

Universidade Federal de Minas Gerais
Escola de Engenharia
Curso de Graduação em Engenharia de Controle e Automação

**Dispositivo IoT para monitoramento de vibração e
temperatura em máquinas elétricas**

Felipe Augusto Vitoriano

Orientador: Prof. Ricardo de Oliveira Duarte, Dr. (DELT/UFMG)

Belo Horizonte, Novembro de 2018

Projeto Final de Curso

Dispositivo IoT para monitoramento de vibração e temperatura em máquinas elétricas

Monografia submetida à banca examinadora designada pelo Colegiado Didático do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Minas Gerais, como parte dos requisitos para aprovação na disciplina Projeto Final de Curso II.

Belo Horizonte, Novembro de 2018

Resumo

Este projeto consiste em um dispositivo no âmbito das tecnologias de IoT (Internet of Things), que objetiva monitorar máquinas elétricas visando predizer falhas nestes equipamentos e otimizar sua manutenção. O dispositivo é equipado com instrumentação capaz de coletar a temperatura da superfície externa do equipamento, bem como a vibração mecânica nos três eixos espaciais. Estas informações são enviadas sem fio para um servidor, armazenando-as em um banco de dados. O servidor também realiza o cálculo da transformada de Fourier para os sinais de vibração, registrando o espectro de frequência dos sinais coletados. As informações são então exibidas em uma interface web desenvolvida para este fim. Com isto, é possível acompanhar a temperatura da máquina e o espectro de frequência de seu estado de vibração - informações importantes no rastreio de falhas mecânicas. Este dispositivo consiste, então, em uma importante ferramenta IoT de monitoramento e predição de falhas de máquinas elétricas.

Palavras-chave: IoT, manutenção, máquinas elétricas

Agradecimentos

Agradeço ao Professor Ricardo de Oliveira Duarte (DELT/EE-UFMG), que me orientou e apoiou neste projeto. Agradeço também aos Professores Maria Lucia Machado Duarte e Lazaro Valentim Donadon (DEMEC/EE-UFMG), que se prontificaram a me auxiliar nos testes no laboratório de Vibrações do DEMEC/EE-UFMG. Deixo também meu agradecimento a todos os que direta ou indiretamente colaboraram para o desenvolvimento deste trabalho.

Sumário

Resumo	i
Agradecimentos	iii
Lista de Figuras	vii
Lista de Tabelas	ix
1 Introdução	1
1.1 Cenário	1
1.2 Manutenção Industrial	2
1.3 O projeto	2
1.4 Objetivos do Projeto	2
1.4.1 Objetivo Geral	2
1.4.2 Objetivos Específicos	3
1.5 Estrutura da Monografia	3
2 Revisão Bibliográfica	5
2.1 A importância da manutenção	5
2.1.1 Manutenção preditiva de máquina elétricas	6
2.1.2 Vibração em máquinas elétricas	6
2.2 Amostragem e processamento de sinais	7
2.2.1 Transformada de Fourier	7
2.2.2 Teorema de Nyquist	7
2.3 Internet of Things	8
2.4 Sistemas Embarcados	8
2.4.1 Raspberry Pi Zero	8
2.4.2 Instrumentação	9
2.4.3 Comunicação e Rede	11
2.5 Linguagens de programação e recursos utilizados	12
2.5.1 Python	12
2.5.2 Postgresql	13
3 Materiais e Métodos	15
3.1 Materiais	15
3.1.1 Eletrônica	15
3.1.2 Comunicação/Processamento	15

3.1.3	Materiais para testes	16
3.2	Considerações gerais	16
3.3	Visão geral da arquitetura do sistema	17
3.4	Instrumentação	17
3.4.1	Diagrama do circuito elétrico	17
3.4.2	Acelerômetro MPU6050	17
3.4.3	Sensor de temperatura Temopar Tipo K	18
3.5	Sistema Embarcado	19
3.5.1	Visão geral	19
3.5.2	Leitura do acelerômetro	19
3.5.3	Leitura da Temperatura	20
3.5.4	Transmissão de dados	20
3.5.5	Main - Programa Principal	20
3.5.6	Server hunter	20
3.5.7	Alimentação do circuito	21
3.6	Comunicação	21
3.7	Servidor	21
3.7.1	Visão geral	21
3.7.2	Receptor de dados	22
3.7.3	Cálculos FFT	22
3.7.4	Banco de dados	22
3.7.5	Servidor Web	23
3.7.6	IP giver - Receptor de Broadcast	23
3.8	Interface web	24
3.9	Resumo do Capítulo	24
4	Resultados	25
4.1	Montagem do Dispositivo	25
4.2	Parâmetros de operação	26
4.2.1	Período de amostragem	26
4.2.2	Intervalo de frequências da FFT	26
4.3	Validação em laboratório	27
4.3.1	Recursos utilizados	27
4.3.2	Resultados dos testes	28
4.4	Caso de aplicação	30
4.5	Interface web	33
4.6	Resumo do Capítulo	34
5	Conclusões	35
5.1	Propostas de Continuidade	35
5.2	Considerações Finais	35
Referências Bibliográficas		37

Lista de Figuras

2.1	Raspberry Pi Zero W [6]	9
2.2	Módulo Comercial - MPU6050	10
2.3	Módulo Comercial - Termopar K e MAX6675	11
3.1	Arquitetura do sistema	17
3.2	Diagrama elétrico	18
3.3	Fluxograma de operação da aplicação embarcada	21
3.4	Exemplo - Gráficos matplotlib	22
4.1	Dispositivo montado	25
4.2	Disposição dos componentes	26
4.3	Excitador eletromecânico	27
4.4	Testes com o excitador eletromecânico	28
4.5	Resultados para excitador a 30Hz	28
4.6	Resultados para excitador a 400Hz	29
4.7	Resultados para excitador a 500Hz	29
4.8	Resultados para excitador a 900Hz	29
4.9	Resultados para excitador a 1250Hz	30
4.10	Resultados para excitador a 1400Hz	30
4.11	Plataforma de teste com motor de indução	31
4.12	Resultados para motor de indução desligado	31
4.13	Resultados para motor de indução ligado SEM desbalanceamento forçado	32
4.14	Resultados para motor de indução ligado COM desbalanceamento forçado	32
4.15	Testes com motor de indução	33
4.16	Interface Web - Exemplo	33
4.17	Interface Web - Recursos	34

Lista de Tabelas

3.1	Ligaçāo entre MPU 6050 e Raspberry Pi	18
3.2	Ligaçāo entre MAX6675 e Raspberry Pi	18
3.3	Tabela 'measurements' - Colunas, datatypes e descriçāo	23

Capítulo 1

Introdução

Este capítulo apresenta a ideia central e a relevância deste presente projeto.

1.1 Cenário

O mundo globalizado tem passado por diferentes revoluções ideológicas, políticas, econômicas, sociais e tecnológicas; assim foi e assim sempre será. A indústria, nos seus mais variados setores, não está inerte nesta dinâmica. As ‘revoluções’ industriais, que inclusive são didaticamente assim tratadas, são provas palpáveis de que as indústrias estão em constante adaptação, conforme os recursos disponíveis e as exigências de produtividade.

Desde a denominada Primeira Revolução Industrial na Inglaterra, quando então passou-se a organizar a produção em fábricas e utilizar máquinas a vapor para viabilizá-la, os processos industriais vêm sofrendo uma gradativa reformulação. Ainda nesse fluxo, a consequente Segunda Revolução trouxe máquinas mais eficientes e elétricas, e modelos de produção em série. Posteriormente, a Terceira Revolução mostrou uma nova abordagem para otimizar a produção utilizando novas tecnologias, como a automação e robótica, e também novos modelos de produção, como o japonês ‘toyotismo’.

A Terceira Revolução Industrial já é realidade consolidada e está dando lugar a outra cronologicamente intuitiva: A Quarta Revolução Industrial, ou Indústria 4.0, como tem sido aclamada pela comunidade industrial. Esta já vem sendo implantada nas indústrias, trazendo um novo e bem mais inteligente modelo de produção, mesclando tecnologias emergentes como automação avançada, processamento de dados, big data, *artificial intelligence* (AI), *cloud computing* e *Internet of Things* (IoT). Desde o chão de fábrica até os escritórios executivos, têm-se ouvido falar a respeito desta necessária “Revolução Digital”.

Estes recursos, unidos, prometem elevar a forma de produção (e a de administrá-la) para um patamar mais inteligente, no qual as máquinas poderão auxiliar humanos na tomada de decisão, na forma de produzir e de se relacionar. Além disso, tais aplicações são assustadoramente flexíveis: podem ser utilizadas tanto no nível operacional e técnico, quanto administrativo, financeiro, comercial, humano e estratégico.

1.2 Manutenção Industrial

Em uma discussão mais específica, uma aplicação destas novas tecnologias se concentra na área de manutenção do maquinário de chão de fábrica. Há um crescente esforço no uso de tecnologias para fornecer informações baseadas nos dados coletados da planta industrial, e consequentemente utilizá-las para melhorar os processos de manutenção.

Pode-se vislumbrar soluções nesta área no que diz respeito à manutenção preditiva. Nesse sentido, busca-se desenvolver sistemas capazes de predizer falhas e notificar o operador de uma máquina ou processo antes mesmo que tais falhas ocorram. Os ganhos desta aplicação ultrapassam discussões técnicas. Fato é que, quando se tem um sistema para prever falhas, o operador é capaz de atuar preventivamente de forma a evitar que defeitos ocorram no maquinário. Isto reflete não só na constância dos recursos e tecnologias, mas também no fluxo da produção, nos ciclos planejados, nos prazos estipulados, na eficiência dos processos, na segurança dos operadores e, em última análise, no rendimento econômico da empresa. Estes são argumentos expressivos que validam e incentivam a utilização de tecnologias inteligentes na manutenção.

1.3 O projeto

Este projeto final de curso visa aplicar conhecimentos integrados do curso de Engenharia de Controle e Automação para fornecer uma solução prática e diferenciada para a área de manutenção preditiva de máquinas elétricas.

O acompanhamento da condição de máquinas elétricas no ambiente industrial é crucial para saber a saúde do equipamento e, muitas vezes, variáveis de estado da máquina podem indicar preventivamente a iminência de um problema de ordem elétrica e/ou mecânica. Dessa forma, o acompanhamento do estado da máquina é uma importante ferramenta para a manutenção preditiva, evitando paradas de equipamento e garantindo que a máquina esteja em sua condição operacional adequada.

Nesse sentido, este projeto consiste em desenvolver um dispositivo eletrônico IoT (Internet of Things) a ser fixado em uma máquina elétrica, capaz de se conectar via internet a um servidor responsável por processar os dados coletados. O sistema será então capaz de medir a vibração e temperatura da máquina monitorada e exibi-las em uma interface para que o usuário acompanhe o estado da mesma. Deste modo, é possível predizer em tempo real algum defeito na máquina e recomendar ao operador a sua ação preventiva contra algum potencial problema.

1.4 Objetivos do Projeto

1.4.1 Objetivo Geral

Desenvolver um dispositivo IoT para coleta e processamento de dados de vibração e de temperatura de máquinas elétricas.

1.4.2 Objetivos Específicos

- Projetar e desenvolver o dispositivo IoT;
- Projetar e desenvolver a arquitetura de comunicação;
- Desenvolver os algoritmos para processamento e armazenamento dos dados;
- Desenvolver o sistema de interface com o usuário;
- Avaliar a eficiência e resultados práticos

1.5 Estrutura da Monografia

Este relatório está dividido em cinco capítulos. Este presente capítulo apresentou uma introdução ao projeto a ser descrito nesta monografia e o contexto ao qual ele se aplica.

