FACULDADE DE TECNOLOGIA DE SÃO BERNARDO DO CAMPO
“ADIB MOISÉS DIB”

ALEX MARTINS DUARTE
IZAQUE SILVA MACHADO
JHONES GOMES SANTOS
THIAGO FERREIRA DA SILVA

IOT EM MOTORES DE INDUÇÃO TRIFÁSICO

São Bernardo do Campo – SP
Junho/2024

ALEX MARTINS DUARTE
IZAQUE SILVA MACHADO
JHONES GOMES SANTOS
THIAGO FERREIRA DA SILVA

IOT EM MOTORES DE INDUÇÃO TRIFÁSICO

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de São Bernardo do Campo “Adib Moises Dib” como requisito parcial para a obtenção do título de Tecnólogo (a) em Automação Industrial.

Orientador: Prof. Esp. Marcos V. Zamboni

São Bernardo do Campo – SP
Junho/2024

ALEX MARTINS DUARTE
IZAQUE SILVA MACHADO
JHONES GOMES SANTOS
THIAGO FERREIRA DA SILVA

IOT EM MOTORES DE INDUÇÃO TRIFÁSICO

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de São Bernardo do Campo “Adib Moises Dib” como requisito parcial para a obtenção do título de Tecnólogo (a) em Automação Industrial.

Orientador: Prof. Esp. Marcos V. Zamboni

Trabalho de Conclusão de Curso apresentado e aprovado em:_____/_____/2024

Banca Examinadora:

_____

Prof. Esp. Marcos Vagner Zamboni, FATEC SBC – Orientador

_____

Prof. XXX XXXXXX, FATEC SBC – Avaliador

_____

Prof. XXX XXXXXX, FATEC SBC – Avaliador

Dedicamos esse trabalho aos nossos familiares que nos incentivaram, aos nossos professores que nos ensinaram e aos colegas que de alguma forma também contribuíram.

.

Agradecemos ao Prof. Esp. Marcos Vagner Zamboni pela ajuda durante a elaboração deste trabalho. Agradecemos também a Fatec SBC.

“Há duas formas de viver a vida: uma é acreditar que não existem milagres, a outra é acreditar que todas as coisas são um milagre”

ALBERT EINSTEIN

RESUMO

Este trabalho de conclusão de curso, aborda a implementação de um sistema para monitoramento de temperatura, corrente e vibração, em motores de indução. Ele envia as medições em tempo real, para um broker na internet via protocolo MQTT, onde podem ser visualizadas por qualquer computador conectado à internet, com o software Node.js instalado. Para isso, foram utilizadas duas plataformas modelo Heltec ESP32 v2 com LoRa. Uma delas é a emissora, que fica ligada próxima ao motor, recebendo os dados dos sensores de temperatura e vibração que foram instalados externamente na carcaça deste, e do sensor de corrente conectado em uma das fases deste. Caso a corrente medida passe do valor selecionado na programação em linguagem C++, a placa ESP32 emissora envia um pulso no módulo relé, instalado no contato de selo do motor, desligando-o. Para religá-lo alguém deverá verificar em campo o que ocorreu, e apertar manualmente o botão ligar, no painel elétrico. A outra plataforma Heltec ESP32 v2 com LoRa, é a receptora que pode estar ligada a uma distância de até 3,5 km em área aberta, recebendo as informações do motor via protocolo de rádio LoRa, e enviando estas para a internet via protocolo MQTT, garantindo transmissão rápida e confiável. Esses dados são apresentados em um dashboard interativo desenvolvido com Node-Red, que oferece visualização em tempo real, e alertas de condições anômalas. A interface amigável do Node-Red facilita a interpretação dos dados, permitindo intervenções de manutenções preventivas. Os testes realizados confirmaram a eficácia do sistema, que demonstrou capacidade de detectar anomalias em tempo real, contribuindo para a redução dos custos operacionais, e aumentando a vida útil dos motores de indução. A solução se mostrou robusta e escalável, proporcionando uma ferramenta poderosa para a manutenção preventiva na indústria. Futuras melhorias podem incluir mais sensores e algoritmos avançados de análise de dados, integrar com banco de dados para gerenciamento e manutenções preditivas, aproveitando assim o contínuo avanço das tecnologias de IoT.

Palavras-chave: Motores. Sensores. ESP32. MQTT. Node-RED.

ABSTRACT

This course completion work addresses the implementation of a system for monitoring temperature, current and vibration in induction motors. It sends the measurements in real time to an internet broker via the MQTT protocol, where they can be viewed by any computer connected to the internet, with the Node.js software installed. For this, two Heltec ESP32 v2 platforms with LoRa were used. One of them is the transmitter, which is connected close to the motor, receiving data from the temperature and vibration sensors that were installed externally on its housing, and from the current sensor connected to one of its phases. If the measured current exceeds the value selected in programming in C++ language, the sending ESP32 board sends a pulse to the relay module, installed on the motor's seal contact, turning it off. To turn it back on, someone must check what happened in the field and manually press the power button on the electrical panel. The other Heltec ESP32 v2 platform with LoRa is the receiver that can be connected at a distance of up to 3.5 km in an open area, receiving information from the engine via the LoRa radio protocol, and sending this to the internet via the MQTT protocol, ensuring fast and reliable transmission. This data is presented in an interactive dashboard developed with Node-Red, which offers real-time visualization and alerts for anomalous conditions. Node-Red's user-friendly interface facilitates data interpretation, allowing preventive maintenance interventions. The tests carried out confirmed the effectiveness of the system, which demonstrated the ability to detect anomalies in real time, contributing to the reduction of operating costs and increasing the useful life of induction motors. The solution proved to be robust and scalable, providing a powerful tool for preventive maintenance in the industry. Future improvements may include more sensors and advanced data analysis algorithms, integration with databases for management and predictive maintenance, thus taking advantage of the continuous advancement of IoT technologies.

Keywords: Engines. Sensors. ESP32. MQTT. Node-RED.

LISTA DE FIGURAS

SUMÁRIO

INTRODUÇÃO

Atualmente, muitos motores de indução instalados nas indústrias e em edifícios comerciais e residenciais, não possuem sensores medindo a corrente de fase, nem a temperatura e a vibração externa no estator. Este projeto consiste em um sistema de automação, que pode ser instalado nestes motores, auxiliando na manutenção preventiva destes.

O objetivo é utilizar tecnologias de IoT, para monitorar à longa distância via sinal de radiofrequência LoRA, sensores de vibração e temperatura conectados externamente no estator de um motor de indução, e o sensor de corrente em uma das fases deste. Essas informações serão enviadas para um serviço na nuvem, permitindo que a equipe de manutenção, onde esse motor estiver instalado, possam acessar esses dados em qualquer local, verificando se há grandes variações nas medições, prevendo possíveis problemas nos componentes destes.

As vantagens desse projeto são: Não precisar parar o funcionamento do motor para a instalação dos sensores; não precisar modificar a programação de supervisório, CLP ou inversor de frequência caso existam instalados; e o baixo investimento financeiro para implantar esse sistema.

Esse trabalho é dividido em três capítulos:

Capítulo 1 – Fundamentação teórica: se encontram as teorias que sustentam todo o desenvolvimento do nosso projeto.

Capítulo 2 – Metodologia: é o percurso necessário a ser trilhado para o desenvolvimento de uma pesquisa, fornecendo técnica, métodos e procedimentos.

Capítulo 3 – Desenvolvimento do projeto: encontram-se o passo a passo da construção e desenvolvimento do projeto. Estabelecendo uma ligação entre a fundamentação teórica, sendo demonstrado com ilustrações.

E finalmente, as Considerações finais: onde são descritos os objetivos e as justificativas do projeto, trazendo à tona as relações entre as aplicações, teorias, objetivos alcançados, pontos a serem destacados e possíveis sugestões para melhorias futuras.

1 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo encontram-se as teorias que dão sustentação ao desenvolvimento do projeto intitulado IoT em Motores Industriais.

1.1 Início dos motores elétricos

Em 1879, a empresa Siemens & Halske apresentou, na feira industrial de Berlim, a primeira locomotiva elétrica, com uma potência de 2kW.

A nova máquina de corrente contínua apresentava vantagens em relação à maquina a vapor, a roda d'água e à força animal. Entretanto, o alto custo de fabricação e a sua vulnerabilidade em serviço (por causa do comutador) marcaram-na de tal modo que muitos cientistas dirigira sua atenção para o desenvolvimento de um motor elétrico mais barato, mais robusto e de menor custo de manutenção. Entre os pesquisadores preocupados com esta idéia, destacam-se o iugoslavo Nikola Tesla, o italiano Galileu Ferrarris e o russo Michael Von Dolivo-Dobrovolski. Os esforços não se restringiram somente ao aperfeiçoamento do motor de corrente contínua, mas também se cogitou de sistemas de corrente alternada, cujas vantagens já eram conhecidas em 1881.

Em 1885, o engenheiro eletricista Galileu Ferraris construiu um motor de corrente alternada de duas fases. Ferraris, apesar de ter inventado o motor de campo girante, concluiu erroneamente que os motores construídos segundo este princípio poderiam, no máximo, obter um rendimento de 50% em relação a potência consumida. E Tesla apresentou, em 1887, um pequeno protótipo de motor de indução bifásico com rotor em curto-circuito. Também esse motor apresentou rendimento insatisfatório, mas impressionou de tal modo a empresa norte-americana Westinghouse, que esta lhe pagou um milhão de dólares pelo privilégio da patente, além de se comprometer ao pagamento de um dólar para cada cavalo de potência que viesse a produzir no futuro. O baixo rendimento desse motor inviabilizou economicamente sua produção e três anos mais tarde as pesquisas foram abandonadas.

Foi o engenheiro eletricista Dobrowolsky, da empresa AEG, de Berlim, entrou em 1889 com o pedido de patente de um motor trifásico com rotor de gaiola. O motor

apresentado tinha uma potência de 80 Watts, um rendimento aproximado de 80% em relação a potência consumida e um excelente conjugado de partida. As vantagens do motor de corrente alternada para o motor de corrente contínua eram marcantes: construção mais simples, silencioso, menos manutenção e alta segurança em operação. Dobrowolsky desenvolveu, em 1891, a primeira fabricação em série de motores assíncronos, nas potências de 0,4 kW a 7,5 kW.

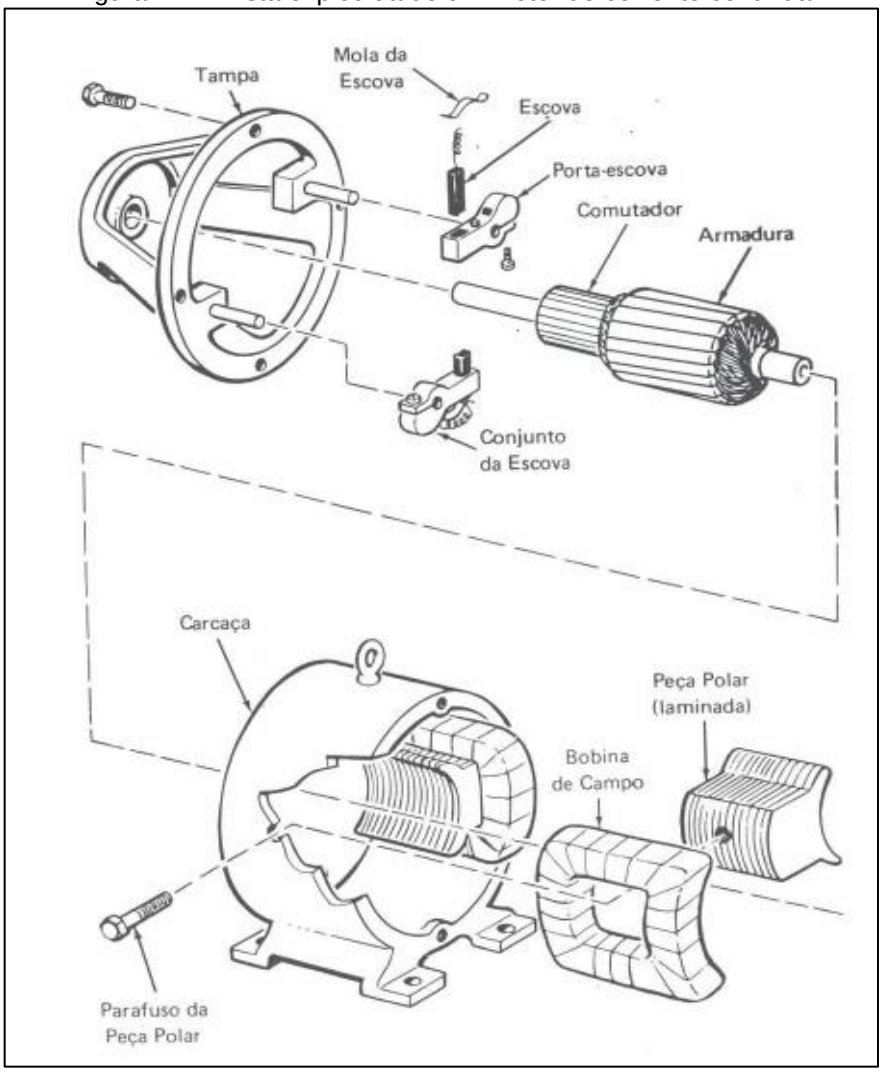
1.2 Vantagens dos motores elétricos

O motor de indução trifásico, também conhecido por MIT, é o mais utilizado em plantas industriais pela sua robustez, vantagens como custo, consumo de energia e fácil manutenção, fatores que impactam diretamente em seu ciclo de vida (CAPELLI, 2013).

Os motores elétricos oferecem várias vantagens significativas em comparação com os motores de combustão interna. Primeiramente, eles são altamente eficientes, convertendo uma maior porcentagem de energia elétrica em energia mecânica, o que resulta em menor consumo de energia e redução de custos operacionais. Além disso, motores elétricos são mais ecológicos, pois não produzem emissões, contribuindo para um ambiente mais limpo. Sua manutenção é geralmente mais simples devido ao menor número de partes móveis, o que reduz a necessidade de reparos frequentes. Outra vantagem é a operação silenciosa, que reduz a poluição sonora e melhora o conforto em aplicações residenciais e comerciais (PINYI MOTOR).

Mesmo com o passar dos anos, os motores de corrente contínua possuem um custo mais elevado e, além disso, precisam ser alimentados em uma fonte de corrente contínua, ou seja, converter a corrente alternada em contínua. Estes motores funcionam com velocidade ajustável entre amplos limites e apresentam controles de grande flexibilidade e precisão. Por isso, seu uso é restrito a casos especiais em que estas exigências compensam o custo muito mais alto da instalação e da manutenção (WEG MOTORES, 2017).

Figura 1.1 – Vista explodida de um motor de corrente contínua

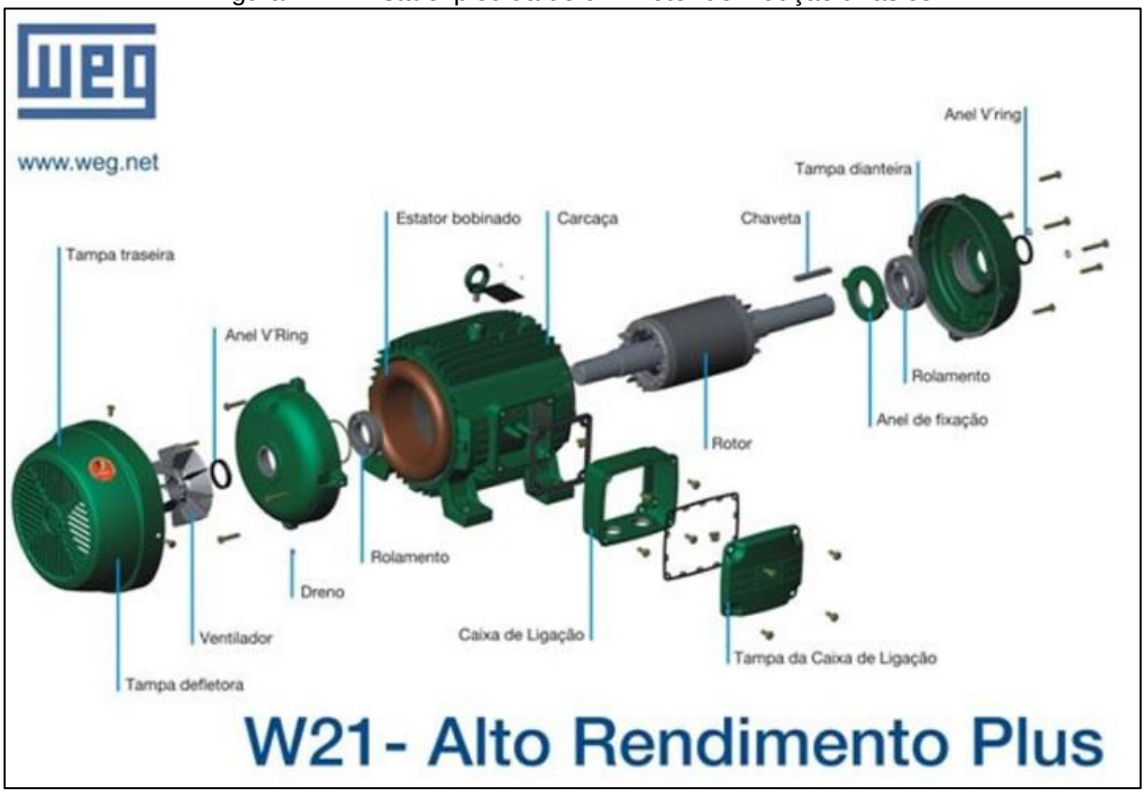


Fonte: https://pt.slideshare.net/danilomagalhaes104/apostila-de-maquinas-cc, 2006

Já os motores de corrente alternada MIT (Motores de Indução Trifásico) são os mais utilizados dentro das plantas industriais, consideras por sua robustez, simplicidade e baixo custo, se dividem em motores síncronos e de indução, tem uma diferença na velocidade do rotor para estator que temos o efeito de escorregamento justamente para garantir seu funcionamento.

