1 segundo depois retorna.
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(1000);
  }
//Encontrado imprime o IP na serial da IDE.
  Serial.println("");
  Serial.println(WiFi.localIP());
}

// Função o cáculo de RMS
  String getAccReadings() {
  mpu.getEvent(&a,&g,&temp);
// Leitura dos eixos em RMS
  RMSx = a.acceleration.x*0.707;
  RMSy = a.acceleration.y*0.707;
```

```
// Offset no eixo Z desconsiderando a força da gravidade.
  RMSz = ((a.acceleration.z - offsetZ)*0.707) ;

  readings["RMSx"] = String(RMSx);
  readings["RMSy"] = String(RMSy);
  readings["RMSz"] = String(RMSz);

  String accString = JSON.stringify (readings);
  return accString;
}

String getTemperature(){
  mpu.getEvent(&a,&g,&temp);
  temperatura = temp.temperature;
  return String(temperatura);
}

void setup() {
// Inicia a serial da IDE
  Serial.begin(115200);
// Inicia o Wi-fi
  initWiFi();
// Innicia a comunicação com a memoria flash.
  initSPIFFS();
// Inisica o sensor MPU 6050
  initMPU();

 //Manipular servidor web
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/index.html", "text/html");
  });
  server.serveStatic("/", SPIFFS, "/");
 //Manipular servidor web Eventos
  events.onConnect([](AsyncEventSourceClient *client){
    if(client->lastId()){
      Serial.printf("Client reconnected! Last message ID that it got is: %u\n", client->lastId());
    }
    // send event with message "hello!", id current millis
    // and set reconnect delay to 1 second
    client->send("Olá!", NULL, millis(), 10000);
  });
  server.addHandler(&events);

  server.begin();
}

void loop() {
```

```
if ((millis() - ultimoTempoAc) > tempoEsperaAcelerometro) {
    // Envie eventos para o servidor web com as leituras do sensor
    events.send(getAccReadings().c_str(),"accelerometer_readings",millis());
    ultimoTempoAc = millis();
    Serial.println(getAccReadings());
  }
  if ((millis() - ultimoTempoTemperatura) > tempoEsperaTemperatura) {
    // Envie eventos para o servidor web com as leituras do sensor
    events.send(getTemperature().c_str(),"temperature_reading",millis());
    ultimoTempoTemperatura = millis();
  }
}
```

ANEXO II – HTML *interface* da web.

```
/*
  Fonte: Adaptado de Rui Santos, 2021.
  Detalhes completos do projeto em https://RandomNerdTutorials.com/esp32-mpu-6050-web-server/
*/


<head>
  <title>ESP Web Server</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" href="data:,">
  <link rel="stylesheet" type="text/css" href="style.css">
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/css/all.css" integrity="sha384-
fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr" crossorigin="anonymous">
  <script src="https://cdnjs.cloudflare.com/ajax/libs/three.js/107/three.min.js"></script>
</head>
<body>
  <div class="topnav">
    <h1><i class="far fa-compass"></i> MPU6050 <i class="far fa-compass"></i></h1>
  </div>
  <div class="content">
    <div class="cards">
      <div class="card">
        <p class="card-title">ACELEROMETRO</p>
        <p><span class="reading">X: <span id="X"></span> ms<sup>2</sup></span></p>
        <p><span class="reading">Y: <span id="Y"></span> ms<sup>2</sup></span></p>
        <p><span class="reading">Z: <span id="Z"></span> ms<sup>2</sup></span></p>
      </div>
      <div class="card">
        <p class="card-title">TEMPERATURA</p>
        <p><span class="reading"><span id="temp"></span> &deg;C</span></p>
      </div>
    </div>
  </div>
<script src="script.js"></script>
</body>
</html>
```

Anexo III – CSS estilo da web.

```
/*
Fonte: Adaptado de  Rui Santos, 2021.
  Detalhes completos do projeto em https://RandomNerdTutorials.com/esp32-mpu-6050-web-server/
*/

html {
  font-family: Arial;
  display: inline-block;
  text-align: center;
}
p {
  font-size: 1.2rem;
}
body {
  margin: 0;
}
.topnav {
  overflow: hidden;
  background-color: #006400;
  color: #FFFFFF;
  font-size: 1rem;
}
.content {
  padding: 20px;
}
.card {
  background-color: white;
  box-shadow: 2px 2px 12px 1px rgba(140,140,140,.5);
}
.card-title {
  color:#006400;
  font-weight: bold;
}
.cards {
  max-width: 800px;
  margin: 0 auto;
  display: grid; grid-gap: 2rem;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
}
.reading {
  font-size: 1.2rem;
}
.cube-content{
  width: 100%;
  background-color: white;
```

```
  height: 300px; margin: auto;
  padding-top:2%;
 }
 #reset{
  border: none;
  color: #006400;
  background-color: #006400;
  padding: 10px;
  text-align: center;
  display: inline-block;
  font-size: 14px; width: 150px;
  border-radius: 4px;
 }
 #resetX, #resetY, #resetZ{
  border: none;
  color: #006400;
  background-color: #006400;
  padding-top: 10px;
  padding-bottom: 10px;
  text-align: center;
  display: inline-block;
  font-size: 14px;
  width: 20px;
  border-radius: 4px;
 }
```