O Capítulo 2 descreve os princípios e conceitos básicos utilizados neste projeto, necessários para um melhor entendimento do mesmo.

O Capítulo 3 aborda os métodos seguidos para o desenvolvimento do trabalho.

O Capítulo 4 apresenta os resultados do trabalho, bem como uma exposição dos testes realizados.

O último e quinto Capítulo apresenta uma conclusão do projeto, e as perspectivas futuras observadas.

Capítulo 2

Revisão Bibliográfica

Este capítulo apresenta os principais conceitos pesquisados e que são relevantes no âmbito deste projeto.

2.1 A importância da manutenção

Nos atuais e complexos sistemas industriais, torna-se árdua a identificação e predição de falhas por humanos em um tempo hábil. Falhas em sistemas industriais estão ligadas não só à eficiência do processo produtivo, mas também às ocorrências de acidentes e, por assim dizer, podem trazer prejuízos econômicos, ambientais e de segurança. Inclusive, setenta por cento dos acidentes industriais são causados por falhas humanas, expressivamente relacionadas à não observação do processo e da proeminência de falhas, segundo estatísticas de VENKATASUBRAMANIAN (2005) [21].

No ambiente industrial, vários processos são complexos, utilizam variados recursos e tecnologias e, assim, são passíveis de falhas. Para evitar e/ou corrigir tais falhas, as tarefas de manutenção desempenham um papel essencial para o funcionamento fluído e contínuo de um processo industrial. Nesse sentido, segundo FOGLIATTO e RIBEIRO (2009, p.4) [17], manutenções são realizadas com o objetivo de prevenir falhas ou de restaurar o sistema a seu estado operante, no caso de ocorrência de uma falha. O objetivo da manutenção é, portanto, manter e melhorar a confiabilidade e regularidade de operação do sistema produtivo.

Em outros aspectos, a manutenção envolve não somente métodos padronizados para execução de tarefas, quais sejam preditivas, preventivas ou corretivas. A manutenção envolve também fatores culturais e de organização que influenciam na maneira como os processos são organizados e implementados. Nesta outra perspectiva, KARDEC e NASCIF (2009) [18] abordam alguns paradigmas da manutenção com relação ao fator temporal e cultural:

- O paradigma do passado: o homem da manutenção sente-se bem quando executa um bom reparo;
- O paradigma do presente: o homem da manutenção sente-se bem quando também evita a necessidade do trabalho, a falha;
- O paradigma do futuro: o homem da manutenção sente-se bem quando ele consegue evitar todas as falhas não planejadas.

Segundo esta análise de 2009, de KARDEC e NASCIF (2009) [18], as empresas brasileiras se debruçavam majoritariamente sobre o paradigma do passado. Porém, num cenário ideal, falhas não deveriam acontecer e, portanto, o paradigma do futuro é a ideologia que deveria ser praticada na indústria inteligente. Dessa forma, falhas, repentinas ou não, devem ser devidamente identificadas e monitoradas para que seu efeito seja o menos oneroso possível.

Nesse sentido, dentre os vários tipos de manutenção, a manutenção preditiva assume um papel relevante. Este tipo de manutenção realiza acompanhamento de variáveis e parâmetros de desempenho de máquinas e equipamentos, visando definir o instante correto da intervenção, com o máximo de aproveitamento do ativo (OTANI MACHADO, 2008) [20].

Nessa perspectiva, a manutenção é uma prática já difundida industrialmente, e o seu tipo preditiva consiste em avaliar constantemente o processo e identificar possibilidade de falhas, a fim de se poder agir previamente e evitar as consequências destas intempéries.

Assim, atualmente já está difundido um movimento no sentido valorizar o 'paradigma do futuro', segundo o qual deve-se evitar as falhas, por meio das tarefas e recursos de manutenção preditiva. Indústrias agora adotam métodos padronizados para acompanhar parâmetros críticos de equipamentos e, em caso de sinalização de potenciais problemas, disparar Ordens de Manutenção para que uma equipe técnica possa avaliar preventivamente o equipamento observado, e atuar caso necessário.

2.1.1 Manutenção preditiva de máquina elétricas

Máquinas elétricas são peça central dos maquinários na maioria dos processos industriais. Segundo FITZGERALD (2014) [16], temos dois tipos principais de máquinas elétricas, as de corrente contínua (CC) e as de corrente alternada (CA), sendo que estas últimas ainda se subdividem em duas categorias comuns: síncronas e de indução. Seus variados tipos estão amplamente presentes na indústria (principalmente as máquinas CA de indução), e são responsáveis pelas principais tarefas e movimentos dos processos industriais.

O monitoramento e identificação de falhas nestes equipamentos constitui ferramenta importante nas tarefas de manutenção preditiva, dado que se tratam de peça fundamental no processo.

Como todas as máquinas rotativas, as máquinas elétricas estão expostas a várias adversidades, como estresses ambientais, térmicos e danos mecânicos, que demandam atenção máxima (LAMBERT et al., 2003) [19].

Falhas comuns em motores de indução, os mais utilizados na indústria, são geralmente relacionadas a fenômenos elétricos ou mecânicos. Dentre os aspectos elétricos que desencadeiam tais falhas, os mais comuns são: desbalanceamento de fase, transientes de tensão, harmônicos e sobrecargas. Por outro lado, dentre os aspectos mecânicos pode-se citar: desalinhamento entre eixo e carga, desbalanceamento do eixo, folgas, rolamentos danificados, e superfície de apoio deficiente.

2.1.2 Vibração em máquinas elétricas

Segundo BONALDI et al. (2012) [15], existem várias técnicas preditivas aplicadas a motores de indução com o objetivo de reduzir o número de paradas inesperadas provocadas pelas falhas citadas, cada uma conforme sua natureza. As técnicas mais comuns são: análise de

vibrações, análise acústica, análise das oscilações de velocidade, descargas parciais, análise do circuito elétrico, acompanhamento de temperatura.

Dentre estas, a análise de vibração é uma ferramenta robusta que pode auxiliar na detecção e predição das falhas mais comuns em motores elétricos. Segundo ÁGOSTON (2015) [22], cada tipo de falha, seja ela de ordem elétrica ou mecânica, geralmente produz uma vibração em uma frequência específica. Por isso, pode-se utilizar instrumentos para medir a vibração mecânica da máquina e, sabendo-se a região normal de operação, é possível identificar alguma anomalia caso uma medição mostre uma condição de operação fora da região de normalidade. Tecnicamente, avalia-se o sinal de vibração da máquina em tempo real e, através do monitoramento do espectro de frequência deste sinal, é possível associar certas frequências a determinadas falhas.

2.2 Amostragem e processamento de sinais

2.2.1 Transformada de Fourier

A análise do estado de vibração das máquinas elétricas pode ser feita através da inspeção do espectro de frequência do sinal de vibração medido, calculado por meio da transformada de Fourier. Esta transformada é uma função desenvolvida por Jean Fourier (1768-1830) em 1822 que decompõe uma função temporal absolutamente integrável em uma combinação linear de funções senoidais. Matematicamente, podemos simplificadamente representar a transformada de Fourier na seguinte expressão:

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(w) e^{iwx} dx$$

Nesta expressão, definimos como a transformada de Fourier de f como sendo a operação que associa a cada função absolutamente integrável $f : \mathbb{R} \rightarrow \mathbb{R}$ (no domínio do tempo) a função $\hat{f} : \mathbb{R} \rightarrow \mathbb{C}$.

Para implementação em computadores, a Transformada Rápida de Fourier (conhecida como FFT, do inglês *Fast Fourier Transform*) é um algoritmo eficiente e amplamente utilizado para se calcular a Transformada discreta de Fourier (DFT) e a sua inversa. Este será o algoritmo utilizado neste projeto para os cálculos aqui discutidos.

A partir da decomposição dos sinais de vibração (no domínio do tempo) no domínio da frequência, por meio da FFT, é possível analisar os pesos de cada frequência, possibilitando rastrear quais são as frequências dos sinais mais evidentes que constituem a função temporal observada. Assim, analisando o espectro de frequência do sinal de vibração de uma máquina elétrica, pode-se observar quais as frequências estão evidentes no regime normal de funcionamento, e quais se tornam evidentes em regimes de falhas e, assim, saber quando há uma falha iminente.

2.2.2 Teorema de Nyquist

Para se obter os sinais de aceleração (associados à vibração da máquina), é necessário amostrar este sinal por meio de sensores. A amostragem de sinais consiste em coletar valores deste sinal em instantes periódicos. Sinais no tempo contínuo (como a aceleração), quando

lidos por um sistema digital, são amostrados repetidamente para que se tenha uma versão digitalizada deste sinal.

Para que a versão digitalizada do sinal seja representativa e coerente com o sinal original (contínuo), certas condições precisam ser observadas. Nesse sentido, o *teorema da amostragem de Nyquist* diz que "*a taxa de amostragem de um sinal dever ser pelo menos duas vezes maior do que a frequência que se deseja registrar*". Pelo teorema, por exemplo, um sinal de 50Hz deve ser amostrado a uma taxa de, no mínimo, 100Hz. De outro lado, caso um sinal seja amostrado a uma taxa de 500Hz, a maior frequência identificável no sinal amostrado será de 250Hz.

2.3 Internet of Things

O termo IoT, do inglês *Internet of Things*, ou 'Internet das Coisas', vêm sendo amplamente empregado na atualidade, e corresponde a uma abordagem na qual dispositivos eletrônicos (desde simples sensores e eletrodomésticos até *smartphones*) são conectados entre si para fornecerem uma aplicação prática. Esta conexão pode usar a rede Internet, de forma que barreiras geográficas não sejam limitantes, e que vários benefícios dos recursos *online* possam ser agregados nestas soluções.

Este termo já ganhou uma nova versão, quando se trata de aplicações industriais e indústria 4.0: IIoT - *Industrial Internet of Things*. Nessa perspectiva, a ideia é utilizar sensores e dispositivos inteligentes, integrados aos processos industriais, de forma a fornecer informações de maneira online e inteligente, trazendo maior eficiência operacional, melhoria nos processos de automação, racionalização e manutenção [9].

2.4 Sistemas Embarcados

Sistemas embarcados, ou sistemas embutidos, são sistemas microcontrolados cuja parte central é um microprocessador projetado para executar uma função específica e dedicada, e geralmente se comunicar com periféricos (sensores, displays, botões, etc). Sistemas embarcados estão presentes em uma variedade de aplicações, como eletrodomésticos, brinquedos, carros, aviões e outros. Estes sistemas geralmente constituem os dispositivos IoT e utilizam diversos modelos de circuitos integrados e periféricos. A seguir, são apresentados alguns casos.

2.4.1 Raspberry Pi Zero

As placas da família Raspberry Pi [5] são sistemas eletrônicos dotados de processador e periféricos em um único módulo, sendo capazes inclusive de executar um sistema operacional comum. Tais placas também possuem GPIOs (*General Purpose I/O*, ou entradas e saídas de uso geral), que permitem conexão com circuitos e dispositivos externos.

A versão Raspberry Pi Zero W [6] foi lançada em fevereiro de 2017, e é uma versão de tamanho reduzido, sendo perfeitamente aplicável a projetos de IoT. A figura 2.1 mostra a versão Pi Zero W.

Dentre as principais características do módulo Raspberry Pi Zero W, podemos citar:

- Conexão 802.11 b/g/n wireless LAN



Figura 2.1: Raspberry Pi Zero W [6].