O motor síncrono funciona com velocidade fixa, sem interferência do escorregamento. Utilizado normalmente para grandes potências devido ao seu alto custo em tamanhos reduzidos. Motor de indução funciona normalmente com uma velocidade constante, que varia ligeiramente com a carga mecânica aplicada ao eixo (WEG MOTORES, 2017). A Figura 1.2 mostra uma vista explodida das partes de um MIT.

Figura 1.2 – Vista explodida de um motor de indução trifásico



Fonte: https://maquinaseletricasi.blogspot.com/2013/04/aula-30-motor-monofasico-com-quatro.html, 2023

Existem dois tipos diferentes de rotores de motor de indução, sendo rotor de gaiola de esquilo e rotor bobinado. O rotor tipo gaiola de esquilo é constituído por barras condutoras colocadas em serie e encaixadas em ranhuras na superfície do rotor, elas são postas em curto-circuito em suas extremidades pelos chamados anéis. Já o rotor bobinado possui um conjunto de enrolamentos parecido com o do 19 estator. Os enrolamentos do rotor são colocados em curto-circuito por meio de escovas (CHAPMAN, 2013).

Ao alimentar o enrolamento do estator, uma tensão é induzida no enrolamento do rotor. Esse campo eletromagnético entre estator e rotor faz com que ocorra o movimento (Chapman, 2013).

1.3 Principais defeitos e manutenções nos motores de indução trifásico

Quando pensamos em manutenção industrial, um dos objetivos é garantir que o equipamento opere o máximo de tempo possível sem falhar. No entanto, sabemos

que vários componentes estão sujeitos a elas. Falhas em motores elétricos podem ter várias causas. O motor elétrico é o principal responsável pelo movimento em máquinas e equipamentos, logo, merece atenção especial para evitar que ele seja o motivo de ter a sua produção interrompida.

Se você trabalha em um setor de manutenção, certamente já viu algum motor falhar. Sabe muito bem o impacto que isso traz para a linha de produção. Por esse motivo, vamos abordar as principais falhas em motores elétricos em um ambiente industrial. Dessa forma, poderemos prevenir que elas ocorram e causem impacto direto em sua produtividade.

Antes de tudo, entenda que se pode evitar várias destas falhas se sua empresa adotar programas de manutenção preditiva. É de fato a maneira mais confiável para impedir que você seja pego de surpresa. Principalmente porque este tipo de manutenção atua com o monitoramento de condição dos equipamentos. Permite identificar a falha ainda em sua origem, ou seja, antes do equipamento parar por uma quebra ou falha total.

Confira agora os tipos de falhas em motores elétricos que podem ocorrer em um ambiente industrial:

● Desgaste dos rolamentos: Segundo dados de um estudo sobre falhas em motores elétricos da Universidade Federal de Minas, mais da metade das manutenções estão associadas aos rolamentos do motor. Os ruídos excessivos, vibrações, aumento da temperatura de funcionamento, entre outros, são os principais indícios de que há problemas com o rolamento Gerais (Araújo, 2011).

Vários fatores estão entre as causas destas falhas, tais como, excesso ou falta de graxa e óleo lubrificante. Além de reduzir o atrito do rolamento, a graxa de rolamento ou óleo lubrificante também o protege de agentes externos que podem danificar o componente como, por exemplo a umidade e sujeira. Outra função do lubrificante é manter a temperatura de trabalho conforme especificação do projeto.

O desalinhamento do motor pode causar esforço excessivo em cima do rolamento, ocasionando desgaste prematuro da peça. Sobreaquecimento, pressão exercida de forma irregular, eixo e/ou mancal com irregularidades na superfície,

armazenamento dos rolamentos novos de forma errada e substituição de rolamentos por outros de características diferentes das especificadas pelo fabricante são também algumas causas do desgaste dos rolamentos.

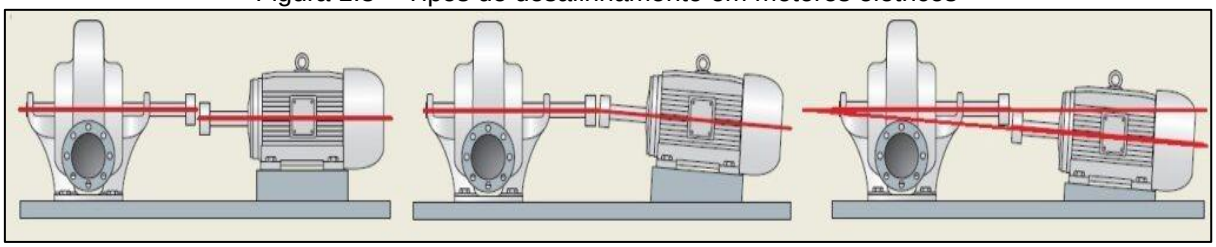
Alguns cuidados com o motor são essenciais e indispensáveis, tais como, guardar o motor com os rolamentos engraxados. Isso evita a ação da umidade ou acúmulo de sujeira. Lembre-se de que existem as graxas e óleos específicos para cada tipo de peça e função. Na dúvida consulte o fabricante. O eixo e o mancal também devem estar o mais uniforme possível. Consulte os parâmetros de regularidade da superfície destes, para evitar qualquer tipo de imperfeição.

Não lave os rolamentos novos. Eles já vêm de fabrica prontos para serem instalados. Na hora da substituição fique atento as características de cada rolamento. Certifique-se de que o peso está distribuído de maneira uniforme sobre toda área do rolamento.

● Desalinhamento: Ruídos excessivos, sobreaquecimento, consumo elevado de energia, trepidações, baixo rendimento, falha de execução, todos esses podem ser sinais de desalinhamento. Existem algumas regras quando vamos fazer o acoplamento entre o motor e a carga que será acionada.

O alinhamento de eixo entre as duas partes deve ser feito corretamente, obedecendo aos parâmetros que ajudam a evitar o desgaste prematuro e as falhas no motor, que ficam sob a influência da pressão que o motor exerce como, por exemplo, os mancais e rolamentos. A avaliação desta falha no motor elétrico é feita através da análise de posicionamento entre as partes, que são divididas em três indicadores: paralela, angular ou combinado. A Figura 1.3 demonstra isso com mais detalhes.

Figura 1.3 – Tipos de desalinhamento em motores elétricos

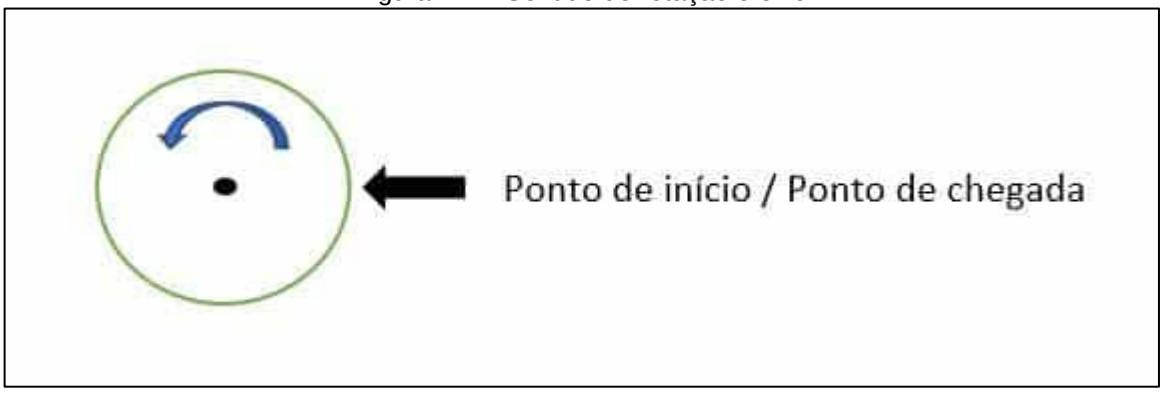


Fonte: https://www.abecom.com.br/falhas-em-motores-eletricos, 2022

Para evitar falhas em motores elétricos por causa do desalinhamento é recomendado usar acoplamentos flexíveis sempre que for possível; fazer alinhamento de eixos a laser, garantindo maior precisão; da mesma forma com o alinhamento dos mancais, monitorar a temperatura de trabalho do motor e monitorar as vibrações.

● Desbalanceamento de eixo: Sabemos que o movimento produzido por um motor é circular. Em outras palavras, após um ciclo completo o ponto final chega no mesmo ponto de partida. A seta azul na Figura 1.4 representa o sentido de rotação do motor, e o ponto preto representa seu eixo.

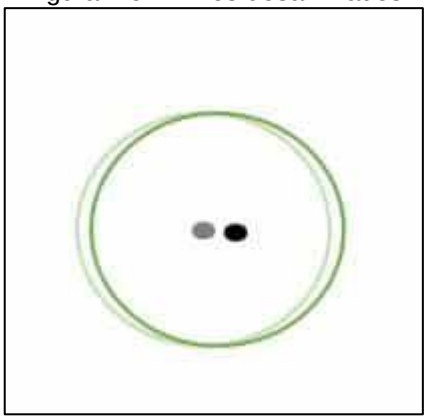
Figura 1.4 – Sentido de rotação e eixo



Fonte: https://www.abecom.com.br/falhas-em-motores-eletricos, 2022

O eixo de rotação da carga que será acionada tem que estar em harmonia, ou seja, tem que estar alinhado com o eixo de rotação do motor, o que não acontece na Figura 1.5.

Figura 1.5 – Eixos desalinhados

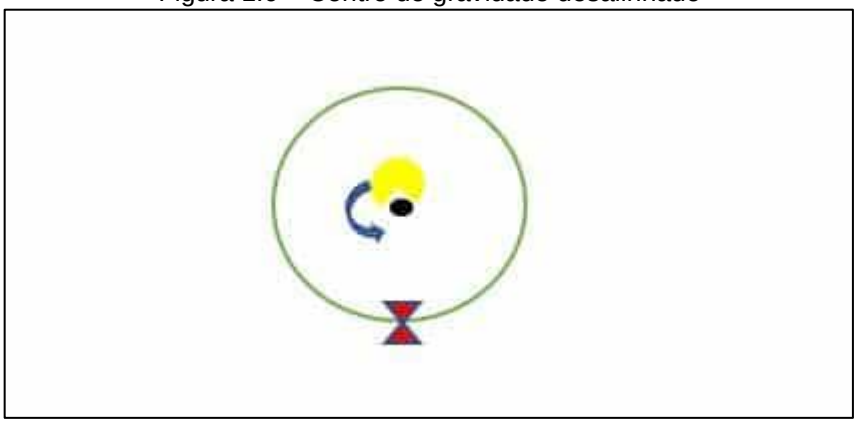


Fonte: https://www.abecom.com.br/falhas-em-motores-eletricos, 2022

Esse desencontro dos eixos cria uma força que na física chamamos de Força Centrífuga. Essa energia empurra o centro de gravidade da rotação para fora do círculo, como se quisesse alterar o ponto de chegada do movimento.

Chamamos o exemplo acima de desbalanceamento estático em eixos paralelos. O outro tipo de classificação desta falha no motor elétrico é denominado de Conjugado. Neste caso o centro de gravidade do rotor e da carga não estão alinhados.

Figura 1.6 – Centro de gravidade desalinhado



Fonte: https://www.abecom.com.br/falhas-em-motores-eletricos, 2022

Na Figura 1.6, os eixos de ambas as partes estão alinhados. O centro de gravidade do motor está em cima do seu eixo, e o centro de gravidade da carga está representado pela figura amarela.

A solução nessa situação é colocar um contrapeso, representado pelo ícone vermelho. Esta ação irá alinhar os centros gravitacionais. E a última, e não menos importante, classificação de desbalanceamento, é denominada Dinâmico, que surge quando ambos os casos citados acima acontecem simultaneamente.

● Folga do eixo: Com o tempo, ou com peças feitas com materiais de baixa resistência, o movimento do motor pode ocasionar desgastes que alteram o ajuste das peças, criando folga entre elas.

Isso é extremamente prejudicial para seu bom funcionamento. A classificação se dá conforme a orientação em que ocorre o desgaste: axial ou radial.

A manutenção deve estar atenta aos sintomas de problemas com folga de eixo como, por exemplo: ruídos excessivos, sobreaquecimento, alto consumo de energia,

trepidações e baixo rendimento. Para evitar esses problemas é importante monitorar ou fazer inspeções preventivas, além de usar peças feitas com materiais resistentes e usar os ajustes corretos durante a montagem.

● Pé suave: Esse termo também tem a ver com o desalinhamento entre as partes. Mas a causa desta falha no motor elétrico está relacionada à superfície onde ele e carga estão fixos. Nessa situação, motor e carga não estão no mesmo plano, ou estão em plano inclinado que dispõe às partes em diferentes alturas.

Quando isso acontece, pode gerar esforço excessivo no propulsor, vindo a danificá-lo. É importante fazer uma análise da superfície de instalação dos conjuntos. Depois disso, criar soluções para que esse problema não ocorra.

● Sobrecarga: A sobrecarga é uma falha do motor elétrico, que acontece quando há uma exigência de torque no motor além da sua capacidade máxima. Tudo que causa esforço excessivo pode vir a sobrecarregar o motor, e isso produz um efeito “bola de neve”. Ou seja, qualquer um dos problemas citados anteriormente, mesmo que isolados, se não forem descobertos ainda na fase inicial, pode culminar em todos ao mesmo tempo.

Os sintomas podem ser superaquecimento, consumo excessivo de corrente e torque insuficiente. Depois que o problema já ocorreu, só resta a manutenção corretiva afim de trocar as peças que sofreram avarias. Como resultado disso, terá que parar a produção e os custos de manutenção sobem.

● Transientes de tensão: Desse tópico em diante vamos falar sobre as anomalias existentes na rede de energia, que afetam diretamente os motores elétricos.

O termo transiente significa uma coisa que não permanece, ou seja, é passageira ou transitória. O transiente de tensão pode ser oscilatório ou impulsivo. Para exemplificar, considere as descargas atmosféricas (raios e relâmpagos). Elas podem ocasionar um alto pico de tensão na rede elétrica por uma faixa de tempo que não costuma ser longa. No entanto, é o suficiente para danificar aparelhos conectados à energia elétrica.

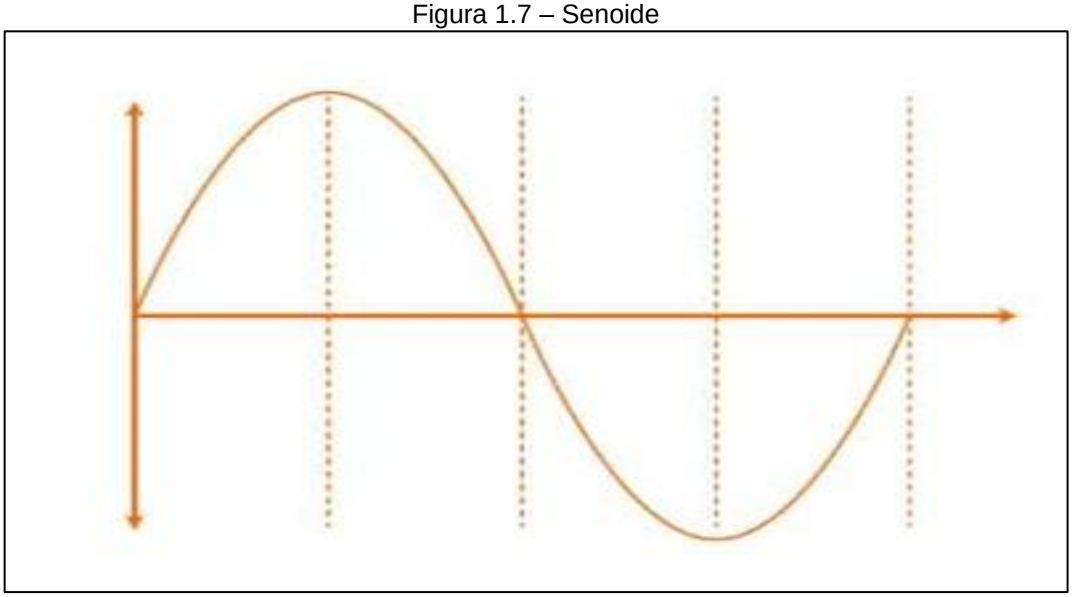
As modificações oscilatórias são mais difíceis de perceber e costumam apresentar seus danos a médio e longo prazo. Transiente impulsivo provem de fontes

externas, e as oscilatórias ocorre dentro da própria rede a partir de manobras de carga, comutação de bancos de capacitores, entre outros.

Para evitar esse tipo de problema, deve-se instalar dispositivos contra surtos de tensão, filtros ou supressores, na entrada de energia do motor.