- Conexão Bluetooth 4.1 e Bluetooth Low Energy (BLE)
- CPU de 1GHz, single-core
- 512MB de memória RAM
- Portas Mini HDMI e USB On-The-Go
- 40 pinos GPIO

As placas Raspberry são capazes de executar um sistema operacional que esteja instalado em um cartão SD inserido no seu slot. O sistema operacional (SO) *Raspbian*[3] é um dos SOs mais adequados para utilização nestes casos. O Raspbian é uma variante enxuta do Debian, otimizada para compatibilidade com o hardware do Raspberry Pi.

2.4.2 Instrumentação

A nível de instrumentação, a proposta é conectar um sensor de vibração e um de temperatura à máquina monitorada.

Sensor de Vibração

Dentre os mais simples sensores de vibração, existem os do tipo piezo, constituídos por um transdutor piezoelétrico que responde às alterações de forças produzindo uma voltagem mensurável. Porém, tais sensores realizam a medição de vibração somente no sentido longitudinal ao seu elemento transdutor.

Para a medição em três dimensões (o que é desejável quando pretende-se medir sinais de vibração provenientes de diversas fontes e, portanto, diversos sentidos), é mais adequado o uso de acelerômetros de três eixos, como o MPU6050 da Invensense [10]. Este é classificado como um MEMS (*Micro Electro-Mechanical Sensor*), equipado com um acelerômetro e um giroscópio tridimensionais. Neste projeto, apenas as funções do acelerômetro serão utilizadas.

Para mensurar a aceleração do meio (que, neste presente projeto, será correspondida a um sinal de vibração), este sensor possui uma massa sísmica que fica suspensa em um meio dielétrico. A capacidade entre placas de um material condutor que envolve tal massa varia

conforme a posição da mesma entre estas placas. Dessa forma, quando a massa for submetida a uma aceleração, ela deslocará de sua posição de equilíbrio e, portanto, a capacitância entre as placas condutoras sofrerá uma alteração. Esta capacitância pode ser medida e correspondida a um sinal de aceleração que, por sua vez, pode ser associado a um sinal de vibração.

O CI MPU6050 possui internamente toda a estrutura para leitura, condicionamento e filtragem dos sinais; fornece o sinal de saída de forma digital e comunica-se com micro-controladores por meio do protocolo I₂C (*Inter-Integrated Circuit*, um barramento serial multimestre desenvolvido pela Philips).

Nesta aplicação, o sensor será acoplado à armadura do motor, de forma a perceber os sinais de vibração aos quais o motor está submetido. A figura a seguir (2.2) mostra um módulo comercial com MPU6050:

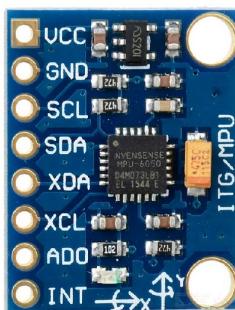


Figura 2.2: Módulo Comercial - MPU6050

Sensor de Temperatura

Valores de temperaturas comuns na superfície da armadura do motor (onde a temperatura será medida), estão em torno de 70°C (considerando apenas o motor elétrico como fonte de calor, e ignorando influências externas).

Imaginando o pior caso, as classes de isolamento dos motores mostram limites de temperatura de até 180°C para classe H, este valor correspondendo à região das bobinas. Nesta situação extrema, a superfície externa da armadura assume temperaturas menores (máximo de 125°C nesta classe), pois não está conectada diretamente à fonte de calor, e também está em contato com as aletas de ventilação e com o ambiente externo.

Neste projeto, o sensor responsável por monitorar a temperatura do motor será colocado em contato com a superfície externa da armadura. Considerando estas condições, o sensor de temperatura do tipo termopar são adequados. Sensores termopar são baratos, simples e robustos. Tais sensores são constituídos por dois metais distintos unidos em uma das extremidades. Quando existe uma diferença de temperatura entre a extremidade unida e as extremidades livres, surge uma diferença de potencial (DDP). Esta DDP pode ser medida por um voltímetro sensível e então associada a um valor de temperatura. Existem diferentes tipos de termopares, com diferentes relações tensão/temperatura.

Um tipo comum de termopar é o tipo K (*Cromel / Alumel*), de uso genérico: são capazes de medir temperaturas entre -200°C e 1200°C, com sensibilidade aproximada de 41V/°C. São, portanto, compatíveis com a aplicação aqui pretendida.

Existem módulos comerciais de termopares tipo K com um circuito condicionador, como o mostrado na figura 2.3.



Figura 2.3: Módulo Comercial - Termopar K e MAX6675

Este conjunto utiliza o CI MAX6675 [12], que executa compensação de junção fria e digitaliza o sinal de temperatura lido (em 12-bits). É capaz de medir temperaturas de -20°C a 800°C, com precisão de 0.25°C. Este módulo possui uma interface SPI (*Serial Peripheral Interface*), um protocolo serial para comunicação com microcontroladores.

2.4.3 Comunicação e Rede

Comunicação entre aplicações em sistemas distribuídos

Para a comunicação com um servidor (operando em um computador local ou na nuvem, por exemplo) incubido de receber, processar e armazenar dos dados coletados pelo Dispositivo, é aplicável a implementação de um protocolo de comunicação que garanta a transferência dos dados entre o Dispositivo e o servidor.

Estratégias comuns de comunicação desta ordem geralmente são apoiadas nas camadas de rede TCP (*Transmission Control Protocol*) e IP (*Internet Protocol*), baseadas no modelo OSI (*Open Systems Interconnection model*).

Um dos protocolos sob estas camadas são os *network sockets* TCP/IP. Sockets são os pontos finais em um canal de comunicação entre dois processos através de uma rede, podendo ser utilizados para se transmitir dados entre duas aplicações em sistemas distribuídos.

Existem dois tipos de sockets principais: sockets TCP e UDP, e cada um apoia na camada que traz no nome. Os sockets TCP são orientados a conexão entre os dois pontos comunicantes, estabelecendo um canal exclusivo de comunicação. Sockets TCP garantem a entrega dos pacotes em ordem, e por isso são mais confiáveis. De outro lado, os sockets UDP não possuem garantia de entrega dos pacotes, e desconsidera a ordem de envio dos mesmos. São mais simples e mais rápidos, porém, não-confiáveis.

Várias linguagens possuem funções para acessar os recursos dos sockets e, neste projeto, foram utilizadas as funções de interface de rede do pacote '*socket*' do Python [4], conforme seção 2.5.1.

Aplicações Web

Aplicações web são sistemas desenvolvidos para utilização através de um navegador de internet ou de aplicativos móveis. Nestas aplicações, temos dois agentes principais: o *front-end* e o *back-end*. Simplificadamente, *front-end* consiste em um conjunto de códigos HTML, CSS e JavaScript que são executados no navegador, e geram a interface com o usuário. Do outro lado, o *back-end*, que consiste em um servidor web, é uma aplicação central responsável por receber ou enviar dados para os clientes (navegadores).

A comunicação entre navegadores (clientes) e o servidores web geralmente é feita por meio de um protocolo de comunicação denominado REST (*REpresentational State Transfer*), baseado no protocolo de hipermídia HTTP (*Hypertext Transfer Protocol*). Este protocolo reúne métodos padronizados para a troca de dados entre o servidor web e os navegadores/aplicativos, e seu uso é bastante difundido nos serviços web de hoje.

2.5 Linguagens de programação e recursos utilizados

2.5.1 Python

Python [2] é uma linguagem de programação de alto nível orientada a objetos e lançada em 1991. Diferentemente de linguagens compiladas (que são convertidas em código fonte, que é então executado diretamente pelo sistema operacional ou processador), Python é uma linguagem interpretada, em que o código fonte é executado por uma aplicação chamada 'interpretador' que, em seguida, é executado pelo sistema operacional ou processador.

Atualmente, a comunidade Python é crescente, trazendo vários recursos e bibliotecas com aplicações matemáticas e científicas bastante robustas. A seguir, temos a descrição de alguns pacotes Python relevantes.

Pacote Sockets

O pacote *sockets*[4] da linguagem Python implementa funções de acesso à interface de sockets do Sistema Operacional, e foi utilizado neste projeto para estabelecer a comunicação entre o Dispositivo IoT e o servidor.

Pacote NumPy

O pacote *NumPy*[8] para a linguagem Python é um conjunto de funções e recursos para cálculos científicos. Este pacote reúne funções para trabalhar com arrays e matrizes multidimensionais, com uma densa coleção de funções matemáticas implementadas.

Destre estas funções, destaca-se a *numpy.fft*[7], que será utilizada neste trabalho. Esta função realiza o cálculo da transformada de Fourier, a ser utilizada para processar os sinais de aceleração, conforme seção 2.2.1

Pacote Matplotlib

O pacote *matplotlib*[11] é uma biblioteca utilizada para a geração de gráficos 2D interativos com várias ferramentas de visualização. Com o *matplotlib*, é possível gerar histogramas, gráficos de barras, gráficos de erros, diagramas de dispersão e outros.

Pacote Flask

Flask[13] é um framework web escrito em Python, utilizado para o desenvolvimento de servidores web, provendo modelos simples para tais fins. É classificado como um micro-framework, pois possui um núcleo simples (não possui recursos como camada de abstração do banco de dados, validação de formulários, etc). Mesmo sendo simples, é um framework extensível e pode ser utilizado em conjunto com outras bibliotecas, permitindo assim o desenvolvimento de aplicações web completas.

2.5.2 Postgresql

Bancos de dados relacionais são um conjunto organizado de dados relacionados entre si. Sistemas de gerenciamento de banco de dados objeto relacional (SGBDOR) são aplicações que gerenciam o acesso e controle de um banco de dados. O PostgreSQL é um SGBDOR robusto de código aberto. O *PSQL*, como também é chamado, utiliza e extende a linguagem SQL (*Structured Query Language*), combinada com uma série de recursos que armazenam e escalam cargas de dados com segurança.

Para a comunicação entre do banco de dados e as aplicações, existe, no caso de aplicações Python, o pacote *psycopg*[14], que é o adaptador PostgreSQL mais popular nesse sentido. O *psycopg* permite acesso a muitos dos recursos oferecidos pelo PostgreSQL.

Capítulo 3

Materiais e Métodos

A primeira proposta deste trabalho é, portanto, desenvolver o dispositivo dotado de instrumentação capaz de medir a vibração e temperatura de uma máquina elétrica; e, em segunda instância, desenvolver um software-servidor que possa receber, processar e arquivar os dados coletados, permitindo a análise dos sinais por meio de gráficos de FFT e de temperatura.

3.1 Materiais

3.1.1 Eletrônica

- 1 placa Raspberry Pi Zero W
- 1 módulo acelerômetro MPU6050
- 1 Sensor Temopar Tipo K com conversor MAX6675
- 1 módulo de bateria recarregável - powerbank 3000mAh 5V
- 1 LED RGB 3mm
- 3 resistores 1/4W 10 KOhm
- 1 chave on-off tipo gangorra
- Fios, conectores, caixa plástica para montagem, ímas para fixação na máquina

3.1.2 Comunicação/Processamento

- Roteador WiFi
- Estrutura de rede
- Microcomputador para instalação do servidor de dados

3.1.3 Materiais para testes

- Motor elétrico em operação
- Excitador Eletromecânico
- Ferramentas em geral

3.2 Considerações gerais

Esta seção apresenta os principais passos seguidos na execução deste projeto, a saber:

1. Primeiramente foi feita uma revisão bibliográfica para identificar trabalhos similares e identificar os recursos disponíveis;
2. Posteriormente, foram selecionados os recursos a serem utilizados (microcontrolador, elementos de comunicação, periféricos, instrumentos, algoritmos) que mais se adequam aos requisitos do projeto;
3. Após isto, foram feitos testes preliminares (prototipagem) com os componentes selecionados, a fim de validar as hipóteses, identificar as limitações e o correto funcionamento das partes, sugerindo alterações, se aplicáveis. Estes testes incluem a validação dos sensores, testes de interferência, teste de alimentação do circuito, e outros;
4. Após prototipagem, foi feito o projeto do dispositivo para conexão definitiva dos componentes e confecção da caixa plástica para proteção dos mesmos;
5. Foi então feito o projeto do algoritmo do sistema embarcado a ser executado na Raspberry Pi, considerando a comunicação via hardware com os sensores, e o sistema de comunicação wireless com o servidor (protocolos, frame de dados, arquitetura);
6. No âmbito do servidor, foi feito o projeto e teste dos algoritmos para recebimento, processamento e arquivamento dos dados coletados pelo Dispositivo IoT;
7. Foi então realizada a validação dos algoritmos para processamento dos dados, utilizando códigos de geração de dados fictícios;
8. Após isto, foi feita a integração dos recursos desenvolvidos (eletrônica, rede e servidor);
9. Testes em laboratório;
10. Desenvolvimento da interface web;
11. Testes gerais em campo e validação da solução.