● Distorção de harmônicas: A energia elétrica caminha pelos condutores em forma de ondas. Em um plano cartesiano conseguimos observar esse movimento variando de forma equivalente entre a área positiva e negativa (corrente alternada).

Figura 1.7 – Senoide



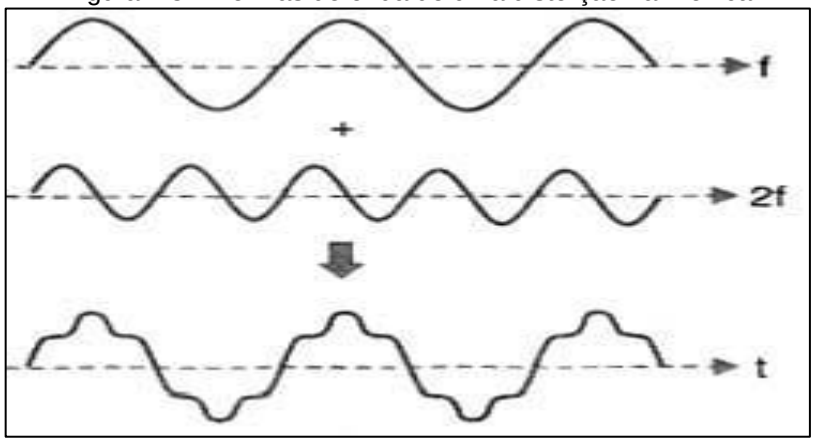
Fonte: https://www.saladaeletrica.com.br/corrente-alternada-e-tensao-alternada, 2018

O consumo dessa energia é classificado de duas formas: linear e não-linear. Linear são cargas que consomem energia sem modificar as características físicas da onda. Nesse caso a relação entre tensão e corrente são lineares.

Quando a carga possui componentes como diodos, transistores ou qualquer tipo de semicondutor, que tem como característica a capacidade de alterar a relação senoidal entre tensão e corrente, temos um sistema não linear.

A não linearidade entre essas grandezas causam deformidades nas características físicas da onda, e é esse efeito que chamamos de distorção harmônica.

Figura 1.8 – Formas de onda de uma distorção harmônica



Fonte: https://www.newtoncbraga.com.br/index.php/eletronica/52-artigos-diversos/5709-art742, [s.d.]

Como podemos observar, é possível prever este fenômeno se analisar os tipos de componentes presente em uma carga. A partir disso é só acrescentar no projeto filtros para minimizar os efeitos negativos dessa distorção.

● Fases desbalanceadas: Em motores trifásicos, ou bifásicos, a disponibilidade de energia elétrica para os condutores de alimentação tem que ser de forma íntegra, ou seja, a corrente que circula no cabeamento tem que estar em equilíbrio.

O consumo excessivo de corrente em uma das fases pode causar sérios danos a toda estrutura relacionada aos motores. Inclusive servir de ignição para incêndios. O desbalanceamento se apresenta em forma de superaquecimento, derretimento da capa isolante dos condutores etc.

Isso acontece por conta de projetos mal dimensionados quanto ao consumo de energia, ou pelo acréscimo posterior de cargas que não estavam previstas. Por isso, toda e qualquer alteração na planta deve ser previamente analisada afim de evitar que ocorram problemas como estes.

A norma NBR 5410 aborda com mais detalhes sobre esse tema, e é muito importante que os profissionais da área elétrica saibam e conheçam tudo sobre a norma.

● Corrente sigma: Sigma é a denominação das correntes parasitas presentes em um condutor elétrico, e que podem diminuir a vida útil do mesmo. Elas são geradas através da variação do campo eletromagnético, que surge com o movimento dos elétrons.

Esta falha no motor elétrico está associada a capacidade de condução elétrica dos condutores, que tem a tendência de ir diminuindo ao longo do tempo. Por sua característica indutiva, esse tipo de corrente pode ser usado propositalmente para gerar calor por indução, o que é comum em eletrodomésticos como em um forno elétrico. Um problema como este pode ser evitado avaliando as capacidades de indutância e reatância dos condutores elétricos.

1.4 Monitoramento

Para verificar o estado em que um motor se encontra, torna-se necessário o uso de técnicas e sensores de monitoramento. No entanto, monitorar qualquer equipamento diz respeito ao monitoramento de grandezas, que por sua vez, nos fornecem dados que indicam a situação atual do equipamento.

● Monitoramento de grandezas: Segundo Thomazini e Albuquerque (2010) no estudo da automação em sistemas industriais, é preciso determinar condições (ou variáveis) do sistema. É necessário obter os valores das variáveis físicas do ambiente a ser monitorado, e este é o trabalho para os sensores. Na Figura 1.9 temos exemplos de grandezas a serem medidas e o dispositivo de entrada (sensor) com o seu atuador.

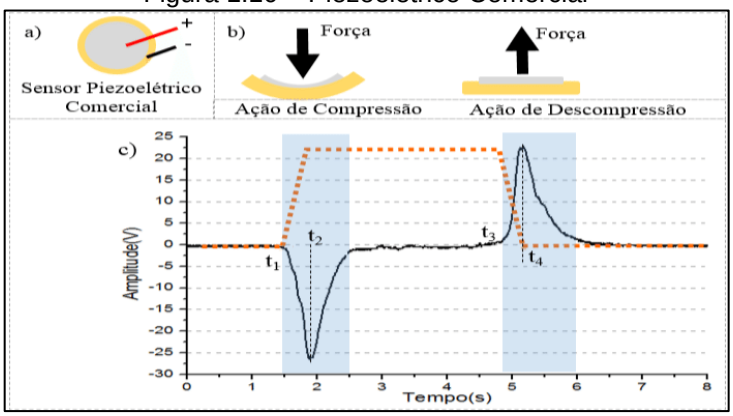
Figura 1.9 – Grandeza a ser medida

Grandeza a ser Medida	Dispositivo de Entrada (Sensor)	Dispositivo de Saída (Atuador)
Intensidade da Luz	Fotoresistor (LDR) Fotodiodo Fototransistor Célula Solar	Luzes & Lâmpadas LED's & Displays Fibra Óptica
Temperatura	Par Termoelétrico Termistor Termostato Detector de Temperatura Resistivo	Aquecedor Ventilador
Força/Pressão	Extensômetro Interruptor de Pressão Células de Carga	Eletroímã Dispositivo de Vibração Elevadores
Posição	Potenciômetro Codificadores Interruptor Óptico LVDT	Motor Solenoide Medidor de Painel
Velocidade	Acoplador Óptico Tacogerador Sensores de Efeito Doppler	Motores AC/DC Motor de Passo Freio
Som	Microfone de Carvão Cristal Piezoelétrico	Alto-falante Buzzer

Fonte: https://www.electronics-tutorials.ws/io/io_1.html, 2023

● Piezoelétricos: A Piezoeletricidade é uma propriedade da qual alguns dielétricos têm de desenvolver uma polarização ao serem submetidos a uma tensão mecânica, criando cargas de polarização gerando um campo elétrico. Tal efeito sendo descoberto no final do Século XIX por Pierre Curie. Na indústria utilizamos para diversas aplicações destacando medidores de força como pressão, vibrações e afins. Nos sensores de vibração para motores utilizam de tal princípio para analisar o comportamento do motor ao longo do tempo e seu desgaste em seu eixo além de folgas ou excesso de peso na carga. A Figura 1.10 apresenta o processo em exemplo (Machado, 2004).

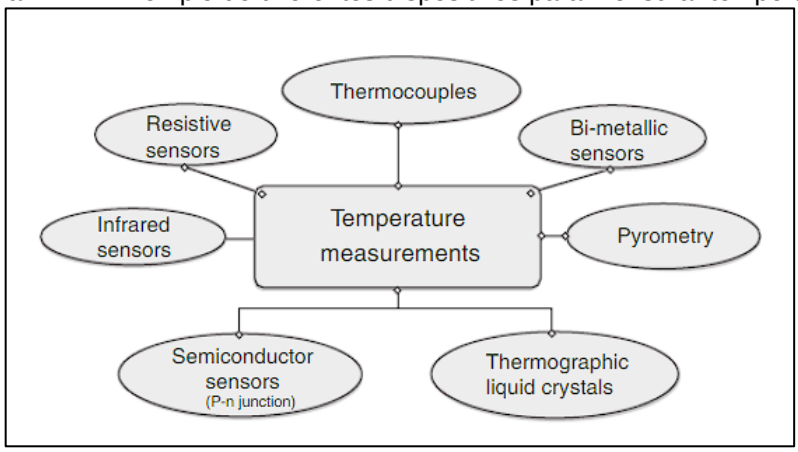
Figura 1.10 – Piezoelétrico Comercial



Fonte: ELIAS, JAIR, BISSI, STEVAN, 2016

● Sensor de temperatura: Sensores de temperaturas possuem uma vasta gama de aplicações e modelos, podendo dar diferentes possibilidades de mensurar a temperatura de um ambiente.

Figura 1.11 – Exemplo de diferentes dispositivos para mensurar temperatura



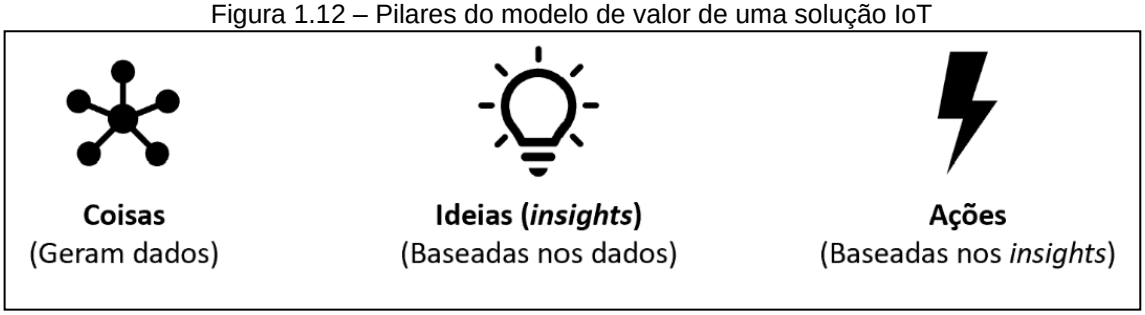
Fonte: WILE, 2010

Umas das aplicações são para o monitoramento de motores atualmente alguns motores trifásicos contam com sensores internos para medição da temperatura ou comprados a parte, e ou conjunto de sensores como o “Motor scan” da WEG captam esta grandeza, para controle do ativo. Os sensores são alocados dentro da máquina no processo de construção em pontos inacessíveis e cruciais. Esse método é utilizado para medição de temperatura dos enrolamentos do estator em máquinas de corrente alternada. Os sensores devem ser instalados ao longo do enrolamento, no mínimo, seis unidades, sendo utilizados dois sensores por fase.

1.5 IoT

Internet of Things (IoT) ou, internet das coisas em português, é um sistema que permite tornar objetos comuns, em dispositivos inteligentes, fazendo com que estes passem a fornecer dados telemétricos e a responder comandos externos para melhorar seu funcionamento. Embora qualquer coisa possa ser tornar um dispositivo IoT, a figura a seguir mostra um modelo de três pilares usado pela Microsoft, para avaliar o valor de uma solução IoT (Ferreira, Manzan e Duraes 2022).

Figura 1.12 – Pilares do modelo de valor de uma solução IoT



Fonte: https://pt.everand.com/book/601064480/Arquitetura-de-solucoes-IoT-Desenvolva-com-Internet-das-Coisas-para-o-mundo-real, 2022

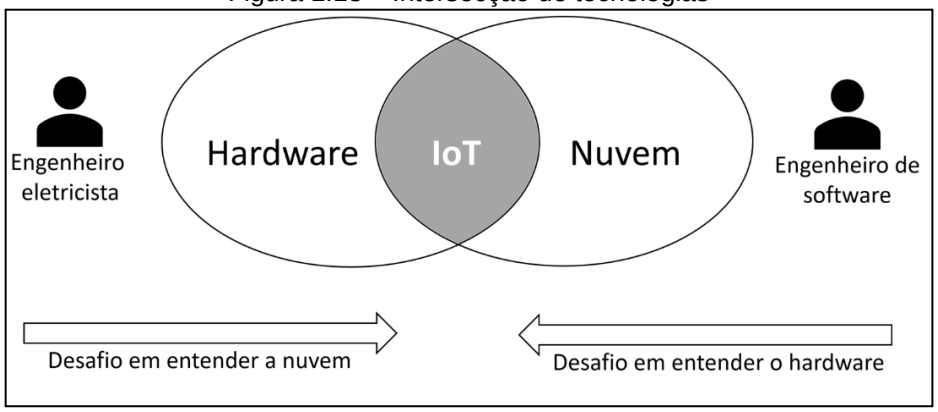
Alguns dos fatores que favorecem o crescimento do mercado de IoT são:

- A redução no custo de produção de sensores;
- A redução no custo de coleta e armazenamento de dados devido às soluções na nuvem;
- A expansão do acesso à internet e o aumento do poder computacional. (Khvoynitskaya, 2021, apud Ferreira, Manzan e Duraes 2022).

“A expectativa é de que existam mais de 1 trilhão de dispositivos IoT em funcionamento até 2030” (LEA, 2020, apud Ferreira, Manzan e Duraes 2022, p.5)

Para o desenvolvimento de uma solução IoT é necessário a intersecção de tecnologias de hardware e software. A figura 1.13 a seguir ilustra isso.

Figura 1.13 – Intersecção de tecnologias

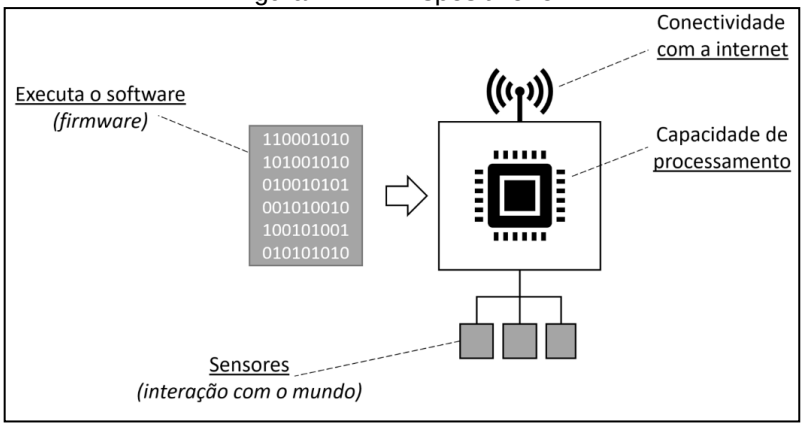


Fonte: https://pt.everand.com/book/601064480/Arquitetura-de-solucoes-IoT-Desenvolva-com-Internet-das-Coisas-para-o-mundo-real, 2022

Para um objeto ser considerado IoT ele precisa:

- Estar conectado direta ou indiretamente à internet em algum momento;
- Ter ao menos um microcontrolador (MCU) para executar um firmware (programação) localmente;
- Interagir com o mundo através de sensores e/ou atuadores. A Figura 1.14 ilustra esse conceito.

Figura 1.14 – Dispositivo IoT



Fonte: https://pt.everand.com/book/601064480/Arquitetura-de-solucoes-IoT-Desenvolva-com-Internet-das-Coisas-para-o-mundo-real, 2022

1.6 Microprocessadores versus microcontroladores

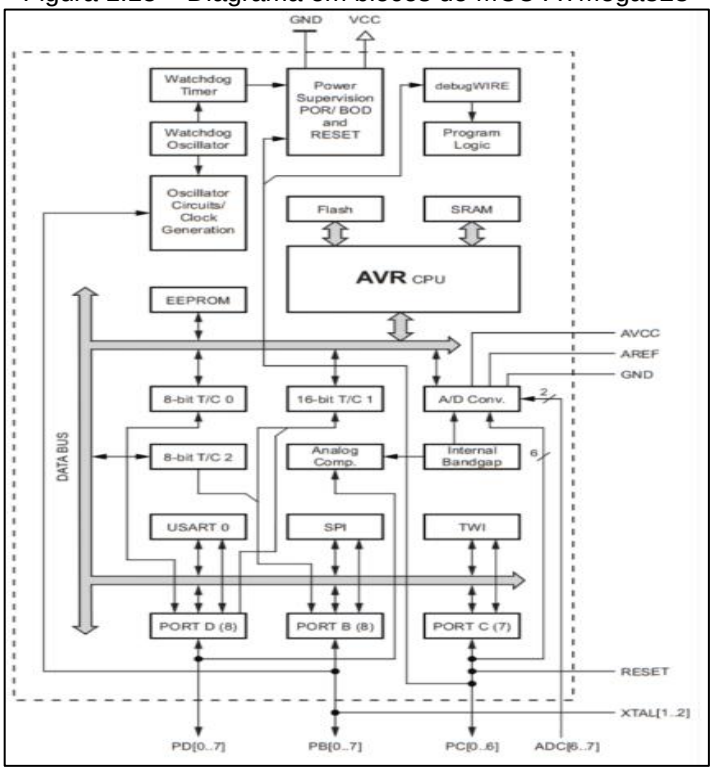
MPU (microprocessor Unit, em português, unidade microprocessadora), muito conhecidos nos computadores pessoais, eles requerem peças externas, memória RAM, HD para armazenar grande quantidade de dados, monitor, placa de Som, periféricos e etc. Os Single – Board computer, em português, computador de placa única baseadas em MPU, como o Raspberry Pi e Latte Panda, executam Linux e Windows nas versões embedded (em português, embarcado ou integrado). MPU pode fazer parte de dispositivos IoT e possuem alta capacidade de processamento e memória (Ferreira, Manzan e Duraes 2022).

Uma das vantagens de usar Microprocessador é porque ele possui portas de I/O que podem ser conectadas diretamente a sensores e atuadores. Possuem também um baixo custo e baixo consumo de energia, comparado ao computador e notebook. Devido ao seu tamanho reduzido, é possível integrar esses dispositivos ao equipamento que eles controlarão (Ferreira, Manzan e Duraes, 2022).

O MCU (Microcontroller Unit, em português Unidade Microcontroladora) possui memória, processamento, temporizadores e interfaces, todos dentro de um mesmo componente. MCU não utilizam sistemas operacionais como Windows ou Linux por exemplo. Eles usam sistemas operacionais compactos, como “RTOS”, ou só o código de aplicação bare – metal (Ferreira, Manzan e Duraes, 2022).

“A figura a seguir, extraída do manual do microcontrolador ATMega328 (utilizado no Arduino Uno – https://www.arduino.cc/en/Main/arduinoBoardUno), apresenta o diagrama em blocos de todos os componentes existentes internamente.” (Ferreira, Manzan e Duraes 2022, p.71, grifo do autor).

Figura 1.15 – Diagrama em blocos do MCU ATMega328



Fonte: Livro Arquitetura de Soluções IoT: Desenvolva com internet das coisas, 2022, p.71

Todos os componentes essenciais estão inclusos no MCU, como gerenciamento de energia, timer, memória (SRAM, EEPROM, Flash), CPU, circuitos de comunicação com vários protocolos (USART, SPI, TWI) e o GPIO (Port D, B e C). (Ferreira, Manzan e Duraes, 2022).

Algumas vantagens de usar microcontrolador são devido ao seu de baixo custo, pois existem centenas de configurações no mercado, permitindo ao desenvolvedor selecionar o MCU que possua a capacidade de processamento e memória para a solução. (Ferreira, Manzan e Duraes, 2022).

Microcontroladores consomem pouca energia devido à baixa velocidade de processamento e capacidade de memória (comparado ao MPU, resulta em menor consumo de energia, o que possibilita alimentação por baterias e energia solar). Quase todos os microcontroladores possuem modos de economia de energia, em que partes do seu circuito podem ser ligadas e desligadas, conforme a necessidade. (Ferreira, Manzan e Duraes, 2022).

Dispositivos com microcontroladores podem ser instalados em ambientes hostis, com grandes variações de temperaturas, vibração, ruído eletromagnético, poeira e etc. Nesses cenários um MPU com componentes externos possui mais pontos de falha. Já o MCU como possui os periféricos internamente, sofre menos falhas. (Ferreira, Manzan e Duraes, 2022).

1.7 Sistemas embarcados para IoT

Geralmente, os dispositivos IoT baseados em MCU's não possuem um sistema operacional internamente para controlar o hardware: interrupções, processador, alocação e liberação de memória, Inputs/Outputs, Timers, velocidade de clock do CPU e etc. Então são utilizadas as seguintes opções de programação para controlar dispositivos IoT baseados em MCU (Ferreira, Manzan e Duraes, 2022):

● Bare-Metal: O código da aplicação controla todos os aspectos do dispositivo. Normalmente usa-se a linguagem C e/ou C++, devido ao desempenho, versatilidade, portabilidade e disponibilidade de bibliotecas para diversos MCU's e periféricos. Aplicações em bare – metal controlam cada aspecto do hardware, assim os desenvolvedores tem controle total sobre cada ciclo de clock do CPU e de cada bit da memória. (Ferreira, Manzan e Duraes, 2022).

É usado quando há limitações extremas de hardware, como no desenvolvimento de marca passo, por exemplo, estações meteorológicas movidas por energia solar, devido ao acesso restrito a energia. É usado em produção de alto volume de dispositivos, cujo custo por unidade impacta o resultado e o preço no mercado, como brinquedos eletrônicos e etc. (Ferreira, Manzan e Duraes, 2022).

Usando bare-metal a solução fica enxuta, mas aumenta a complexidade para escrever, testar e manter a aplicação. Técnicas como pooling (onde o código é executado em loop infinito) ou maquinas de estado, são normalmente usadas em conjunto com interrupções de hardware para o controle de aplicações em bare metal. (Ferreira, Manzan e Duraes 2022).

● RTOS: Usa-se um sistema operacional em tempo real, que normalmente é feito em C e,ou C++. É mais simples que o bare-metal, ele é vantajoso para ser usado em cenários complexos, como em equipamentos: industriais, médicos, de suporte a vida, em automóveis, entre outros (Ferreira, Manzan e Duraes 2022).

● Interpretadores: É um modelo mais recente de desenvolvimento baseados em MCU's, ele é um meio termo entre o desenvolvimento em bare-metal e o uso de um RTOS. Uma das vantagens é a possibilidade de usar código de alto nível como o Python e C# (Ferreira, Manzan e Duraes, 2022).

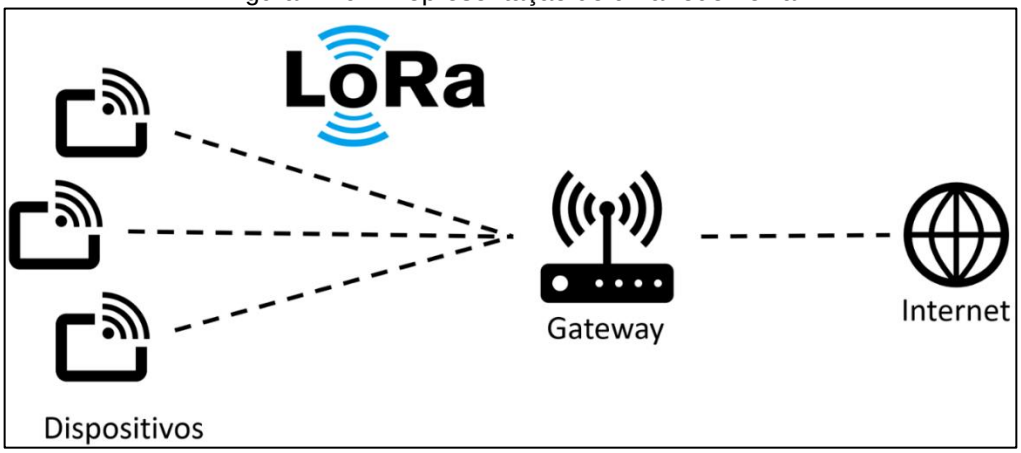
1.8 Rede LoRa

Rede LoRa (Long Range, em português, longo alcance) é uma rede ideal para IoT, graças à sua combinação única de características: longo alcance sem fio e baixo consumo de energia. LoRa opera em região de frequências não regulamentada, portanto é de uso gratuito. A frequência varia em todo o mundo, mas no Brasil é de 915 MHz. (Ferreira, Manzan e Duraes, 2022).

Dispositivos conectados a uma rede LoRa podem comunicar-se com múltiplos receptores a distâncias de até 10 Km. O uso de energia necessário para o tráfego de pacotes LoRa é baixo, próximo a 25 miliwatts (Ferreira, Manzan e Duraes 2022, apud VOITOVYCH, 2019). Dispositivos podem ter seu consumo de energia otimizado, utilizando a mesma bateria por anos, sem a necessidade de recarga. (Ferreira, Manzan e Duraes, 2022).

Redes LoRa trasferem dados em uma frequência de 0,3 Kb/s à 50 Kb/s (Ferreira, Manzan e Duraes 2022, apud LoRa ALLIANCE, 2022). É uma velocidade baixa, comparada a outras redes. O objetivo da rede LoRa é suprir cenários de comunicação com localizações remotas ou de difícil acesso, onde pequenos pacotes de dados podem ser usados, para ativar sistemas, monitorar, emitir atualizações periódicas de estados, ou alarmar situações críticas (Ferreira, Manzan e Duraes, 2022).

Figura 1.16 – Representação de uma rede LoRa



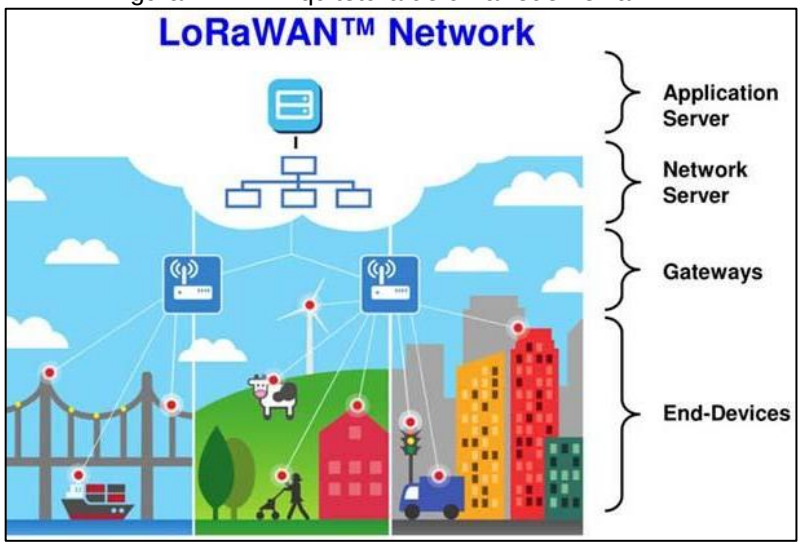
Fonte: Livro Arquitetura de Soluções IoT: Desenvolva com internet das coisas, 2022, p.109

LoRaWAN (Long Range Wide-Area Network), é um padrão de comunicação aberto, mantido pela LoRa ALLIANCE, uma associação sem fins lucrativos, que visa aperfeiçoar e manter o protocolo LoRaWAN (Ferreira, Manzan e Duraes, 2022).

Dispositivos precisam autenticar-se antes de se conectarem à rede. Isso tornam as redes LoRa mais seguras ao não permitir que qualquer dispositivo acesse e intercepte pacotes sendo trafegados (Ferreira, Manzan e Duraes, 2022).

Para construir uma rede LoRa é necessário que exista ao menos um dispositivo cliente e um gateway. O dispositivo cliente emite ou recebe pacotes LoRa do gateway, enquanto o gateway é responsável por manter a conexão com os dispositivos clientes e conectar-se a internet. A Figura 1.17 ilustra exemplos de usos de uma rede LoRaWAN, em diferentes áreas do cotidiano (Ferreira, Manzan e Duraes, 2022).

Figura 1.17 – Arquitetura de uma rede LoRaWAN



Fonte: Projetos com ESP32 e LoRa, 2019, p.20

Muitas soluções IoT não necessitam estar conectadas à internet, mas sim à um servidor local, dentro da rede, esse é o cenário local. Quando é conectada a internet e a nuvem, chama-se cenário conectado.

1.9 MQTT

O MQTT (Message Queuing Telemetry Transport) é um protocolo de comunicação leve, projetado especificamente para dispositivos de baixo consumo e redes com largura de banda limitada, características comuns em aplicações de Internet das Coisas (IoT). Criado pela IBM nos anos 1990, o MQTT foi desenvolvido para ser simples e eficiente, permitindo a troca de mensagens entre dispositivos de maneira rápida e confiável. Este protocolo utiliza um modelo de publicação/assinatura (publish/subscribe), onde os dispositivos podem publicar dados em tópicos específicos e/ou subscrever-se a esses tópicos para receber atualizações, facilitando a comunicação escalável entre muitos dispositivos (YUAN, 2023).

A arquitetura do MQTT é composta por três componentes principais: o cliente (que pode ser um publicador ou um assinante), o broker (servidor que gerencia a distribuição das mensagens) e os tópicos (canais de comunicação). Os clientes enviam mensagens para o broker, que as distribui aos assinantes interessados nos tópicos correspondentes. Essa abordagem reduz significativamente o tráfego de rede, uma vez que cada mensagem é enviada apenas uma vez ao broker, que então a redistribui, em vez de cada dispositivo enviar a mensagem a todos os outros dispositivos diretamente. Essa eficiência é crucial em redes IoT, onde a economia de energia e largura de banda é fundamental (Hivemq, 2023).

A popularidade do MQTT tem crescido significativamente devido à sua simplicidade, eficiência e robustez. Ele é amplamente utilizado em diversas aplicações, desde casas inteligentes e monitoramento ambiental até automação industrial e veículos conectados. O protocolo suporta níveis de Qualidade de Serviço (QoS) que garantem a entrega das mensagens, mesmo em condições de rede instáveis, e fornece mecanismos de segurança, como autenticação e criptografia, para proteger os dados transmitidos. Em suma, o MQTT é uma solução versátil e poderosa

para facilitar a comunicação em ambientes IoT, permitindo a construção de sistemas distribuídos e escaláveis com uma infraestrutura de comunicação eficiente e confiável (Eclipse Foundation, 2023).

1.10 Node-RED

Node-RED é uma plataforma de desenvolvimento baseada em fluxo que foi criada pela IBM para facilitar a conexão e a automação de dispositivos, APIs e serviços online. A principal característica do Node-RED é sua interface gráfica intuitiva, que permite aos usuários construir fluxos de dados através de uma abordagem visual, conectando nós que representam diferentes funcionalidades. Cada nó pode executar uma tarefa específica, como ler dados de um sensor, processar informações, ou enviar comandos para um atuador. Esta abordagem modular e visual reduz a complexidade da programação tradicional, tornando o desenvolvimento mais acessível, especialmente para aqueles que podem não ter uma forte base em programação.

O Node-RED é executado sobre o Node.js, um ambiente de execução para JavaScript, o que garante alto desempenho e escalabilidade. A plataforma é extensível, permitindo que os desenvolvedores criem seus próprios nós personalizados ou instalem nós adicionais da vasta biblioteca disponível na comunidade. Isso significa que, independentemente da aplicação, é provável que existam nós pré-existentes que possam ser utilizados para integrar dispositivos ou serviços específicos. A flexibilidade e a extensibilidade do Node-RED fazem dele uma escolha popular para uma ampla variedade de projetos, desde automação residencial até sistemas industriais complexos.

Uma das grandes vantagens do Node-RED é a sua capacidade de se integrar facilmente com protocolos de comunicação populares no mundo IoT, como MQTT (Message Queuing Telemetry Transport). MQTT é um protocolo leve e eficiente para a troca de mensagens entre dispositivos, e a integração com Node-RED é direta. Usando blocos MQTT no Node-RED, é possível conectar-se a um broker MQTT para publicar e subscrever mensagens em tópicos específicos, facilitando a comunicação

entre diferentes dispositivos e sistemas. Isso é especialmente útil em projetos de IoT onde a confiabilidade e a eficiência da comunicação são cruciais.

Além das capacidades técnicas, Node-RED possui uma comunidade ativa e vibrante que contribui para seu desenvolvimento contínuo e fornece suporte aos novos usuários. Há uma abundância de recursos disponíveis, incluindo tutoriais, fóruns, e exemplos de projetos, que ajudam os usuários a aprender e tirar o máximo proveito da plataforma. A documentação oficial do Node-RED é detalhada e abrangente, cobrindo desde os conceitos básicos até tópicos avançados, o que facilita a curva de aprendizado. Com uma combinação de poder, flexibilidade e suporte comunitário, Node-RED se estabeleceu como uma ferramenta indispensável para desenvolvedores que trabalham na interseção de hardware, software e serviços online (Node-RED, 2023).

2 METODOLOGIA

Neste capítulo encontra-se a trajetória para o desenvolvimento e construção do projeto intitulado IoT em Motores Industriais, trata-se de uma pesquisa aplicada que é desenvolvida nas dependências da FATEC São Bernardo do Campo e nas residências dos integrantes do grupo.

2.1 O que é metodologia?

Prodanov e Freitas (2013) destacam que a metodologia consiste em estudar, compreender e avaliar os métodos disponíveis para a realização da construção do protótipo. Enfatiza que os métodos são procedimentos amplos do raciocínio, enquanto as técnicas são procedimentos mais restritas que operacionalizam os métodos mediante emprego de instrumentos adequados.

A redação do trabalho tem como base as regras e as normas contidas no Manual de Normalização de Projeto de Trabalho de Graduação da FATEC-SBCampo (2017) que se encontra amparada nas normas da ABNT.