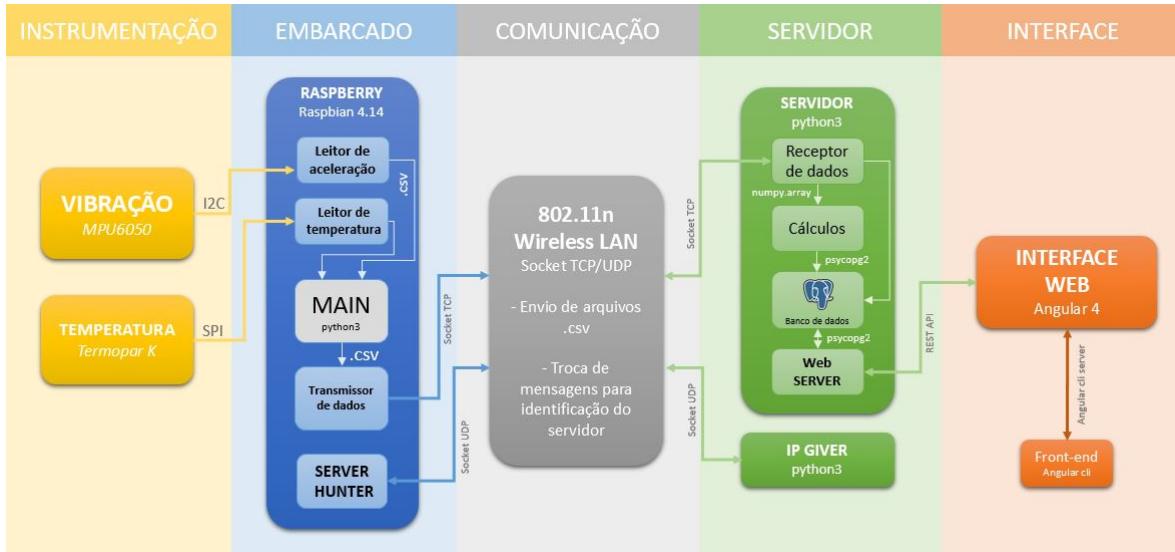


Figura 3.1: Arquitetura do sistema

3.3 Visão geral da arquitetura do sistema

A figura (3.1) mostra uma visão geral da arquitetura do sistema. Percebe-se que o sistema foi dividido em grupos funcionais conforme sua função principal, sendo:

1. Instrumentação
2. Sistema Embarcado
3. Comunicação
4. Servidor
5. Interface web

Os tópicos seguintes apresentam, em detalhes, cada uma das partes contidas na arquitetura apresentada, mostrando suas funções e recursos utilizados.

3.4 Instrumentação

3.4.1 Diagrama do circuito elétrico

O esquema de ligação elétrica entre o Raspberry Pi e os periféricos (Sensor de Temperatura, Acelerômetro e LED RGB - para sinalização de funcionamento) é mostrado na figura 3.2. A seguir, são apresentados detalhes de cada interface.

3.4.2 Acelerômetro MPU6050

Conforme explicitado no diagrama, a conexão do Acelerômetro MPU6050 com o Raspberry é feita via barramento I2C, nos pinos de GPIO mostrados. A correspondência de pinos é mostrada na tabela 3.1:

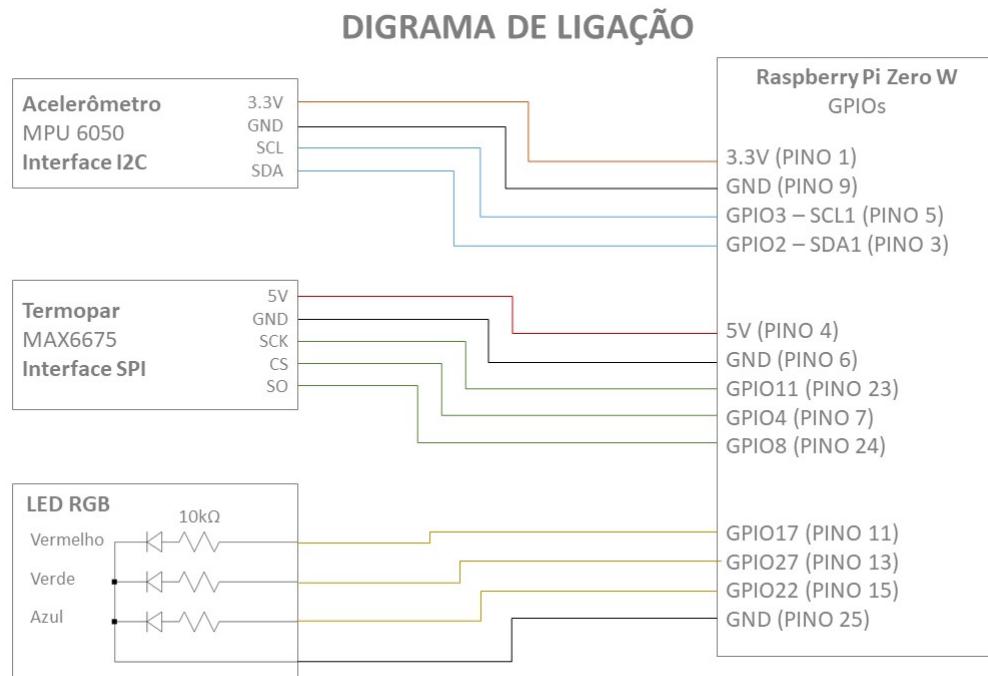


Figura 3.2: Diagrama elétrico

MPU6050	Raspberry Pi
3.3V	PINO 1 - 3.3V
GND	PINO 9 GROUND
SCL	PINO 5 - GPIO3
SDA	PINO 3 - GPIO2

Tabela 3.1: Ligação entre MPU 6050 e Raspberry Pi

3.4.3 Sensor de temperatura Termopar Tipo K

Conforme diagrama, a conexão do conversor digital MAX6675 com o Raspberry é feita via barramento serial SPI. A correspondência de pinos é mostrada na tabela 3.2:

MAX6675	Raspberry Pi
5V	PINO 4 - 5V
GND	PINO 6 GROUND
SCK	PINO 23 - GPIO11
CS	PINO 7 - GPIO4
SO	PINO 24 - GPIO8

Tabela 3.2: Ligação entre MAX6675 e Raspberry Pi

3.5 Sistema Embarcado

3.5.1 Visão geral

O componente central do Dispositivo IoT é a placa Raspberry Pi Zero W, que executa a leitura dos sensores e envia os dados para o servidor. Foi instalado na Raspberry o Sistema Operacional Raspbian, conforme discutido na seção 2.4.1. A aplicação que é executada na Raspberry foi desenvolvida em Python 3, e é executada ciclicamente, chamando as funções que leem os dados dos periféricos e enviam tais dados ao servidor. Os principais passos desta aplicação são:

1. Realizar a leitura do acelerômetro
2. Realizar a leitura do termopar
3. Integrar os dados lidos e enviar ao servidor, por meio da função de transmissão de dados
4. Aguardar 30 segundos
5. Retornar ao passo 1

A seguir, são apresentados detalhes de cada função.

3.5.2 Leitura do acelerômetro

O acesso aos dados do acelerômetro é feito via código em linguagem C, utilizando o driver *i2c-dev*, presente no Raspbian OS.

O código desenvolvido realiza 42000 leituras consecutivas (valor padrão, mas configurável) dos registradores do MPU6050 que armazenam o valor instantâneo da aceleração em cada um dos eixos X, Y e Z. Para efeitos de velocidade de execução, estes dados são salvos na memória RAM em três *arrays* do tipo *long*, uma para cada eixo. Além disso, o código também calcula o período médio de amostragem, que corresponde ao tempo entre duas amostragens completas.

O MPU6050 também possui um registrador que contém o valor da temperatura interna do Circuito Integrado, que pode ser convertida para graus Celsius. Este valor de temperatura também é lido pelo código C. O código também acessa a data e hora do SO, para registrar o momento da coleta dos dados.

Após a conclusão de todas as leituras, o código salva os dados em um arquivo no formato .csv (*Comma Separated Values*), contendo três colunas, uma para cada eixo de amostragem da aceleração (X,Y,Z). Os demais dados (número de amostras, período médio de amostragem, temperatura interna) são concatenados no nome do referido arquivo, utilizando o padrão seguinte:

Nome do arquivo:

Data_AAAA-MM-DD-hh-mm-ss_Namostras_Tmedio_TempInterna.csv (3.1)

Sendo: **AAAA** = Ano da data de coleta; **MM** = Mês da data de coleta; **DD** = Dia da data de coleta; **hh** = Hora da data de coleta; **mm** = Minuto da data de coleta; **ss** = Segundo da data de coleta; **Namostras** = Número de amostras de aceleração coletadas; **Tmedio** = Período médio de amostragem; **TempInterna** = Temperatura Interna do MPU6050.

O arquivo é salvo em uma pasta temporária, e é enviado posteriormente para o servidor pela função de transmissão de dados.

3.5.3 Leitura da Temperatura

Os dados de leitura do temopar são obtidos em uma resolução de 12-bits, por meio de dois registradores de 8-bits do MAX6675. Foi desenvolvido um código em Python para acessar estes dois registradores, obter os 12-bits correspondentes à leitura do termopar, e calcular o valor instantâneo da temperatura em graus Celsius correspondente. Este valor de temperatura é enviado ao servidor juntamente com o arquivo dos dados de aceleração coletados.

3.5.4 Transmissão de dados

O transmissor de dados é uma função em Python3 que implementa um socket TCP (garantindo a entrega dos pacotes). Esta função se conecta ao servidor, e envia o arquivo .csv com os dados coletados.

3.5.5 Main - Programa Principal

O programa principal é uma aplicação desenvolvida também em Python 3 que gerencia a execução de todas as funções. Esta aplicação roda ciclicamente a cada trinta segundos e, ao ser disparada, chama as funções para leitura do acelerômetro e do termopar e, em seguida, integra os dados lidos e chama função de envio de dados ao servidor. O fluxograma da figura 3.3 representa este processo.

Conforme diagrama elétrico da figura 3.2, esta aplicação também controla um LED RGB ligado às portas GPIO do Raspberry Pi, que é utilizado para informar o status da execução do programa, conforme o seguinte padrão: Cor azul - Lendo sensores; Cores azul e verde - Enviando dados ao servidor; Cor vermelha - Erro ao acessar o servidor.

3.5.6 Server hunter

Além da aplicação principal, existe uma aplicação em Python3 que é executada para identificar o IP do servidor, se ativo. Isto é vantajoso pois o Dispositivo IoT consegue encontrar o servidor na rede, sem necessidade de configuração prévia. Esta aplicação é executada no momento da inicialização do dispositivo, ou em casos em que a conexão com o servidor é perdida. Esta aplicação implementa um socket UDP (sem garantia de entrega dos pacotes),



Figura 3.3: Fluxograma de operação da aplicação embarcada

enviando uma mensagem *broadcast* na rede, em uma porta específica (número 60000). O servidor, uma vez ativo, possui uma aplicação que está preparada para receber requisições nesta porta e, uma vez recebida a mensagem de algum cliente, o servidor responde com uma mensagem de *acknowledgement*, contendo o seu IP atual (ver seção 3.7.6).

3.5.7 Alimentação do circuito

A alimentação da placa Raspberry (5V) e seus periféricos é feita por um circuito com bateria recarregável de 3000mAh, retirado de um *powerbank* (normalmente utilizado para carga de celulares). Dessa forma, a partir da bateria completamente carregada, o Dispositivo é capaz de operar ininterruptamente por aproximadamente doze horas.

3.6 Comunicação

A comunicação, como discutida, é feita por meio de sockets TCP (para envio dos dados) e UDP (para identificação do servidor). A rede é implementada via interface sem fio, já que o Raspberry Pi possui adaptador de rede wireless. O servidor pode ser executado em qualquer computador que esteja conectado à mesma rede local na qual o Dispositivo IoT se encontra conectado.

3.7 Servidor

3.7.1 Visão geral

O servidor é responsável por receber os dados do dispositivo IoT, realizar os cálculos de FFT e armazenar os resultados no banco de dados. Também é papel do servidor executar uma aplicação para responder às requisições de browsers que estão executando a interface

web, que será discutida posteriormente. Finalmente, o servidor também executa a aplicação *IP Giver*, que é responsável por responder às mensagens broadcast de clientes que estão à procura do IP do servidor, conforme já mencionado.

O servidor é uma aplicação *.py* que réune funções para cada tarefa principal, e as executa sequencialmente conforme a ordem descrita a seguir.

3.7.2 Receptor de dados

O receptor de dados é uma função em Python 3 que implementa um servidor de conexões via sockets TCP, e recebe os arquivos *.csv* que o Dispositivo IoT envia. Este bloco de função carrega os dados do conteúdo e do nome do arquivo em variáveis do programa. As temperaturas lidas (interna e do sensor termopar), o período de amostragem e a data de coleta de dados são atribuídos a variáveis globais do código. Os dados de aceleração XYZ são carregados em *arrays* do pacote *NumPy*[8], que será discutido no próximo item.

3.7.3 Cálculos FFT

Os dados de aceleração coletados necessitam ser processados para se obter a transformada de Fourier correspondente. Isto é feito por meio da função *numpy.fft*[7], que implementa os cálculos da Transformada Discreta de Fourier. Tanto os dados de aceleração (inputs para a função *numpy.fft*) quanto os de fft (inputs para a função *numpy.fft*) são salvos em *arrays* do pacote *NumPy*[8].

Após os cálculos da FFT, temos quatro vetores de resultado: três vetores com os valores absolutos da FFT calculada (um para cada eixo XYZ, em m/s²); outro vetor com as frequências correspondentes a cada ponto da FFT (em *Hertz* - Hz).

Para visualização do espectro de frequência, são gerados gráficos a partir destes dados, por meio do pacote *matplotlib*. Um exemplo dos gráficos gerados é mostrado na figura 3.4.

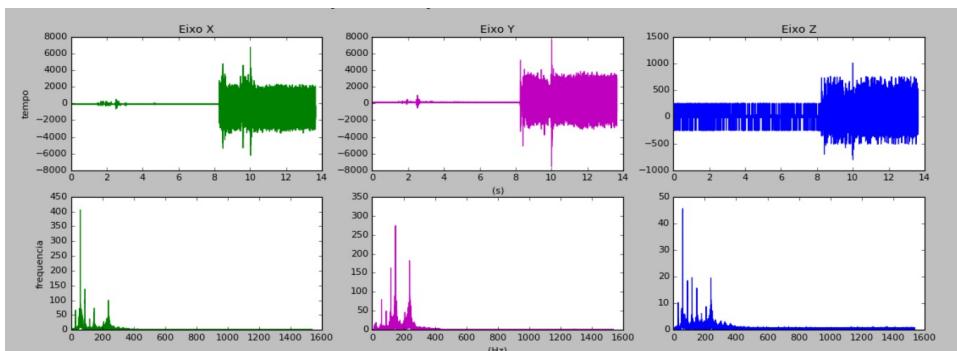


Figura 3.4: Exemplo - Gráficos matplotlib

Estes dados são então salvos no banco de dados por meio das funções do *psycopg*[14], como será discutido na próxima seção.

3.7.4 Banco de dados

O acesso ao banco de dados PostgreSQL é realizado por meio dos recursos do pacote *psycopg*[14], através de instruções SQL. Os dados são salvos em uma tabela chamada *measure-*

ments, com as colunas especificadas na tabela 3.3 a seguir.

Tabela 3.3: Tabela 'measurements' - Colunas, datatypes e descrição

Nome	Tipo de dado	Descrição
device	varchar(30)	String de identificação única do dispositivo
date_read	DateTime	Data/hora de coleta dos dados
date_saved	DateTime	Data/hora de armazenamento dos dados
x_fft	double precision[]	Vetor com os valores da FFT (eixo X)
y_fft	double precision[]	Vetor com os valores da FFT (eixo Y)
z_fft	double precision[]	Vetor com os valores da FFT (eixo Z)
min_freq	double precision	Mínima frequência do vetor de frequências da FFT
max_freq	double precision	Máxima frequência do vetor de frequências da FFT
internal_temperature	double precision	Valor da temperatura interna (acelerômetro)
external_temperature	double precision	Valor da temperatura externa (termopar)

Dessa forma, cada linha da tabela corresponde a uma coleta de dados do Dispositivo IoT.

3.7.5 Servidor Web

Foi desenvolvida uma interface web para visualização dos dados via navegador. Por isso, existem dois recursos que operam para permitir o funcionamento da interface web, a saber: front-end e back-end.

Os serviços de front-end são melhor detalhados na seção 3.8. O **Back-end (Servidor Web)** é uma aplicação Python que utiliza o framework *flask* [13], para implementar uma API (*Application Programming Interface*) REST.

O servidor web estabelece conexão com o banco PostgreSQL, e responde às requisições dos cliente que solicitam os dados salvos no banco. O servidor web opera na mesma máquina em que o servidor central é executado, conforme diagrama na figura 3.1.

3.7.6 IP giver - Receptor de Broadcast

Uma outra aplicação, denominada *IP Giver*, como mostrado no diagrama em 3.1, é uma aplicação que responde às mensagens broadcast enviadas pelo Dispositivo IoT quando o mesmo está à procura do servidor. A resposta a estas mensagens consiste no número atual do IP do servidor, informando o Dispositivo IoT o endereço para o qual este deve enviar os dados.

Esta aplicação é executada em paralelo com o servidor principal (em uma thread separada), e implementa um servidor de sockets UDP. Nesse caso, utilizou-se sockets UDP devido às mensagens broadcast, que não têm garantia de entrega. O servidor fica na escuta por uma mensagem específica na porta 60000 (Mensagem: "WhereIsTheServer") e, caso esta mensagem seja recebida, o servidor responde com uma mensagem contendo seu IP (Ex.: "IP:192.168.1.5").

3.8 Interface web

A interface web é um recurso desenvolvido para que o usuário possa acessar e visualizar os dados coletados por meio de um navegador de internet. Os códigos *front-end* (HTML, CSS, e JavaScript) para a interface web foram desenvolvidos dentro da plataforma *Angular 4*[1], que facilita a criação de aplicações web. Foi utilizado como servidor para os códigos de front-end o *Angular CLI*, uma interface Angular em 'linha de comando', que permite hospedar um website localmente.

3.9 Resumo do Capítulo

Este capítulo apresentou os detalhes relevantes de cada uma das principais partes do sistema desenvolvido e a interação entre as mesmas: instrumentação, sistema embarcado, comunicação, servidor, e interface web. O próximo capítulo apresenta os resultados obtidos na execução deste projeto através dos teste realizados.

Capítulo 4

Resultados

Este capítulo apresenta os resultados do sistema desenvolvido, por meio de testes e validações executados.

4.1 Montagem do Dispositivo

O Dispositivo (contendo os módulos dos sensores, a placa Raspberry, o LED RGB e o circuito de alimentação) foram montados em uma caixa plástica de dimensões 100x60x25mm. A imagem a seguir (4.1) mostra o Dispositivo.



Figura 4.1: Dispositivo montado

Percebe-se na figura 4.1 a sonda do sensor termopar (objeto metálico à direita), que pode ser colocado no local em que se deseja aferir a temperatura.

Foi colocado um íma de *neodimium*, encontrado em Discos Rígidos (HDs)抗igos, na parte de trás da caixa plástica, de forma que o Dispositivo se fixe automaticamente à su-

perfície metálica das máquinas elétricas monitoradas. Também foi colocado um íma similar na ponta de prova do termopar, para que o mesmo também se fixe à superfície metálica.

A figura 4.2 mostra cada um dos componentes montados no interior da caixa.

DISPOSIÇÃO DOS COMPONENTES

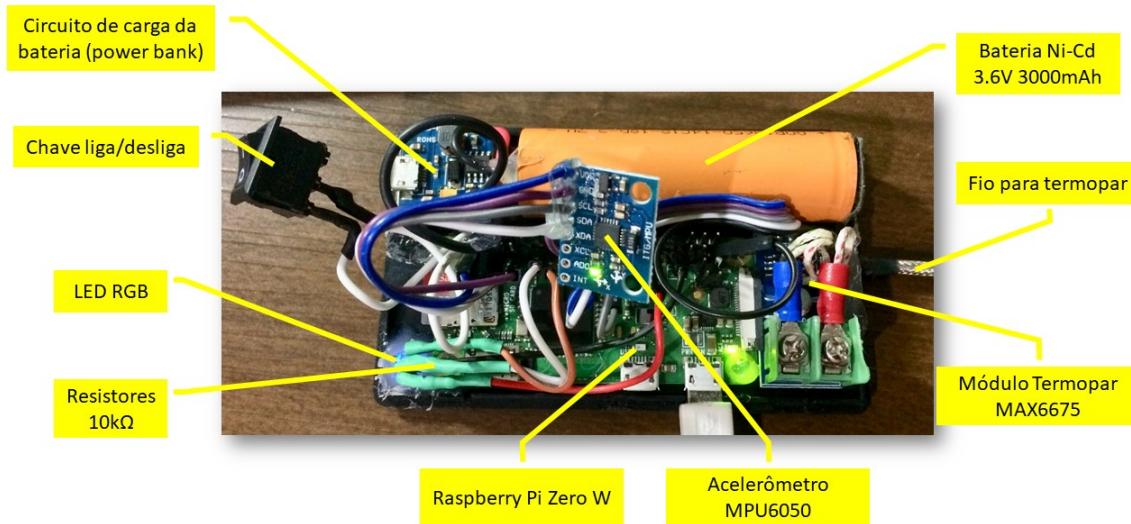


Figura 4.2: Disposição dos componentes

4.2 Parâmetros de operação

4.2.1 Período de amostragem

Um dos parâmetros importantes identificados foi o período de amostragem da aceleração. O tempo médio de amostragem dos três valores de aceleração (um para cada eixo X, Y e Z) foi de 330us.