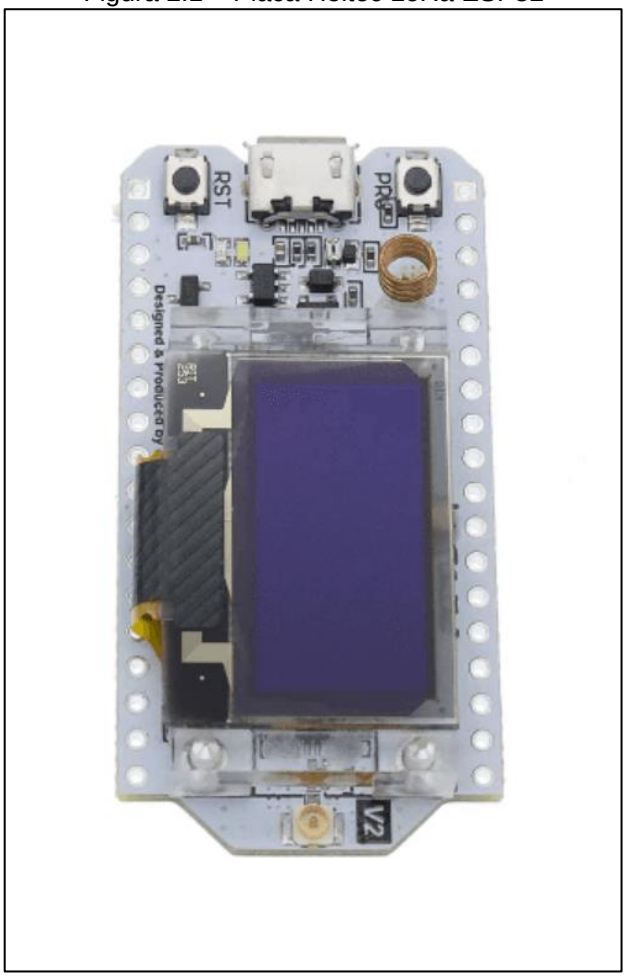
2.2 O tema-problema, justificativa e diagrama de funcionamento

O tema-problema surgiu através de pesquisas, conhecimento adquiridos em aulas e experiência profissional dos integrantes do grupo. Na Fatec observou-se a necessidade do monitoramento das motobombas que abastecem a caixa de água da instituição, devido a isso nosso projeto destina-se ao monitoramento de motores elétricos de modo geral, utilizando a comunicação via radiofrequência.

Para realizar esse monitoramento e comunicação via radiofrequência optamos por utilizar a Placa Heltec LoRa ESP32, que é baseada no chip SX1276, com longo alcance. Com tecnologia de comunicação sem fio aplicada em redes LPWAM (Low Power Wide Area Network). LoRa que significa “Long Range” (Longo Alcance), em conjunto com a tecnologia ESP32 criou uma plataforma completa com Wifi, Bluetooth e LoRa, além do Hardware e Software. O LoRa e o ESP32 criaram uma ótima

plataforma para desenvolvimento de projetos eletrônicos voltados para esse segmento, pois foi direcionada ao mundo IoT (Internet da Coisas). A figura 2.1 mostra a placa utilizada nesse projeto.

Figura 2.1 – Placa Heltec LoRa ESP32

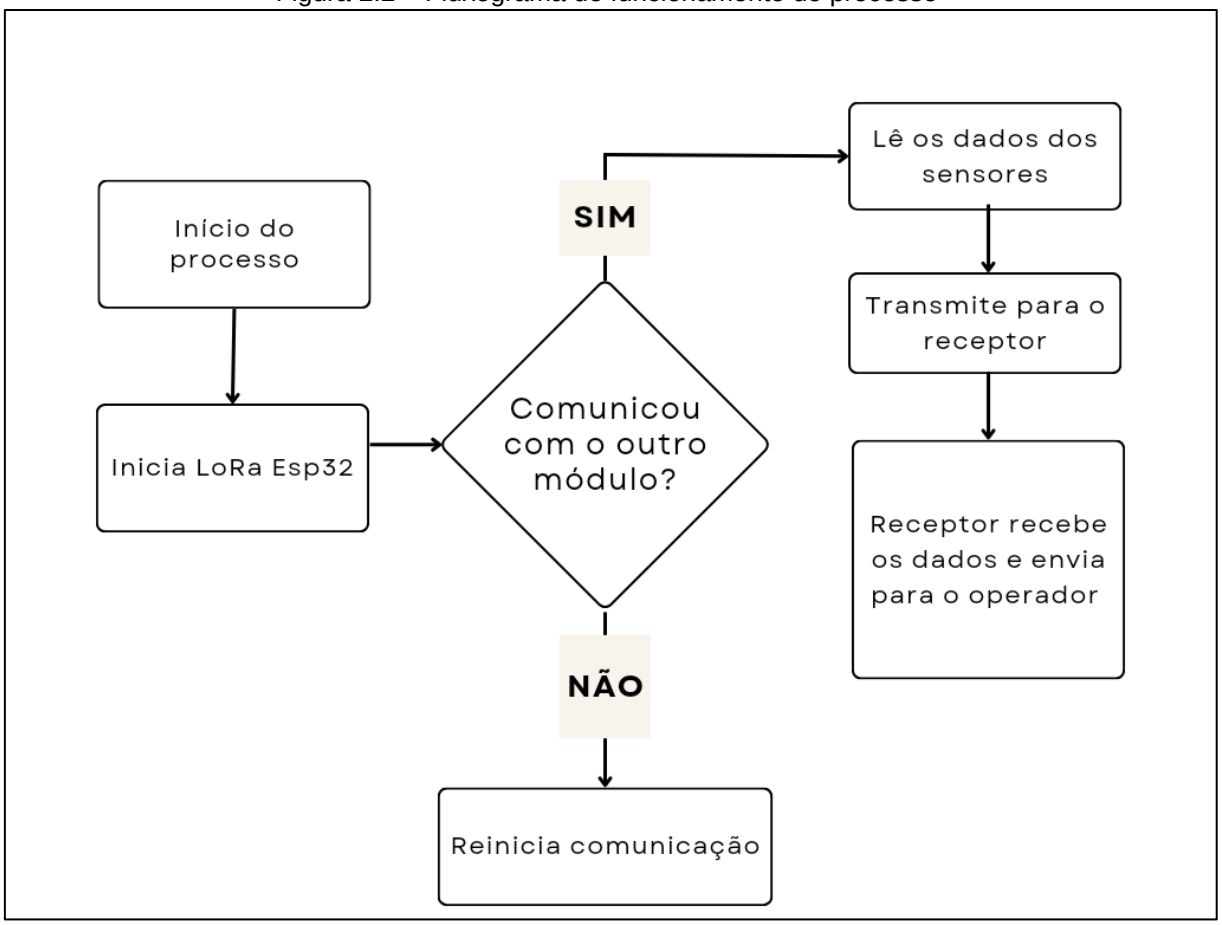


Fonte: https://www.institutodigital.com.br/produto/placa-lora-wifi-esp32-868-915mhz-com-oled-096/, 2024

A Placa Heltec LoRa ESP32 com OLED possui um Display OLED de 0,96 polegadas de 128×64 pixels assim, transmite em tempo real as informações obtidas sem a necessidade de ferramentas externas para visualizá-las. Contém a tecnologia Bluetooth e Wifi, mas com um alcance maior que o normal e baixo consumo de energia. É possível ter um alcance de até 15km em áreas rurais e 4km em áreas urbanas com comunicação sem fio, por causa de uma antena que o acompanha. A frequência de operação desta placa é de 868 ou 915 MHz. É possível conectar módulos e sensores na placa para obter dados ou executar ações através do módulo de comunicação LoRa. É possível acessar esses dados e informações por Bluetooth ou Wifi.

Para o desenvolvimento do projeto é construído um fluxograma de funcionamento do processo, conforme ilustra Figura 2.2.

Figura 2.2 – Fluxograma de funcionamento do processo



Fonte: Autoria própria, 2024

Através desse fluxograma foi realizado os primeiros insights para a forma como seria utilizado o microcontrolador Heltec LoRa ESP32, coletando os dados dos sensores e transmitindo via radiofrequência para o receptor. Caso houver falha de comunicação, ambos os módulos (transmissor e receptor) enviam sinal um para o outro a fim de restabelecer a comunicação. A placa Heltec LoRa ESP32 868/915Mhz é muito versátil quanto à sua programação, podendo ser programada diretamente pela IDE Arduino em C/C++, suporta MicroPython e CircuitPython, LUA e outras linguagens de programação de alto nível. A Arduino IDE pode ser baixada gratuitamente no site oficial da Arduino. Não é necessário ter um conhecimento avançado em programação ou linguagem C para começar a programar a placa, devido a isso essa placa se tornou viável em nosso projeto graças à sua versatilidade e custo benefício.

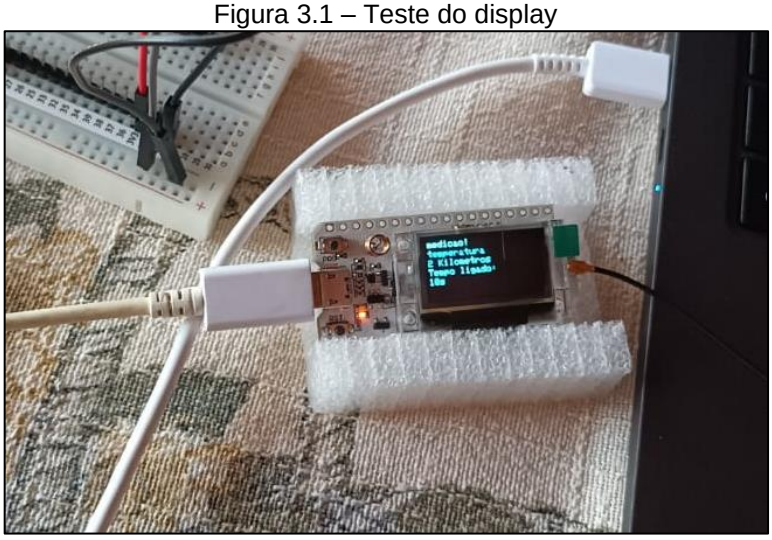
3 CONSTRUÇÃO DO DISPOSITIVO

Neste capitulo é mostrada a construção passo a passo do protótipo. A construção ou adaptação da estrutura mecânica, os dispositivos de acionamento, as placas eletrônicas de controle e a programação.

3.1 O protótipo – visão geral

Para começar os testes de comunicação entre os dois módulos Heltec LoRa ESP32, foi realizado algumas programações de teste, a fim de verificar a comunicação ponto a ponto via radiofrequência. A figura 3.1 demonstra alguns testes do display.

Figura 3.1 – Teste do display

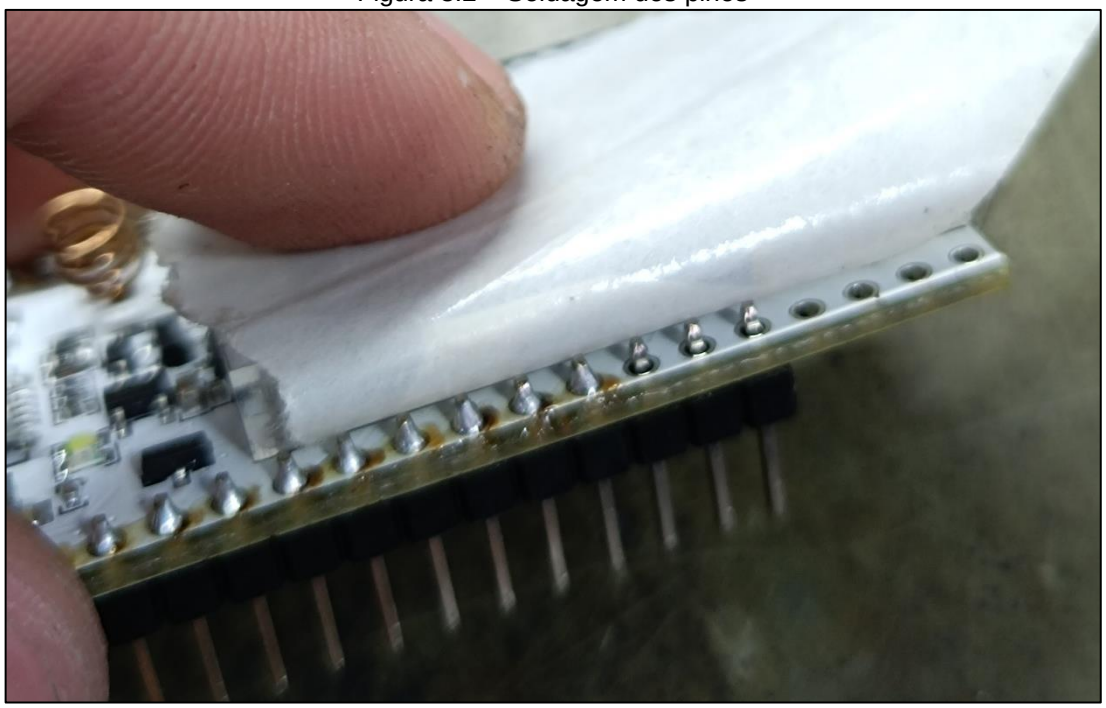


Fonte: Autoria própria, 2024

Como é possível observar, a placa Heltec LoRa ESP32 possui um display que possibilita visualizar uma informação desejada baseada na programação. Isso facilita na identificação dos dados que estão sendo transmitidos ou enviados.

Após os testes do display, foi necessário realizar a soldagem dos pinos de entradas e saídas da placa, evitando assim o mal contato durante os testes ou até mesmo falha na leitura dos sensores. A figura 3.2 demonstra o método utilizado para soldagem, colocando uma proteção no cabo Flat do display. Dessa forma foi possível realizar a soldagem sem danificar o cabo Flat do display.

Figura 3.2 – Soldagem dos pinos



Fonte: Autoria própria, 2024

Com os pinos soldados é possível realizar as conexões elétricas sem que haja interferência de mal contato. Dessa forma a placa Heltec LoRa ESP32 já está pronta para realizar a leitura de sensores conectados em suas entradas e transmitir para o outro módulo receptor. Foi conectado então o sensor de temperatura DHT22 e realizado o teste de envio dos dados entre transmissor e receptor, para avaliar a integridade dos dados. A figura 3.3 demonstra esse teste realizado.

Figura 3.3 – Teste envio de temperatura

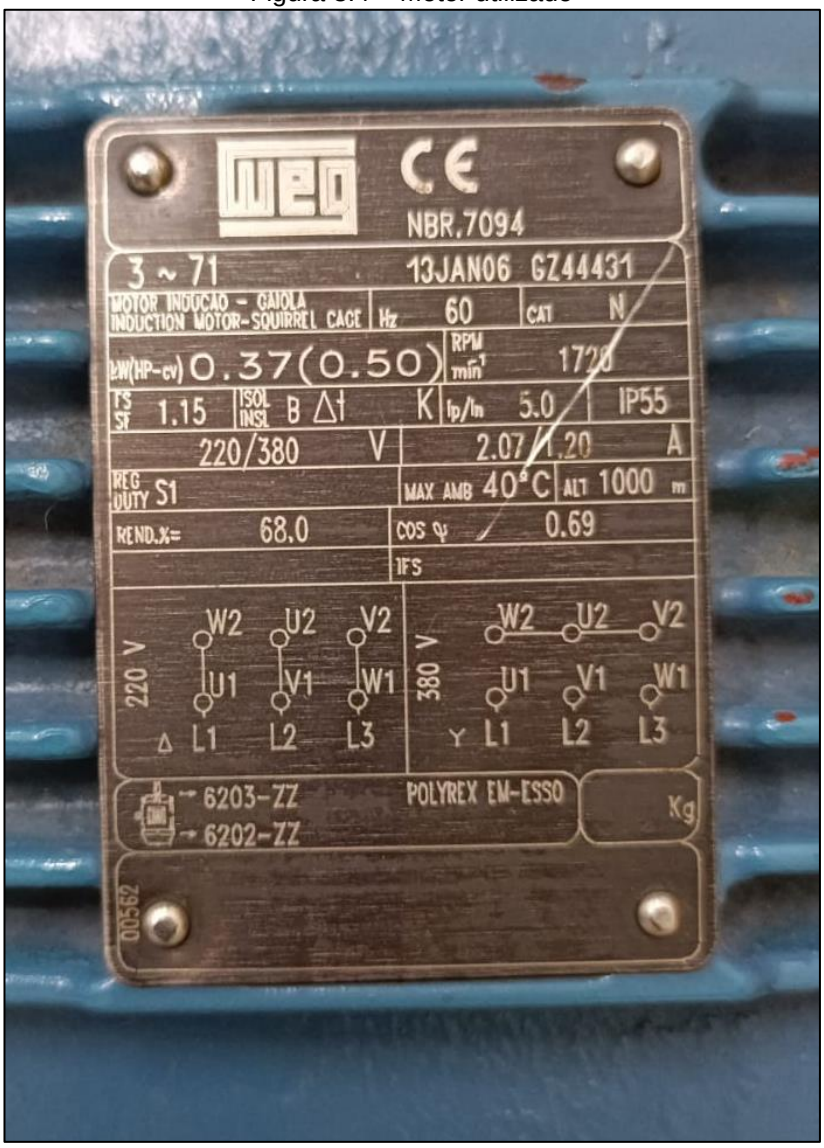
Fonte: Autoria própria, 2024

Na demonstração anterior é possível observar o módulo transmissor do lado esquerdo enviando o valor da temperatura lido pelo sensor e do lado direito, o módulo receptor recebendo esse dado e apresentando no display.

3.2 A parte eletrônica e elétrica

Na montagem do nosso projeto utilizamos alguns sensores e o motor trifásico da WEG de 0.5 CV, disponibilizado pela Fatec SBC. Na figura 3.4 mostra a plaqueta lateral do motor, informando todas as características elétricas.