Dessa forma, a frequência de amostragem da aceleração é de aproximadamente 3,3kHz. Assim, considerando o número de amostras configurado para 42000, temos que o intervalo de tempo total em cada coleta de dados de aceleração é de

$$t_{\text{coleta}} = \text{Periodo_amostragem} * \text{Num_de_amostras} = 330 * 10^{-6} * 4200 = 14s$$

Portanto, o tempo médio de coleta dos sinais de vibração é de aproximadamente **14 segundos**. Este tempo se mostrou suficiente para rastrear os sinais de vibração, como será mostrado nos testes.

4.2.2 Intervalo de frequências da FFT

Pelo Teorema de Nyquist discutido na seção 2.2.2, a frequência de amostragem de um sinal deve ser pelo menos duas vezes maior do que a maior frequência das componentes do referido sinal.

Assim, sabendo que a taxa de amostragem do Dispositivo IoT é de 3,3kHz, temos que a **frequência máxima identificável é de aproximadamente 1,6kHz**. Dessa forma, o eixo das abscissas nos gráficos da FFT vai de zero a 1600Hz.

4.3 Validação em laboratório

Com o objetivo de validar o adequado funcionamento do sistema, bem como a assertividade das medições de vibração, foram realizados testes em laboratório, utilizando inclusive um acelerômetro comercial calibrado para confrontar os dados.

4.3.1 Recursos utilizados

Os testes foram realizados no Laboratório de Vibrações do Departamento de Engenharia Mecânica da escola de Engenharia da UFMG, no campus Pampulha, em Belo Horizonte-MG.

Para realizar os testes, foi utilizado um excitador eletromecânico (*shaker*), como mostrado na figura 4.3. Este equipamento é capaz de gerar uma vibração mecânica de amplitude e frequência configuráveis via computador. Este excitador foi utilizado, portanto, para a geração de uma vibração específica, a ser lida pelo Dispositivo IoT, para que pudesse ser verificada a consistência da medição do mesmo.



Figura 4.3: Excitador eletromecânico

Primeiramente, o excitador foi calibrado conforme procedimentos padrão do laboratório, utilizando um acelerômetro comercial (pequeno cilindro sob o excitador, mostrado na figura 4.3) para, então, se proceder com os testes com o Dispositivo IoT.

4.3.2 Resultados dos testes

Para a realização dos testes, o acelerômetro do Dispositivo IoT foi colocado sob a superfície de vibração do excitador, conforme figura 4.4.

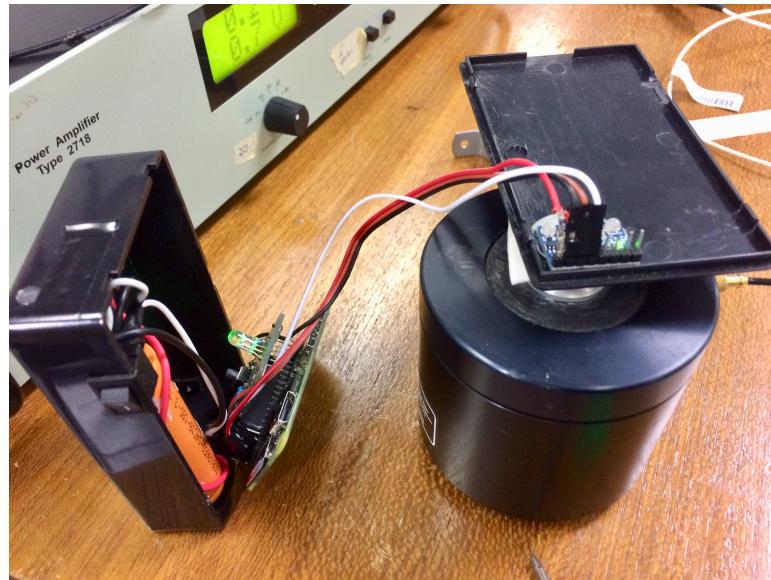


Figura 4.4: Testes com o excitador eletromecânico

O excitador foi configurado para gerar sinais em certos valores de frequência e, para cada valor, o Dispositivo IoT coletou os dados e os enviou para o servidor que, por sua vez, realizou os cálculos e exibiu os resultados nos gráficos. Foram escolhidos seis valores aleatórios de frequências dentro da região identificável: 30, 400, 500, 900, 1250 e 1400Hz. Os resultados são mostrados nas figuras a seguir, que apresentam o sinal de aceleração medido e a FTT corresponde, para cada eixo espacial:

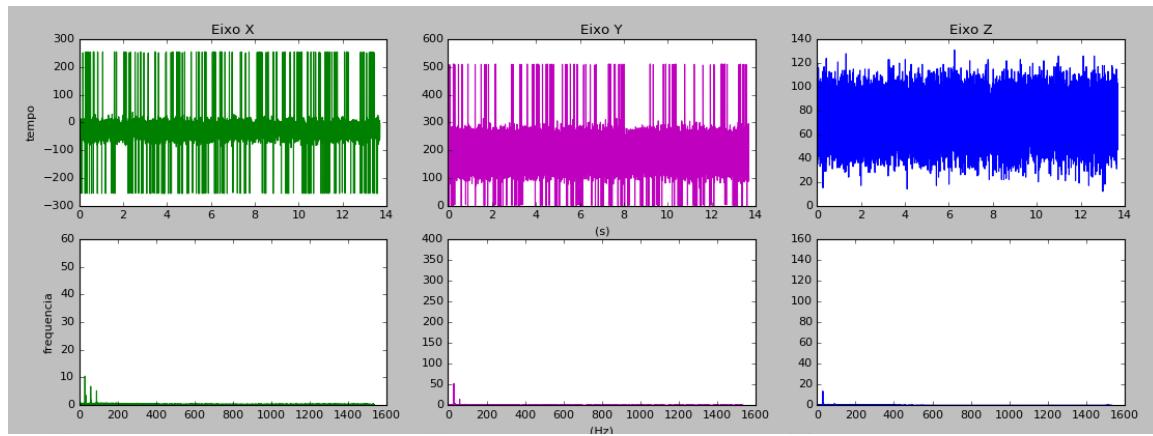


Figura 4.5: Resultados para excitador a 30Hz

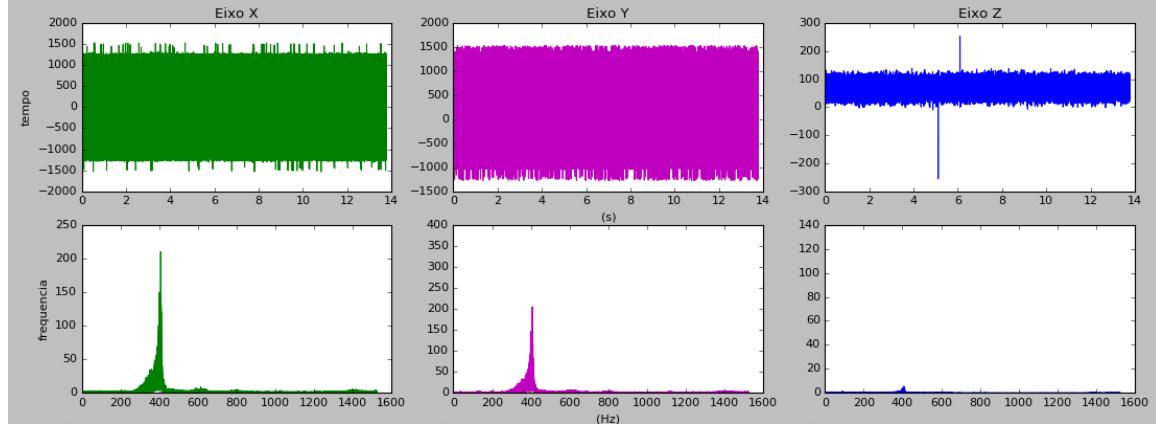


Figura 4.6: Resultados para excitador a 400Hz

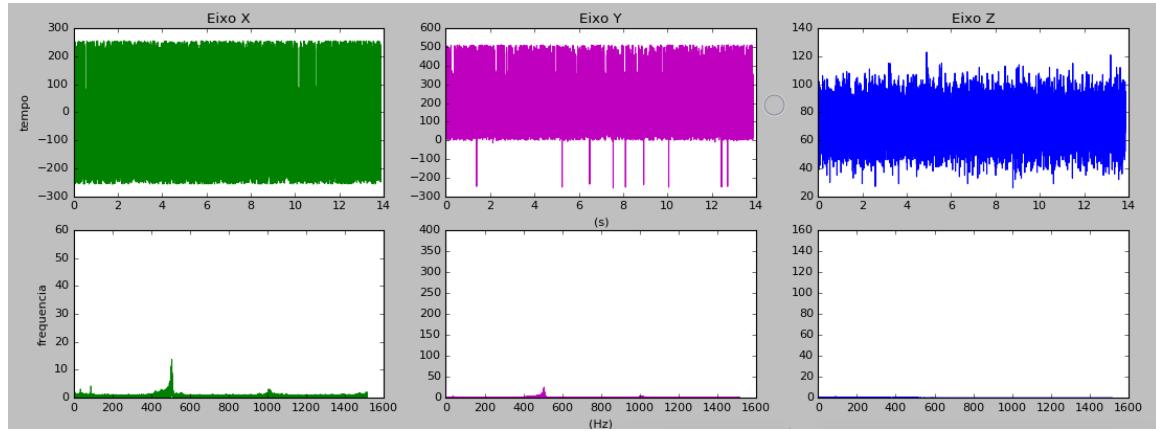


Figura 4.7: Resultados para excitador a 500Hz

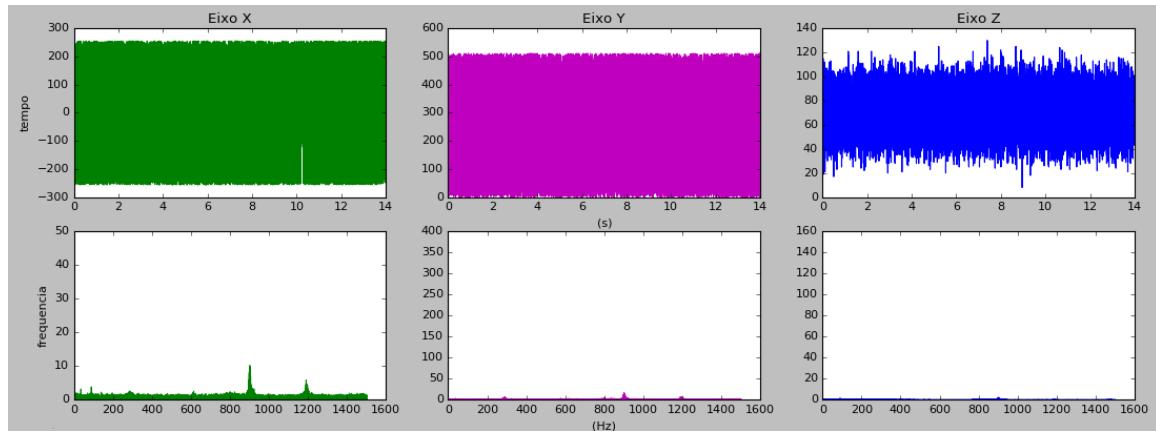


Figura 4.8: Resultados para excitador a 900Hz

Observando as figuras, percebe-se nos gráficos da FFT um peso maior ao redor da frequência do sinal gerado. Em alguns casos, também percebe-se frequências múltiplas da fundamental, que correspondem aos harmônicos.