Figura 3.4 – Motor utilizado

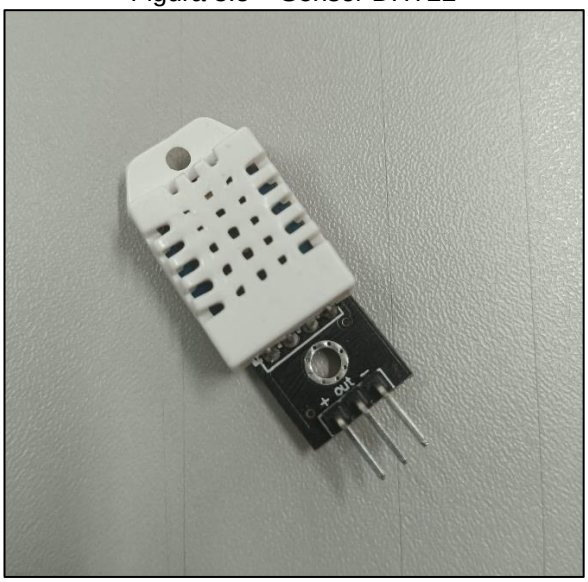


Fonte: Autoria própria, 2024

Além do motor da WEG, utilizamos três sensores principais, para o monitoramento da temperatura, corrente e vibração. Esses três sensores foram acoplados na carcaça do motor e conectados aos pinos de entradas do módulo transmissor Heltec LoRa ESP32.

O primeiro sensor que foi conectado e também utilizado para testes iniciais, foi o sensor de temperatura DHT22. O DHT22 é um sensor digital de temperatura e umidade amplamente utilizado em projetos de eletrônica e Internet das Coisas (IoT) devido à sua precisão e facilidade de uso. Ele pode medir temperaturas de -40 a 80 graus Celsius com uma precisão de ±0.5°C e umidade de 0 a 100% com uma precisão de 2-5%. O sensor possui um termistor e um sensor capacitivo de umidade, além de um microcontrolador para conversão de sinais analógicos para digitais. A figura 3.5 mostra este sensor que foi utilizado.

Figura 3.5 – Sensor DHT22



Fonte: Autoria própria, 2024

Outro sensor que foi utilizado para o monitoramento da corrente elétrica, foi o ZMCT103C. Ele é um sensor de corrente de alta precisão, amplamente utilizado para medir correntes AC (corrente alternada) em aplicações de monitoramento e controle de energia elétrica. Este sensor é baseado em um transformador de corrente, que permite a medição de correntes elevadas com segurança, convertendo-as em um sinal de baixa tensão proporcional, adequado para entrada em microcontroladores como Arduino e ESP32. O ZMCT103C é capaz de medir correntes de até 5A, com

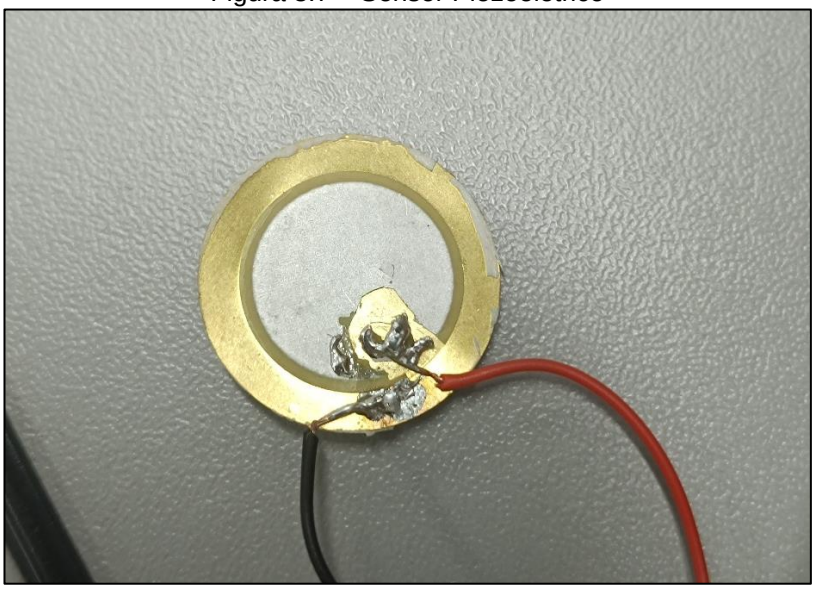
uma saída linear que facilita a leitura e processamento dos dados. A figura 3.6 mostra a estrutura física desse sensor.

Figura 3.6 – Sensor ZMCT103C

Fonte: Autoria própria, 2024

Por fim, nosso terceiro sensor utilizado foi o piezoelétrico. O sensor piezoelétrico é um dispositivo que aproveita a propriedade piezoelétrica de certos materiais, como cristais de quartzo ou cerâmicas piezoelétricas, para converter forças mecânicas, como pressão ou vibração, em sinais elétricos. Quando submetidos a deformações mecânicas, esses materiais geram uma tensão elétrica proporcional à magnitude da força aplicada. Além disso, os sensores piezoelétricos são valorizados por sua durabilidade e resistência a ambientes adversos, tornando-os adequados para aplicação em nosso projeto. Na figura 3.7 mostra o sensor piezoelétrico utilizado.

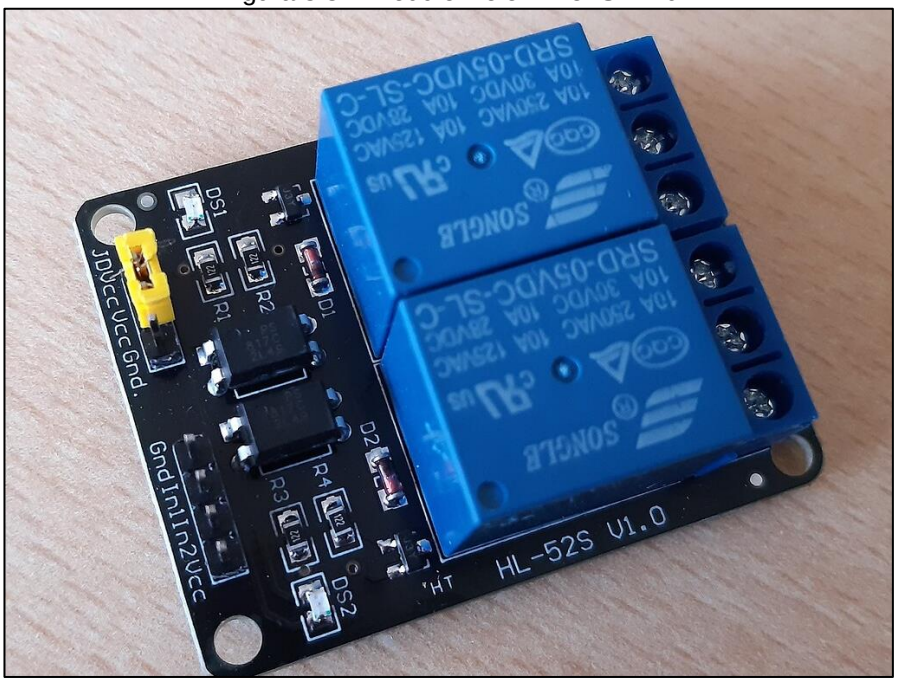
Figura 3.7 – Sensor Piezoelétrico



Fonte: Autoria própria, 2024

Além dos sensores já mencionados, utilizamos também o módulo relé HL-52S V1.0 para realizar o controle de desligamento do motor, caso haja alguma falha em potencial identificada pelos sensores. Esse módulo permite a comutação de cargas elétricas de alta potência por meio de sinais de baixa potência provenientes de microcontroladores, no caso, por meio do Heltec LoRa ESP32 que estamos utilizando. Este módulo geralmente opera em 5V e pode controlar dispositivos de até 10A a 250V AC ou 30V DC. Ele é equipado com um isolador óptico para proteger o circuito de controle de picos de tensão e interferências. Além disso, inclui LED's indicadores para facilitar o monitoramento do estado do relé. Na figura 3.8 mostra o módulo relé utilizado.

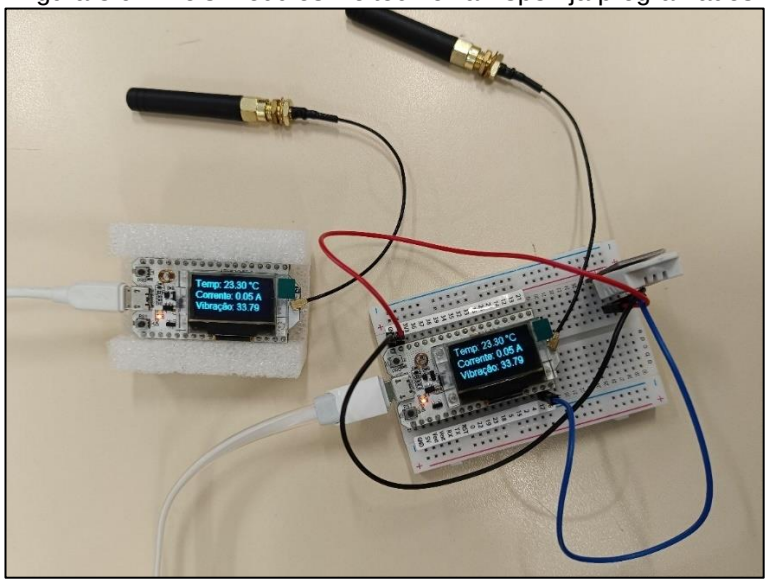
Figura 3.8 – Módulo Relé HL-52S V1.0



Fonte: Autoria própria, 2024

Com os sensores e dispositivos em mãos, partimos então para a implementação ao código do transmissor e receptor Heltec LoRa ESP32. Adicionando esses sensores na programação e conectando às entradas digitais, podemos observar no display do Heltec LoRa ESP32 que os dados dos sensores eram coletados e transmitidos através do protocolo LoRa ao módulo receptor em tempo real. Na figura 3.8 mostra os dois módulos já programados, mas sem todos os sensores conectados, para verificar a demonstração visual no display de ambos.

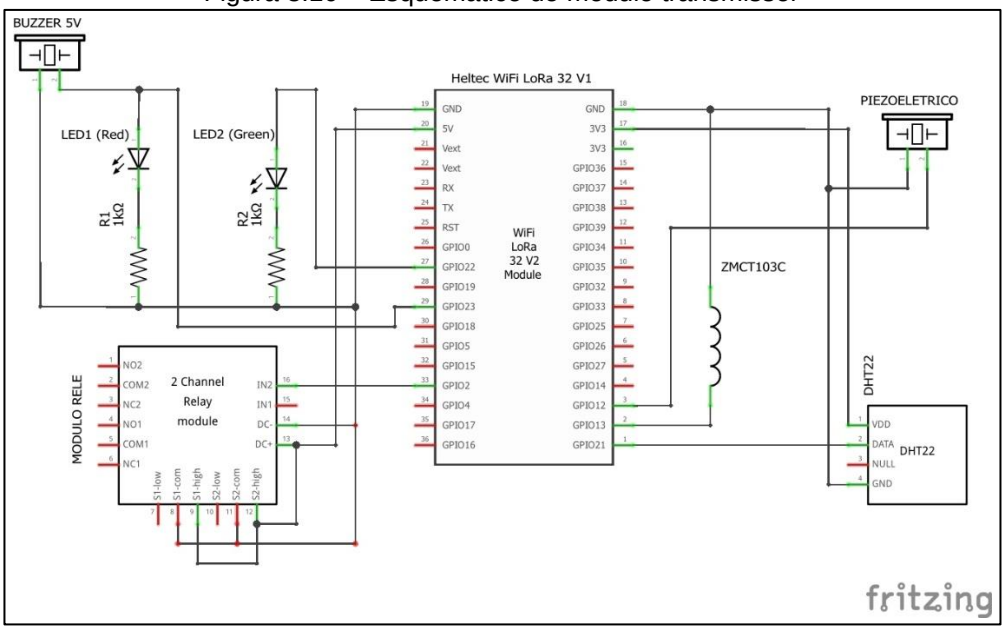
Figura 3.9 – Dois módulos Heltec LoRa Esp32 já programados



Fonte: Autoria própria, 2024

Após concluídos os testes de transmissão via radiofrequência, partimos para a montagem do circuito, seguindo os pinos de entradas e saídas já definidos em programação. Com essas informações, desenvolvemos o layout do nosso circuito no software Fritzing, que é uma plataforma de design de hardware eletrônico. Essa plataforma permite criar todo o circuito esquemático e diagrama das conexões. Na figura 3.10 mostra o circuito esquemático da parte de controle do módulo transmissor Heltec LoRa ESP32.

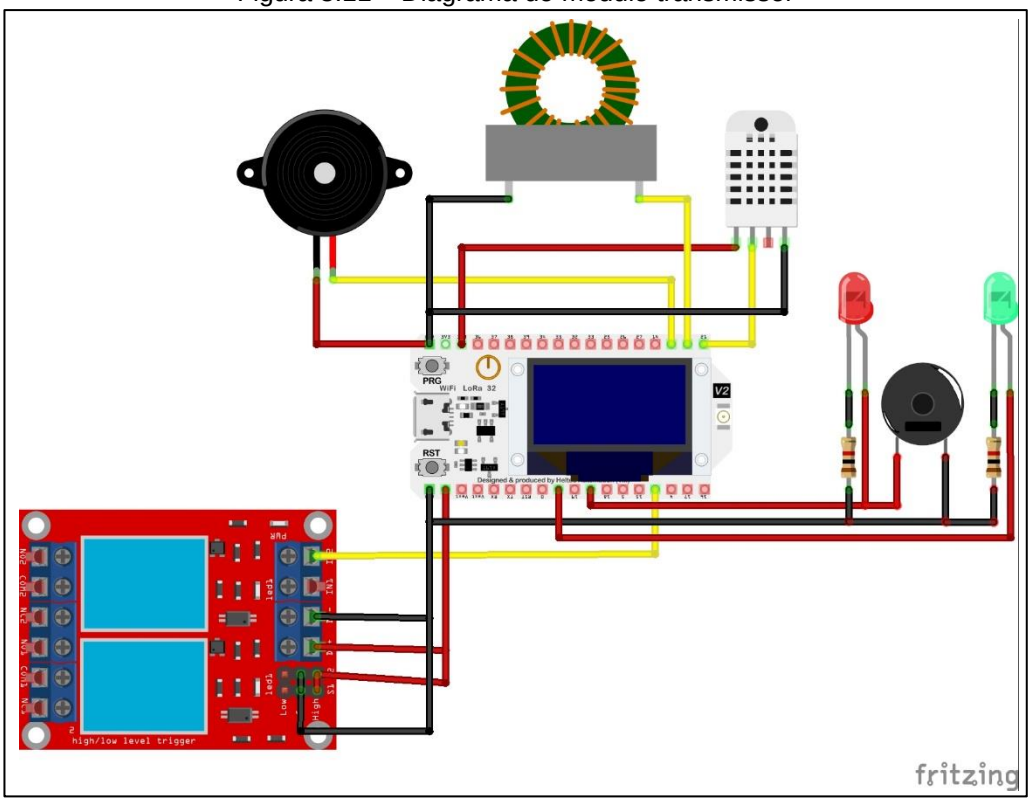
Figura 3.10 – Esquemático do módulo transmissor



Fonte: Autoria própria, 2024

Além do circuito esquemático, desenvolvemos também no mesmo software Fritzing o diagrama das conexões do módulo transmissor, para uma melhor compreensão. Importante ressaltar que somente o módulo transmissor possui essas conexões, pois é ele quem vai receber os dados dos sensores. Já o módulo receptor irá somente receber esses dados via radiofrequência, dessa forma não necessita da montagem de um circuito externo. Na figura 3.11 mostra o diagrama das conexões elétricas do módulo transmissor.

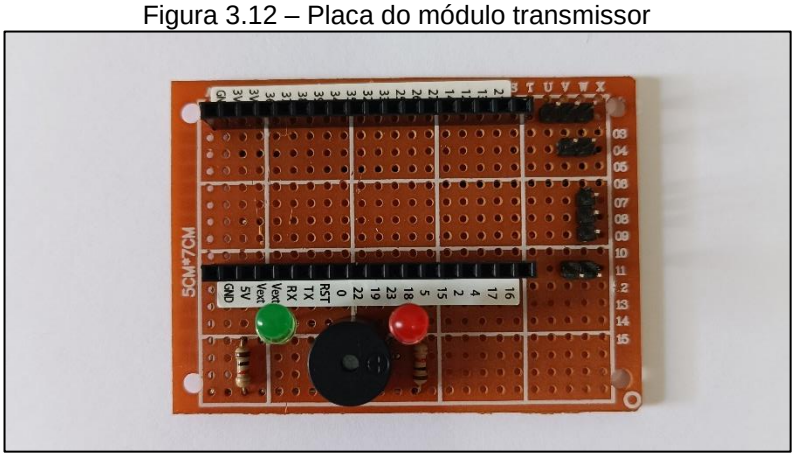
Figura 3.11 – Diagrama do módulo transmissor



Fonte: Autoria própria, 2024

Com a programação e conexões já elaboradas, realizamos a montagem do circuito em uma placa de fenolite perfurada, pois como ela vem perfurada e com as ilhas de cobre, não é necessário o uso de substâncias corrosivas para confecção da placa. Quanto aos componentes adicionamos somente barras de pino header fêmea, para conectar os módulos Heltec LoRa ESP32. Utilizamos também componentes como LED’s, resistores de 1 kΩ, buzzer 5V e mais pinos header macho para as conexões elétricas dos sensores e dispositivos externos. Na figura 3.12 mostra a placa do módulo transmissor já montada.

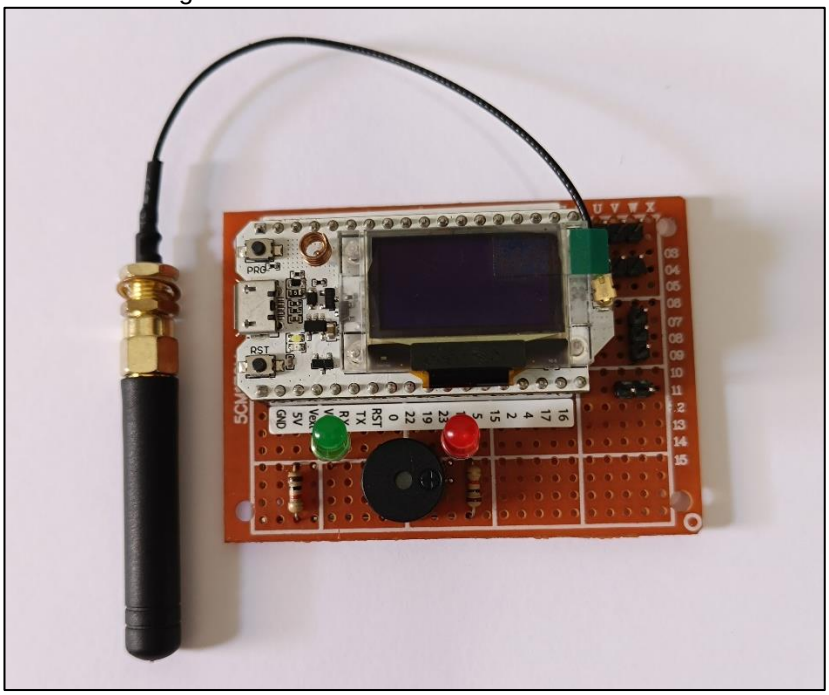
Figura 3.12 – Placa do módulo transmissor



Fonte: Autoria própria, 2024

Com a placa do módulo transmissor já finalizada, basta somente conectar o módulo Heltec LoRa Esp32 no soquete header fêmea da placa e posteriormente conectar os sensores nos pinos header macho ao lado direito. Na figura 3.13 mostra o módulo transmissor Heltec LoRa Esp32 já instalado nessa placa.

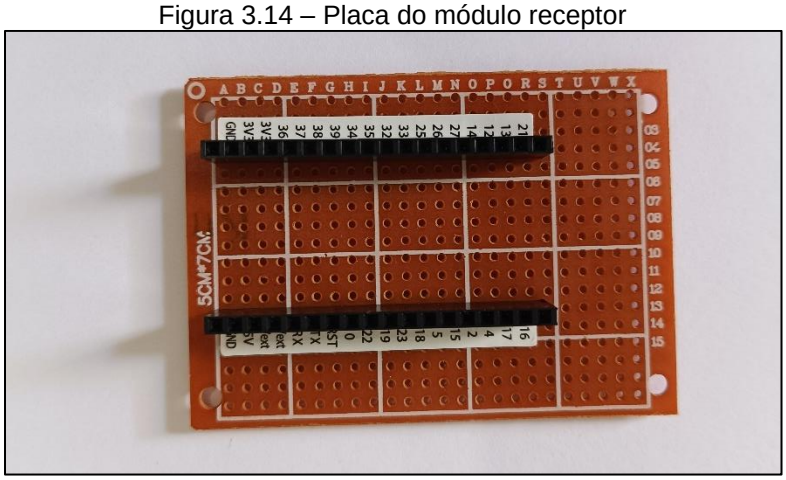
Figura 3.13 – Módulo transmissor finalizado



Fonte: Autoria própria, 2024

Em relação ao módulo receptor, fizemos somente o soquete da placa, tendo em vista que ele irá somente receber os dados via LoRa e transmitir ao broker MQTT na nuvem. Na figura 3.14 mostra a placa do módulo receptor.

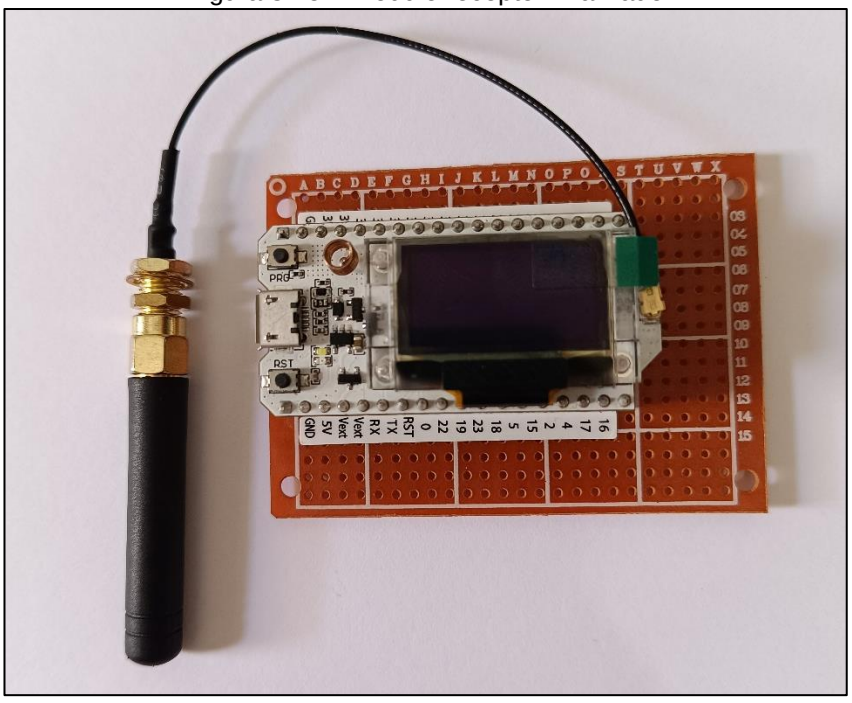
Figura 3.14 – Placa do módulo receptor



Fonte: Autoria própria, 2024

Com a placa do módulo receptor finalizada, conectamos então o módulo receptor Heltec LoRa Esp32 ao soquete da placa. Na figura 3.15 mostra a placa do módulo receptor finalizada.

Figura 3.15 – Módulo receptor finalizado

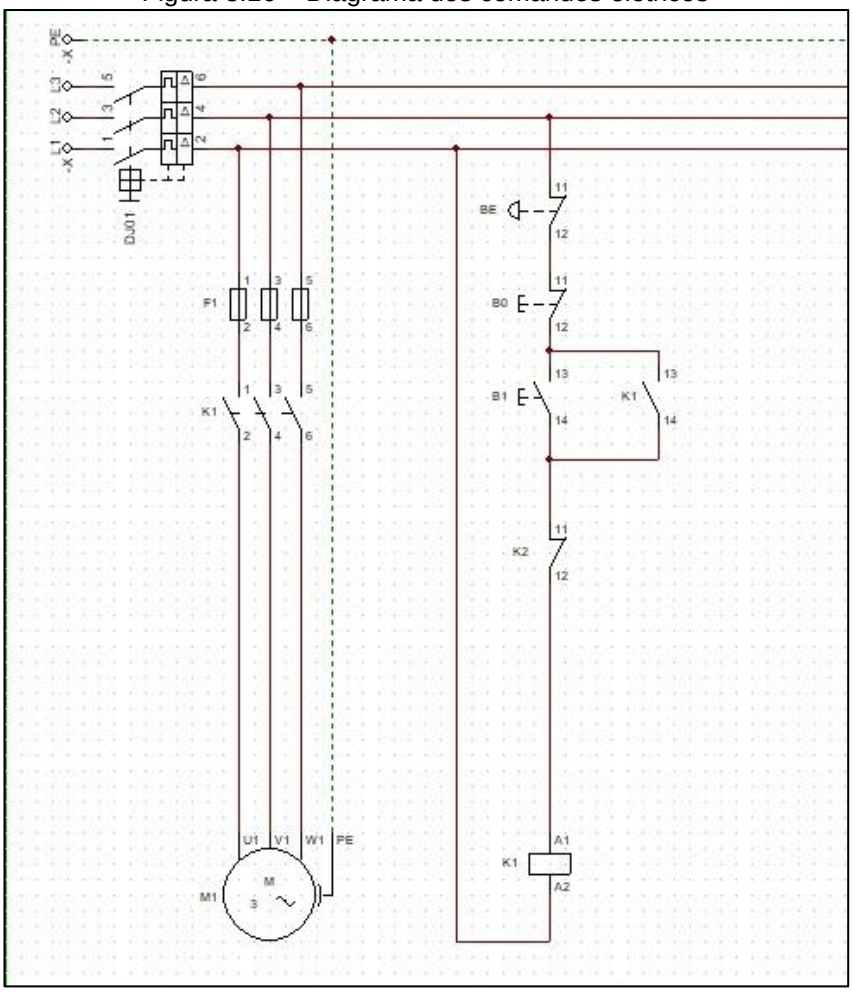


Fonte: Autoria própria, 2024

Com ambos os módulos já finalizados, fizemos a instalação dos sensores e módulo relé ao motor, a fim de dar continuidade aos nossos testes já em situação real de operação. Antes disso, fizemos um diagrama de comandos elétricos para realizar

a montagem dos acionamentos elétricos do motor e proteção dos circuitos. Na figura 3.16 mostra o diagrama dos comandos elétricos.

Figura 3.16 – Diagrama dos comandos elétricos

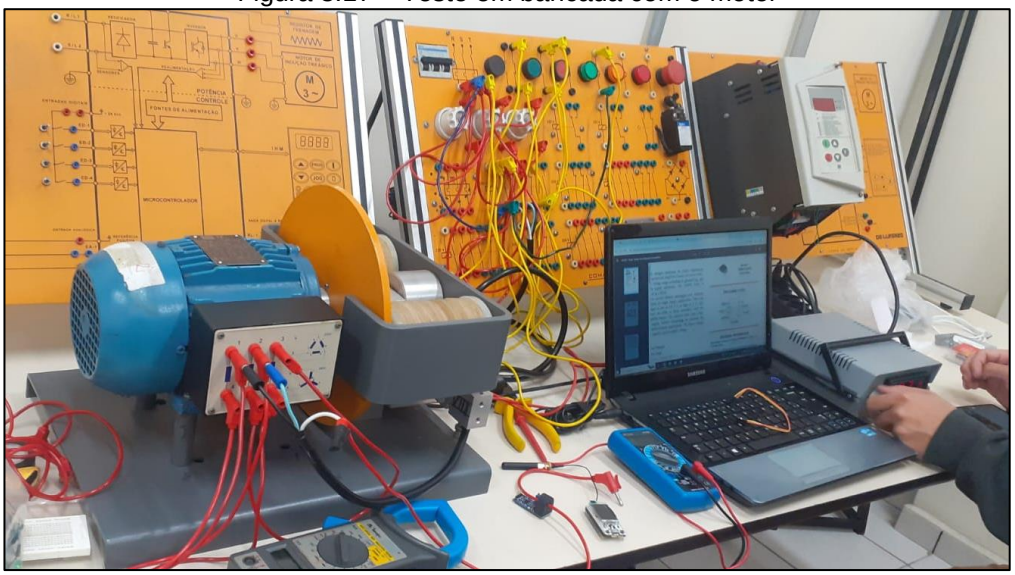


Fonte: Autoria própria, 2024

Com o diagrama já elaborado, realizamos a instalação dos sensores e módulo relé ao motor e executamos toda a montagem elétrica para o acionamento e controle do motor. Antes de energizar o sistema, analisamos todo o circuito e conexões elétricas de forma geral, a fim de evitar possíveis curtos ou falhas durante o processo de montagem. Realizar essa inspeção é um passo importante, pois dessa forma garantimos toda a segurança dos operadores e dos componentes que envolvem todo o sistema. Importante ressaltar que fizemos o circuito de partida direta do motor, onde o mesmo está sendo alimentado com 220V AC e o módulo relé realiza o desligamento do motor. Ligamos o módulo relé em série com a bobina do contator, para que dessa forma o relé quebre o selo da bobina do contator e consequentemente, o contator

desliga as fases do motor de forma segura, caso haja eventuais problemas de operação ou situações de falha com o motor. Na figura 3.17 mostra todo o circuito sendo montado e já em fase de operação.

Figura 3.17 – Teste em bancada com o motor



Fonte: Autoria própria, 2024

3.3 Linguagem de programação

A linguagem de programação utilizada no Heltec LoRa ESP32 para permitir a comunicação entre transmissor e receptor e comunicação MQTT é predominantemente C++, facilitada pela plataforma de desenvolvimento Arduino. A programação completa desse projeto, tanto do módulo receptor como do módulo transmissor, está no apêndice A e B dessa monografia. É importante ressaltar que é necessário baixar e instalar também todas as bibliotecas mencionadas no início da programação, para que o código funcione de forma adequada e sem erros. Os módulos Heltec LoRa ESP32 também necessitam de uma URL específica para ser reconhecida na IDE do Arduino. Essa URL também se encontra disponível no apêndice C dessa monografia.

Para a comunicação MQTT, a linguagem C++ continua sendo utilizada, aproveitando bibliotecas como a PubSubClient, que é amplamente adotada na comunidade Arduino. Essa biblioteca permite estabelecer conexões MQTT, publicar e subscrever tópicos, gerenciar a reconexão em caso de falhas e lidar com mensagens

recebidas de forma eficiente. A programação em C++ no Heltec LoRa ESP32 para MQTT envolve a configuração dos parâmetros de rede, como o endereço do broker MQTT, as credenciais de autenticação e a definição dos tópicos de interesse, possibilitando uma comunicação robusta e de baixo consumo de recursos.

A combinação de C++ e as bibliotecas fornecidas pelo Arduino torna o desenvolvimento para o Heltec LoRa ESP32 bastante eficiente, permitindo a criação de aplicações complexas com um código relativamente simples e de fácil manutenção. As bibliotecas abstraem muitos dos detalhes de baixo nível da comunicação e gerenciamento de hardware, permitindo que os desenvolvedores se concentrem na lógica da aplicação e na funcionalidade desejada. Isso é crucial para aplicações IoT, onde a integração rápida e eficaz de múltiplos dispositivos e protocolos de comunicação, como LoRa e MQTT, é essencial para o sucesso do projeto. A programação desenvolvida dos módulos transmissor e receptor se encontram no apêndice dessa monografia.

3.4 Broker MQTT e fluxo Node-RED

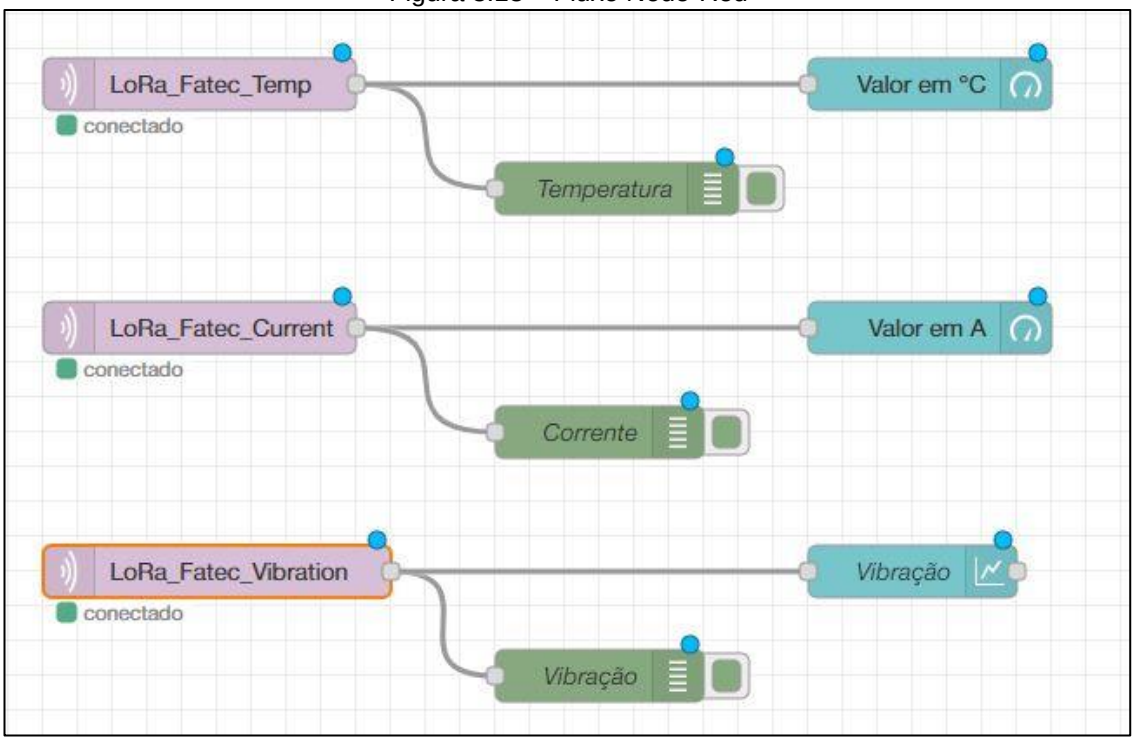
O módulo receptor é o responsável por publicar todos os dados dos sensores em um broker MQTT, para que esses dados sejam acessados e monitorados posteriormente. Optamos por utilizar um broker público, que é o HiveMQ, onde é possível publicar e se inscrever em diversos tópicos.

O módulo receptor publica em três tópicos diferentes os valores recebidos de temperatura, corrente e vibração. Dessa forma é possível separar corretamente os valores para que não haja uma mistura dos dados coletados.