Dessa forma, pode-se dizer que as medições de aceleração e consequentes cálculos da FFT estão coerentes com o sinal gerado pelo excitador, validando a assertividade do Dispo-

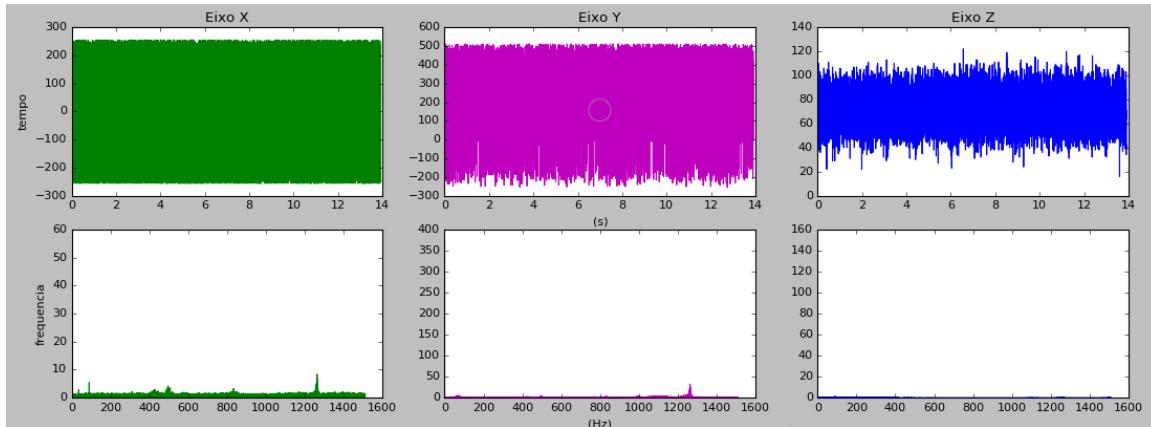


Figura 4.9: Resultados para excitador a 1250Hz

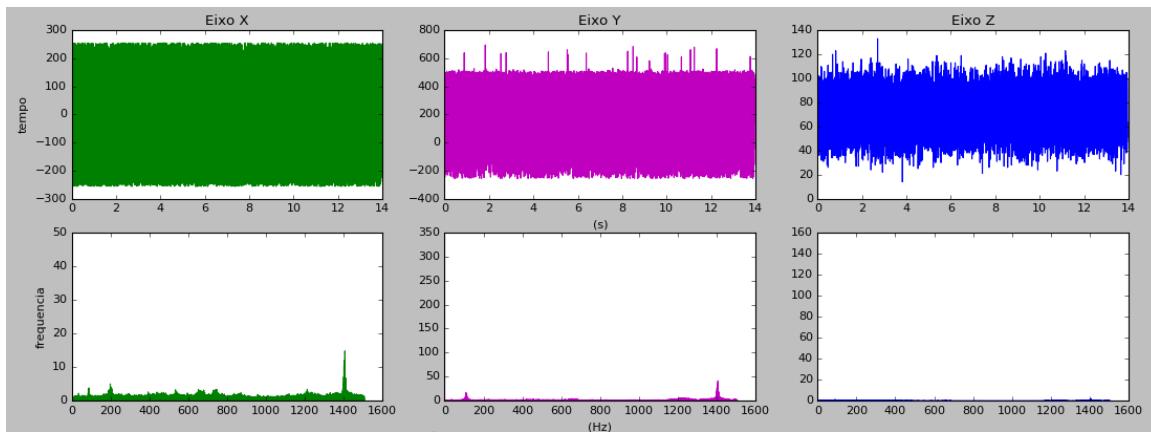


Figura 4.10: Resultados para excitador a 1400Hz

sitivo IoT.

Além disso, o valor medido de temperatura externa foi confrontado com medições de um sensor de temperatura ambiente, e sua exatidão foi confirmada (sem considerar valores fractionários - irrelevantes para a aplicação pretendida). Além disso, esta exatidão já é garantida pelas especificações do MAX6675 e pela linearidade do termopar utilizado.

4.4 Caso de aplicação

Para fins de validação do Dispositivo IoT em uma aplicação real (monitoramento de máquinas elétricas), foi utilizado uma plataforma de testes do mesmo Laboratório de Vibração.

O dispositivo foi fixado em um motor elétrico de indução, que possui um disco perfurado para se colocar parafusos que causam o desbalanceamento forçado do conjunto rotativo. A imagem a seguir (4.11) mostra esta plataforma e o dispositivo anexado ao motor.

Foram simuladas três condições principais: motor desligado; motor ligado sem desbalanceamento forçado; motor ligado com desbalanceamento forçado. A análise de Fourier para estes três casos é mostrada a seguir.



Figura 4.11: Plataforma de teste com motor de indução

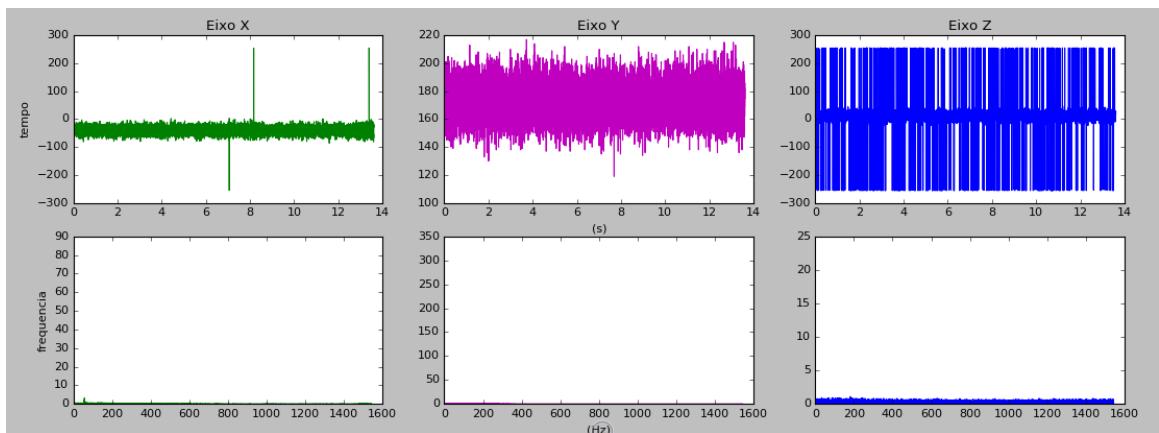


Figura 4.12: Resultados para motor de indução desligado

No caso mostrado na figura 4.12, a FFT não apresenta uma frequência expressiva. Há uma pequena proeminência em torno de 50Hz, mas não significativa. Esta pode ser causada por fontes externas de vibração (ventoinha de computadores apoiados sob a mesma bancada, por exemplo). A próxima figura mostra uma leitura para o motor ligado sem desbalanceamento.

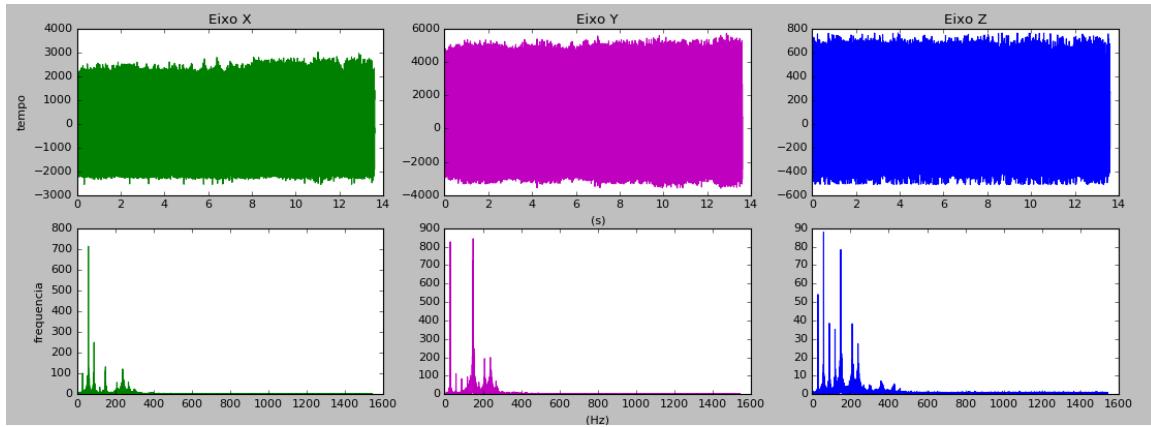


Figura 4.13: Resultados para motor de indução ligado SEM desbalanceamento forçado

Ao ligar o motor, percebe-se na figura 4.13 uma série de frequências evidentes no sinal amostrado, características naturais do conjunto mecânico em rotação.

Posteriormente, foi colocado um parafuso no disco rotativo, para gerar um desbalanceamento forçado. A imagem 4.14 mostra os resultados.

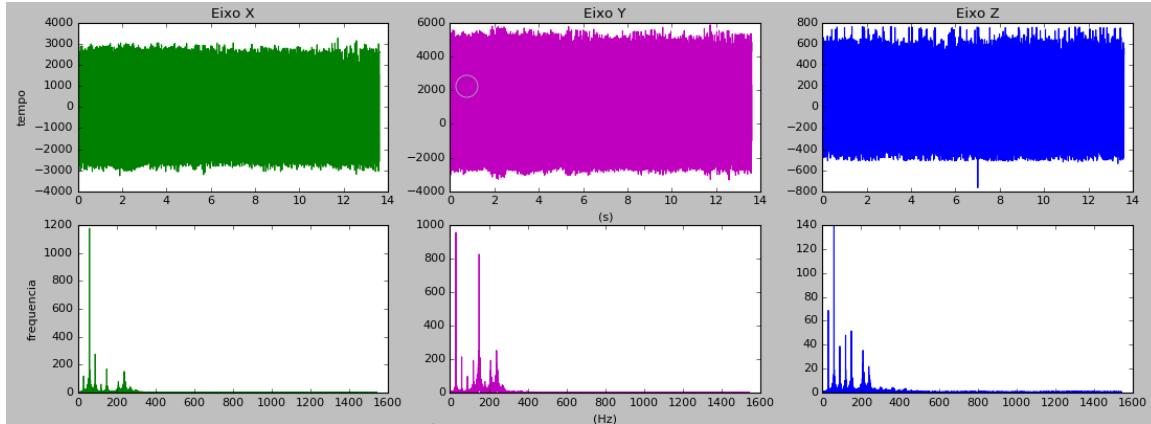


Figura 4.14: Resultados para motor de indução ligado COM desbalanceamento forçado

Neste teste foi colocado um parafuso no disco de desbalanceamento. Assim, fica evidente nos gráficos de FFT que outras frequências foram detectadas e, principalmente, as frequências observadas anteriormente (sem desbalanceamento) ganharam um maior peso.

No eixo X, por exemplo, o valor absoluto máximo verificado na FFT foi de aproximadamente 700 m/s² quando o motor estava sem desbalanceamento forçado. Com desbalanceamento, este valor foi para 1200 m/s². Isto evidencia que o Dispositivo percebeu um aumento na amplitude das vibrações quando o motor foi submetido a um desbalanceamento forçado. A figura 4.15 retrata os testes em execução.