Após a publicação desses dados no broker MQTT, iniciamos o processo de acesso desses dados e o monitoramento via Node-RED. Para isso, realizamos o download e instalação do Node-RED em um notebook. Com a ferramenta instalada, criamos um fluxo onde três blocos são responsáveis por se inscrever nos tópicos publicados via MQTT pelo receptor. Dessa maneira, agora o Node-RED já recebe os dados dos sensores publicados via MQTT. Criamos também mais três blocos de dashboard que são responsáveis por receber esses dados e criar gráficos e

medidores, a fim de apresentar de forma gráfica os valores dos sensores em tempo real. A figura 3.18 mostra como ficou nosso fluxo no Node-RED.

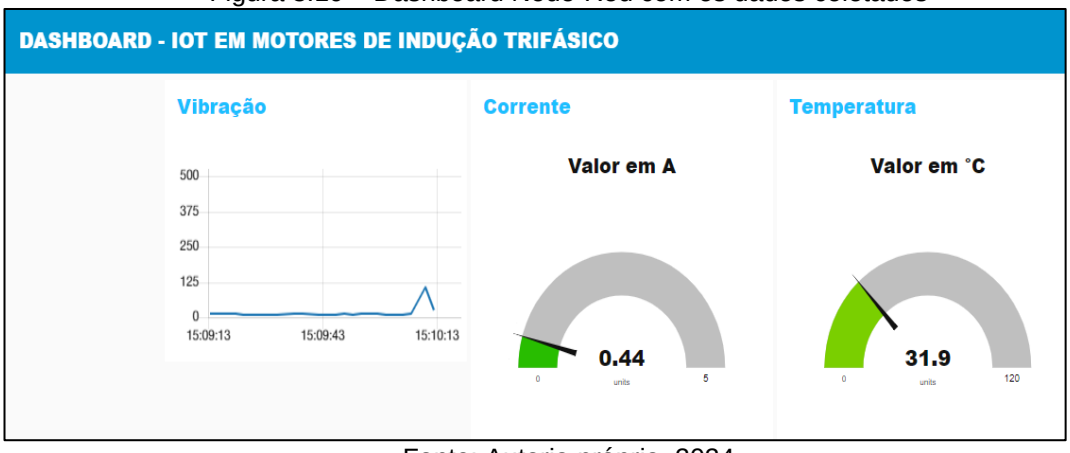
Figura 3.18 – Fluxo Node-Red



Fonte: Autoria própria, 2024

Com o fluxo pronto do Node-RED, a própria plataforma gera um Dashboard para a representação dos valores dos blocos em azul. Por fim, realizamos alguns ajustes de posicionamento e escalas dos gráficos gerados, para que se torne claro e de fácil visualização. Na figura 3.19 mostra o resultado final do Dashboard exibindo os dados em tempo real via MQTT.

Figura 3.19 – Dashboard Node-Red com os dados coletados



Fonte: Autoria própria, 2024

CONSIDERAÇÕES FINAIS

O trabalho concluiu que a implementação de um sistema de monitoramento de motores elétricos utilizando sensores e o protocolo LoRa, integrado com o microcontrolador Heltec LoRa ESP32, é uma solução eficaz e de baixo custo. A comunicação de longa distância e o baixo consumo de energia do protocolo LoRa, aliados à capacidade de processamento e conectividade do ESP32, permitem a coleta e transmissão contínua de dados críticos sobre o estado dos motores, como vibração, temperatura e corrente. Essa infraestrutura tecnológica possibilita a detecção precoce de anomalias e a realização de manutenção preventiva, prolongando a vida útil dos equipamentos e reduzindo custos operacionais.

A integração do protocolo MQTT para a transmissão de dados para um servidor central e a utilização do Node-RED para visualização e análise demonstraram ser escolhas acertadas. O MQTT, com sua eficiência e baixa latência, garantiu uma comunicação confiável e rápida, enquanto o Node-RED proporcionou uma interface intuitiva e poderosa para monitoramento em tempo real e análise histórica. Esse setup permite que operadores e técnicos de manutenção tenham acesso imediato a informações vitais, facilitando a tomada de decisões rápidas e informadas para evitar falhas e otimizar o desempenho dos motores.

Em suma, o projeto provou que tecnologias de IoT podem transformar o monitoramento industrial, oferecendo soluções escaláveis e acessíveis. O sistema desenvolvido não apenas atende às necessidades atuais de monitoramento, mas também estabelece uma base sólida para futuras expansões e melhorias, como a inclusão de algoritmos de inteligência artificial para análise preditiva e a integração com sistemas mais amplos de gerenciamento de manutenção. A combinação de sensores precisos, comunicação eficiente e uma plataforma de visualização robusta posiciona este sistema como uma ferramenta essencial para a modernização e melhoria contínua da manutenção industrial.

REFERÊNCIAS

ABECOM. **Falhas em motores elétricos.** Disponível em: https://www.abecom.com.br/falhas-em-motores-eletricos. Acesso em 05 de nov. 2023.

ARAÚJO, Romero de Souza. **Desgaste prematuro e falhas recorrentes em rolamentos de motores de indução alimentados por inversores:** Análise e proposta de solução. 2011. 27 p. Dissertação (Mestrado) – Universidade Federal de Minas Gerais, Minas Gerais: UFMG, 2011. Disponível em: https://www.ppgee.ufmg.br/defesas/175M.PDF. Acesso em 05 de nov. 2023.

BERTOLETI, Pedro. **Projetos com ESP32 e LoRa.** 1. ed. São Paulo: Editora NCB, 2019.

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

CHAPMAN, Stephen. J. **Fundamentos de máquinas elétricas.** Tradução de Anatólio Laschuk. 5. ed. Porto Alegre: AMGH, 2013.

ECLIPSE FOUNDATION. **Introdução ao protocolo MQTT.** Disponível em: https://projects.eclipse.org/projects/iot.mqtt. Acesso em 02 mar. 2024

FACULDADE DE TECNOLOGIA DE SÃO BERNARDO DO CAMPO “ADIB MOISÉS DIB”. **Manual de Normalização de Projeto de Trabalho de Graduação:** Material didático para utilização nos projetos de trabalho de graduação, dos cursos de Tecnologia em Automação Industrial e Tecnologia em Informática para Negócios. 5. ed. revisada e atualizada. São Bernardo do Campo: Centro Paula Souza, 2017.

FERREIRA, Fernando; MANZAN, Renato; DURAES, Wellington. **Arquitetura de soluções IoT:** Desenvolva com Internet das Coisas para o mundo real. Controle do movimento e processos contínuos. 1. ed. São Paulo: Casa do Código, 2022.

HIVEMQ. **Fundamentos do MQTT.** Disponível em: https://www.hivemq.com/mqtt/. Acesso em 02 mar. 2024

NODE-RED. **Programação low-code para aplicações orientadas a eventos.** Disponível em: https://nodered.org/. Acesso em 02 mar. 2024

PANUNZIO, Paulo. **História de motores elétricos.** Disponível em: https://sistemas.eel.usp.br/docentes/arquivos/5840834/59/Historiamotoreseletricos.doc. Acesso em 03 nov. 2023.

PINYI MOTOR. **As vantagens e aplicações dos motores elétricos.** Disponível em: https://electricalmotor.net/blogs/blogs/the-advantages-and-applications-of-electric-motors-5f39b8a8. Acesso em 03 nov. 2023.

PRODANOV, C. C.; FREITAS, E. C. **Metodologia do trabalho científico:** Métodos e técnicas da pesquisa e do trabalho científico. 2. ed. Rio Grande do Sul: Universidade Feevale, 2013.

REZENDE, Sergio Machado. **Materiais e Dispositivos Eletrônicos.** 2.ed. São Paulo: Livraria da Física, 2004

THOMAZINI, Daniel; ALBUQUERQUE, Pedro. **Acionamentos industriais:** Fundamentos e aplicações. 4. ed. São Paulo: Érica, 2010.

YUAN, Michael. **Conhecendo o MQTT.** Disponível em: https://developer.ibm.com/articles/iot-mqtt-why-good-for-iot/. Acesso em 02 mar. 2024.

WEG. **Weg motores.** Disponível em: https://static.weg.net/medias/downloadcenter/h32/hc5/WEG-motores-eletricos-guia-de-especificacao-50032749-brochure-portuguese-web.pdf. Acesso em 03 nov. 2023

APÊNDICES

APÊNDICE A – PROGRAMAÇÃO DO MÓDULO HELTEC LORA (TRANSMISSOR)

```
#include <Wire.h>
#include <heltec.h>
#include <DHTesp.h>
#include <EmonLib.h>

#define DHTPIN 21            // Sensor de temperatura DHT22
#define CURRENT_SENSOR_PIN 13 // Sensor de corrente
#define PIEZO_SENSOR_PIN 12   // Sensor de vibração
#define RELE 2               // Relé para acionamento do motor
#define ALARM_LED 23         // Led de alarme, indicando algum erro
#define LED_OK 22            // Led indicando que o motor está ligado
#define BAND 915E6           // Frequência de transmissão LoRa

DHTesp dht;
EnergyMonitor emon1; // Cria uma instância para o sensor de corrente
EnergyMonitor emon2; // Cria uma instância para o sensor de vibração

//unsigned long lastSendTime = 0; // Timestamp da última transmissão
//const unsigned long interval = 5000; // Intervalo entre transmissões

void setup() {
  Serial.begin(9600);
  dht.setup(DHTPIN, DHTesp::DHT22);         // Inicializa o sensor DHT22
  emon1.current(CURRENT_SENSOR_PIN, 1.97);  // Calibração para ZMCT103C.
Ajuste se necessário
  emon2.current(PIEZO_SENSOR_PIN, 111.1);   // Calibração para o Piezoeletrico.
Ajuste se necessário
  pinMode(RELE, OUTPUT);                    // Define o Relé como saída
  digitalWrite(RELE, HIGH);                // Rele começa desligado. Observe que o
Módulo Relé aciona com sinal 0 (LOW)
  pinMode(ALARM_LED, OUTPUT);               // Define o Led de Alarme como saída
  digitalWrite(ALARM_LED, LOW);            // Led de Alarme começa desligado
  pinMode(LED_OK, OUTPUT);                  // Define o Led de Funcionamento do
motor como saída

  // Inicializa a biblioteca Heltec
  Heltec.begin(true, true, true, true, BAND);
  Heltec.display->init();
  Heltec.display->flipScreenVertically();
  Heltec.display->setFont(ArialMT_Plain_16);
  Heltec.display->clear();
  Heltec.display->drawString(33, 5, "Iniciado");
  Heltec.display->drawString(10, 30, "com Sucesso!");
```

```
  Heltec.display->display();
  delay(1000);

  // Use BAND e PABOOST
  if (!LoRa.begin(BAND, true)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }
}

void loop() {

    float temperature = dht.getTemperature();
    float current = emon1.calcIrms(1480);    // Número de amostras, ajuste conforme
necessário
    float vibration = emon2.calcIrms(1480);  // Número de amostras, ajuste conforme
necessário

    if (isnan(temperature)) {
      Serial.println("Falha na leitura do sensor de temperatura!");
      digitalWrite(ALARM_LED, HIGH); // Liga o Led de alarme
      delay(1000);
    } else {
      digitalWrite(ALARM_LED, LOW);
    }

  // Verifica se há comandos recebidos via LoRa
  int command = LoRa.parsePacket();
  if (command) {
    String command = "";
    while (LoRa.available()) {
      command += (char)LoRa.read();
    }
  }

    // Display values on OLED
    Heltec.display->clear();
    Heltec.display->drawString(0, 0, "Temp: " + String(temperature) + " °C");
    Heltec.display->drawString(0, 20, "Corrente: " + String(current) + " A");
    Heltec.display->drawString(0, 40, "Vibração: " + String(vibration) + " Hz");
    Heltec.display->display();

    // Debugging information
    Serial.print("Temperatura: ");
    Serial.print(temperature);
    Serial.print(" °C, Corrente: ");
    Serial.print(current);
    Serial.print(" A, Vibração: ");
    Serial.println(vibration);
```

```
    // Desliga o RELE
    if (current >= 1.5 || temperature >= 40.0) {
      digitalWrite(RELE, LOW); // Liga o RELE
      digitalWrite(LED_OK, LOW);
      digitalWrite(ALARM_LED, HIGH); // Liga o Led de alarme
    } else {
      digitalWrite(RELE, HIGH); // Desliga o RELE
      digitalWrite(LED_OK, HIGH);
      digitalWrite(ALARM_LED, LOW); // Desliga o Led de alarme
    }

    // Send values via LoRa
    LoRa.beginPacket();
    LoRa.print("Temp: ");
    LoRa.print(temperature);
    LoRa.println(" °C");
    LoRa.print("Corrente: ");
    LoRa.print(current);
    LoRa.println(" A");
    LoRa.print("Vibração: ");
    LoRa.print(vibration);
    LoRa.endPacket();

  delay(1000);  // Reduz o delay para melhorar a responsividade
}
```

APÊNDICE B – PROGRAMAÇÃO DO MÓDULO HELTEC LORA (RECEPTOR)

```
#include <Wire.h>
#include <heltec.h>
#include <WiFi.h>
#include <PubSubClient.h>

#define BAND 915E6

// Configurações WiFi e MQTT
const char* ssid = "Thiago"; // Substitua pelo nome da sua rede WiFi
const char* password = "Th1@g0FS"; // Substitua pela senha da sua rede WiFi
const char* mqtt_server = "broker.hivemq.com";

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE  (50)
char msg[MSG_BUFFER_SIZE];
int value = 0;

// Função de callback MQTT
void callback(char* topic, byte* payload, unsigned int length) {
  // Função de callback MQTT para lidar com comandos de controle
  Serial.print("Mensagem recebida [");
  Serial.print(topic);
  Serial.print("] ");
  String message = "";
  for (int i = 0; i < length; i++) {
    message += (char)payload[i];
  }
  Serial.println(message);

  if (String(topic) == "LoRa_Fatec_Control") {
    LoRa.beginPacket();
    LoRa.print(message);
    LoRa.endPacket();
    Serial.println("Comando enviado via LoRa: " + message);
  }
}

// Função de reconexão MQTT
void reconnect() {
  // Loop até estar conectado
  while (!client.connected()) {
    Serial.print("Tentando conexão MQTT...");
    // Create a random client ID
    String clientId = "LoRaEsp32";
    clientId += String(random(0xffff), HEX);
```

```
    // Attempt to connect
    if (client.connect(clientId.c_str())) {
      Serial.println("Conectado!");
      // Once connected, publish an announcement...
      client.publish("LoRa_Fatec_Out", "LoRa_Ok");
      // ... and resubscribe
      client.subscribe("LoRa_Fatec_In");
      client.subscribe("LoRa_Fatec_Control");
    } else {
      Serial.print("Falhou, rc=");
      Serial.print(client.state());
      Serial.println(" Tentando novamente em 5 segundos");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}

void setup() {
  Serial.begin(9600);
  Heltec.begin(true, true, true, true, BAND);
  Heltec.display->init();
  Heltec.display->flipScreenVertically();
  Heltec.display->setFont(ArialMT_Plain_16);
  Heltec.display->clear();
  Heltec.display->drawString(33, 5, "Iniciado");
  Heltec.display->drawString(10, 30, "com Sucesso!");
  Heltec.display->display();
  delay(1000);

  // Use BAND e PABOOST
  if (!LoRa.begin(BAND, true)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }

  // Conexão WiFi
  setup_wifi();

  // Configuração MQTT
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void setup_wifi() {
  delay(10);
  // Conectando à rede WiFi
  Serial.println();
  Serial.print("Conectando a ");
  Serial.println(ssid);
```

```
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi conectado");
  Serial.println("Endereço IP: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  if (!client.connected()) {
    reconnect(); // Reconnect MQTT if not connected
  }

  client.loop();

  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    String receivedText = "";
    while (LoRa.available()) {
      receivedText += (char)LoRa.read();
    }

    Serial.println("Dados recebidos via LoRa: " + receivedText);
    // Display received text on OLED
    Heltec.display->clear();
    Heltec.display->setFont(ArialMT_Plain_16);
    Heltec.display->drawString(0, 0, receivedText);
    Heltec.display->display();

    // Extrair os valores numéricos de cada métrica
    float temperature = getValue(receivedText, "Temp: ", " °C");
    float current = getValue(receivedText, "Corrente: ", " A");
    int vibration = getValue(receivedText, "Vibração: ", "");

    // Publicar os valores em tópicos MQTT separados
    if (client.connected()) {
      client.publish("LoRa_Fatec_Temp", String(temperature).c_str());
      client.publish("LoRa_Fatec_Current", String(current).c_str());
      client.publish("LoRa_Fatec_Vibration", String(vibration).c_str());
      Serial.println("Dados enviados via MQTT");
    } else {
      Serial.println("Cliente MQTT não conectado");
    }
  }
```

```
}

// Função para extrair valores numéricos de uma string
float getValue(String data, String prefix, String suffix) {
  int prefixIndex = data.indexOf(prefix);
  if (prefixIndex < 0) return 0.0;
  prefixIndex += prefix.length();
  int suffixIndex = suffix.length() > 0 ? data.indexOf(suffix, prefixIndex) : data.length();
  String valueStr = data.substring(prefixIndex, suffixIndex);
  return valueStr.toFloat();
}
```

APÊNDICE C – URL DO MÓDULO HELTEC LORA PARA O IDE ARDUINO

https://github.com/Heltec-Aaron-Lee/WiFi_Kit_series/releases/download/0.0.5/package_heltec_esp32_index.json