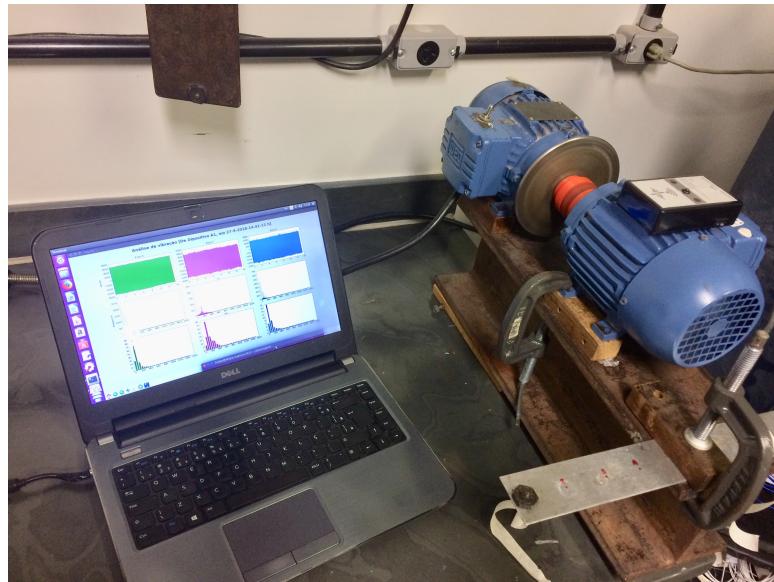


Figura 4.15: Testes com motor de indução

4.5 Interface web

Conforme apresentado na seção 3.8, foi desenvolvido uma interface web para visualização dos dados em um navegador. A figura (4.16) ilustra tal interface. Na imagem seguinte (4.17) são evidenciados os recursos apresentados na tela. Nesta interface, o usuário pode acompanhar em tempo real os resultados das medições, bem como navegar pelo histórico. Esta interface permite acompanhar a FFT dos sinais medidos, além da temperatura interna do acelerômetro e a da ponta de prova (termopar).

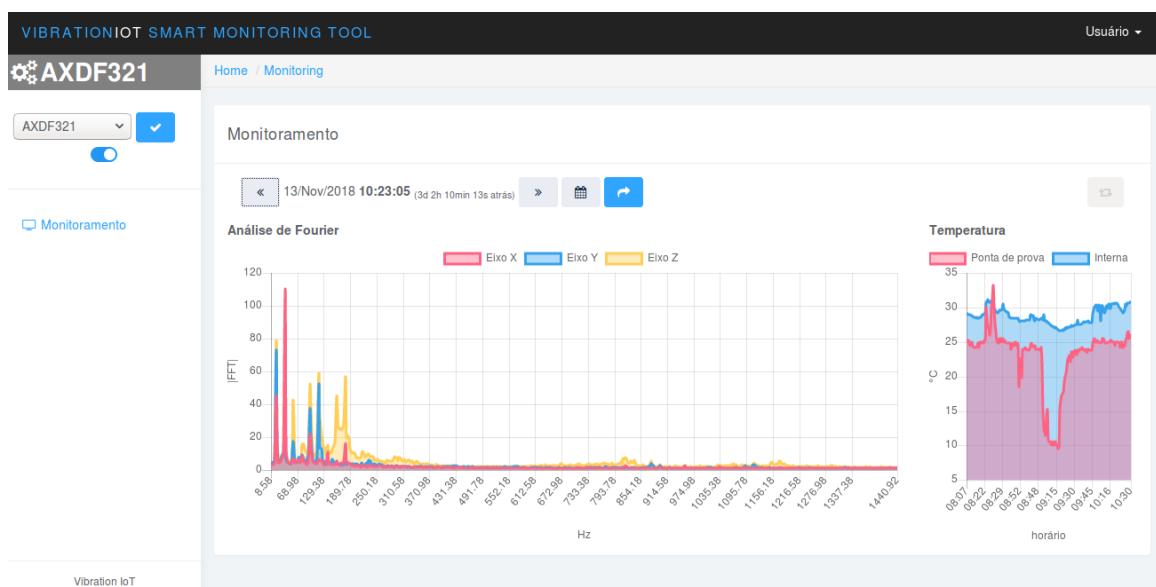


Figura 4.16: Interface Web - Exemplo

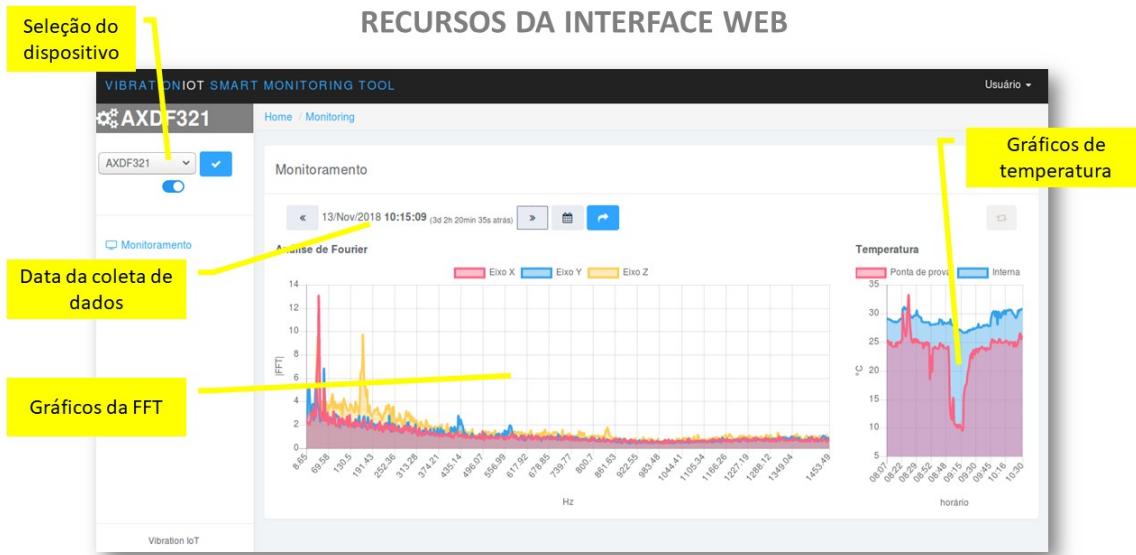


Figura 4.17: Interface Web - Recursos

4.6 Resumo do Capítulo

Este capítulo apresentou os resultados dos testes realizados, que demonstram que o Dispositivo IoT foi capaz de funcionar conforme projetado. O próximo capítulo apresenta a conclusão deste trabalho, e perspectivas futuras de melhorias e avanços.

Capítulo 5

Conclusões

5.1 Propostas de Continuidade

Existem vários ponto de melhoria no sistema projetado, a citar:

- Otimização do consumo de bateria do Dispositivo IoT;
- Desenvolvimento de uma proteção da caixa plástica, de forma a resistir a poeiras e umidade, características dos ambiente industriais, conforme os padrões e classes de proteção IP;
- Desenvolvimento de uma rede *mesh* para que, quando houverem vários Dispositivos IoT em operação, estes possam se comunicar e estabelecer uma rede mesh, dispensando a instalação de vários *gateways* para cobrir toda a área de instalação dos dispositivos;
- Desenvolvimento de um algoritomo de *machine learning* para aprender os padrões de vibração comuns da máquina, e ter condições de classificar cada amostra de dados em modos de operação normais (ligado, desligado, etc). Com isso, será também possível predizer falhas por meio do acompanhamento da evolução dos padrões de frequência medidos.

Estes são alguns dos pontos relevantes que podem ser observados para se obter um sistema mais robusto e com mais funcionalidades.

5.2 Considerações Finais

Os testes realizados (tanto com o excitador eletromecânico, quanto com o motor de indução) evidenciam que o dispositivo é uma ferramenta adequada para a medição de temperatura e vibração em máquinas elétricas em operação, sendo capaz de armazenar os dados coletados/processados e exibi-los em uma interface amigável.

Estes resultados mostram, portanto, que o Dispositivo IoT funcionou conforme projetado. O mesmo pode ser uma ferramenta útil no monitoramento de máquinas elétricas e rastreamento de falhas mecânicas/elétricas visíveis por meio da temperatura da máquina e do espectro de frequências dos sinais de vibração.

Além disso, percebe-se a interdisciplinaridade deste projeto, que envolveu e integrou conceitos de eletrônica, sensores, sistemas processadores e periféricos, comunicação com periféricos, sistemas distribuídos e redes, processamento de sinais, banco de dados, desenvolvimento web, dentre outros. Assim, este trabalho envolveu vários conhecimentos adquiridos ao longo do curso de Engenharia de Controle e Automação, reunindo-os em prol de uma solução aplicável em situações reais.

Referências Bibliográficas

- [1] *Angular - What is Angular*, 2018 (Acesso em 01/11/2018). <https://angular.io/docs>.
- [2] *Python software foundation*, 2018 (Acesso em 03/04/2018). <https://www.python.org/>.
- [3] *Sistema Operacional Raspbian*, 2018 (Acesso em 03/06/2018). <https://www.raspbian.org/>.
- [4] *Sockets for python — Low-level networking interface*, 2018 (Acesso em 03/08/2018). <https://docs.python.org/3/library/socket.html>.
- [5] *Raspberry Foundation*, 2018 (Acesso em 05/11/2018). <https://www.raspberrypi.org>.
- [6] *Raspberry Pi Zero W*, 2018 (Acesso em 05/11/2018). <https://www.raspberrypi.org/products/raspberry-pi-zero-w/>.
- [7] *Discrete Fourier Transform (numpy.fft)*, 2018 (Acesso em 10/09/2018). <https://docs.scipy.org/doc/numpy-1.15.0/reference/routines.fft.html>.
- [8] *NumPy Org*, 2018 (Acesso em 10/09/2018). <http://www.numpy.org/>.
- [9] *The Industrial Internet of Things (IIoT): the business guide to Industrial IoT*, 2018 (Acesso em 12/06/2018). <https://www.i-scoop.eu/internet-of-things-guide/industrial-internet-things-iiot-saving-costs-innovation>.
- [10] *MPU-6050 Six-Axis (Gyro + Accelerometer) MEMS MotionTrackingTM Devices*, 2018 (Acesso em 12/06/2018). <https://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/>.
- [11] *Matplotlib: Python plotting — Matplotlib 3.0.2 documentation*, 2018 (Acesso em 20/08/2018). <https://matplotlib.org/>.
- [12] *MAX6675 Cold-Junction-Compensated K-Thermocouple-to-Digital Converter*, 2018 (Acesso em 22/08/2018). <https://www.maximintegrated.com/en/products/sensors/MAX6675.html>.
- [13] *Flask - Web Development, one drop at a time*, 2018 (Acesso em 25/08/2018). <http://initd.org/psycopg/>.
- [14] *Psycopg - PostgreSQL interface for Python*, 2018 (Acesso em 25/08/2018). <http://initd.org/psycopg/>.
- [15] J.G.B.; Lambert-Torresm G.; da Silva L.E.B. Bonaldi, E.; Oliveira L.E.L.; Silva. Predictive maintenance by electrical signature analysis to induction motors. 2012.

- [16] C.; UMANS-S.D. FITZGERALD, A. E.; KINGSLEY. *Máquinas Elétricas 7ed.* AMGH, 2014.
- [17] L. FOGLIATTO, F.; RIBEIRO. *Confiabilidade e Manutenção Industrial. 1ª edição.* Elsevier, 2009.
- [18] A.; NASCIF J. KARDEC. *Manutenção: função estratégica. 3ª edição.* Rio de Janeiro: Qualitymark: Petrobrás, 2009. 384 p.
- [19] E.L.; Borges da Silva L.E.; de Oliveira L.E.L. Lambert-Torres, G.; Bonaldi. An intelligent classifier for induction motors fault diagnosis. 2003.
- [20] W. V. OTANI, M.; MACHADO. *A proposta de desenvolvimento de gestão da manutenção industrial na busca da excelência ou classe mundial.* Revista Gestão Industrial. Vol.4, n.2, 2008.
- [21] V. Venkatasubramanian. Prognostic and diagnostic monitoring of complex systems for product lifecycle management: Challenges and opportunities. 2005.
- [22] K. Ágoston. Fault detection of the electrical motors based on vibration analysis. 2015.