UNIVERSIDADE FEDERAL DE MINAS GERAIS

PROGRAMA DE PÓS-GRADUAÇÃO EM INOVAÇÃO TECNOLÓGICA

MARCELO SIMÕES REIS

DESENVOLVIMENTO DE UMA INFRAESTRUTURA DE DADOS COMO CÓDIGO E COM CAPACIDADE ANALÍTICA PARA MONITORAMENTO DE BOMBAS INDUSTRIAIS UTILIZANDO INTERNET DAS COISAS INDUSTRIAL

Belo Horizonte

2024

MARCELO SIMÕES REIS

DESENVOLVIMENTO DE UMA INFRAESTRUTURA DE DADOS COMO CÓDIGO E COM CAPACIDADE ANALÍTICA PARA MONITORAMENTO DE BOMBAS INDUSTRIAIS UTILIZANDO INTERNET DAS COISAS INDUSTRIAL

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Mestrado Profissional em Inovação Tecnológica e Propriedade Intelectual da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Mestre em Inovação Tecnológica

Área de concentração: Inovação Tecnológica

Linha de pesquisa: Desenvolvimento de Soluções de Internet das Coisas (IoT) para Solução de Problemas Interdisciplinares

Orientador: Prof. Dr. Adriano Borges da Cunha

BELO HORIZONTE
2024

043 Reis, Marcelo Simões.
Desenvolvimento de uma infraestrutura de dados como código e com capacidade analítica para monitoramento de bombas industriais utilizando internet das coisas industrial [manuscrito] / Marcelo Simões Reis. – 2024.
94 f.: il. ; 29,5 cm.

Orientador: Prof. Dr. Adriano Borges da Cunha.
Dissertação (mestrado) – Universidade Federal de Minas Gerais, Instituto de Ciências Biológicas. Mestrado Profissional em Inovação Tecnológica e Propriedade Intelectual.

1. Inovação tecnológica. 2. Internet das Coisas. 3. Arquitetura de rede de sistemas (SNA). 4. Indústria. I. Cunha, Adriano Borges da. II. Universidade Federal de Minas Gerais. Instituto de Ciências Biológicas. III. Título.

CDU: 608.5

Ficha catalográfica elaborada pela bibliotecária Fabiane C M Reis – CRB 6 – 2680

"DESENVOLVIMENTO DE UMA INFRAESTRUTURA DE DADOS COMO CÓDIGO E COM CAPACIDADE ANALITICA PARA MONITORAMENTO DE BOMBAS INDUSTRIAIS UTILIZANDO INTERNET DAS COISAS INDUSTRIAL"

MARCELO SIMÕES REIS

Dissertação de Mestrado defendida e aprovada no dia 26 de junho de 2024, pela Banca Examinadora constituída pelos seguintes membros:

PROF. DR. LUCAS VINÍCIUS RIBEIRO ALVES,
COLTEC/UFMG

PROF. DR. JOÃO EDUARDO MONTANDON DE ARAÚJO FILHO
COLTEC/UFMG

ME. DANIEL CARDOSO LOPES

PROF. DR. ADRIANO BORGES DA CUNHA – ORIENTADOR
COLTEC/UFMG

Instituto de Ciências Biológicas – Universidade Federal de Minas Gerais – UFMG

Belo Horizonte, 26 de junho de 2024

RESUMO

Este trabalho apresenta o desenvolvimento e a implementação de uma infraestrutura de dados para ambientes industriais, abrangendo todo o fluxo de informações, desde a coleta de dados dos sensores em campo, armazenamento em nuvem até a apresentação em dashboards. São descritas as etapas de desenvolvimento do hardware, as decisões sobre tecnologias de comunicação e protocolos de rede sem fio, definição da arquitetura, a implementação de firmwares e softwares para otimização de custo e performance, e a modelagem da estrutura de dados.

Inicialmente, desenvolveu-se uma infraestrutura física de baixo custo destinada a conectar ilhas de informação dentro de ambientes industriais. O objetivo foi garantir que mesmo as áreas mais isoladas dentro de uma planta industrial possam se comunicar de maneira eficaz com o sistema central e criar uma base de dados qualificada.

Em seguida, a pesquisa se concentrou na criação de uma infraestrutura lógica escalável e econômica. Esta infraestrutura foi projetada para armazenar grandes volumes de dados industriais, com foco em uma implementação rápida e na rastreabilidade das modificações realizadas, utilizando conceitos de contêiners e infraestrutura como código.

Por fim, foi garantida a capacidade da infraestrutura de dados, tanto lógica quanto física, de fornecer informações dos elementos monitorados de forma que sistemas de terceiros possam consumir esses dados para análise. Essa capacidade de fornecer dados de qualidade e com bom tempo de resposta é crucial para transformar o perfil de manutenção da indústria de um modelo reativo para um modelo proativo.

Os resultados obtidos demonstram a viabilidade e a eficácia da solução proposta, oferecendo uma infraestrutura robusta e de baixo custo que pode ser facilmente integrada e escalonada em diversos ambientes industriais. A dissertação conclui que a implementação de tais sistemas pode trazer significativas melhorias operacionais e econômicas para a indústria, promovendo uma manutenção proativa e uma gestão de dados mais eficiente.

Palavras-chave: IIOT, IaC, LoRa, Containers, Kubernetes

ABSTRACT

This work presents the development and implementation of a data infrastructure for industrial environments, covering the entire information flow, from data collection from sensors in the field, cloud storage to presentation on dashboards. The hardware development steps, decisions on communication technologies and wireless network protocols, the definition of the architecture, the implementation of firmware and software for cost and performance optimization, and a data structure model are described.

Initially, the development is a low-cost physical infrastructure designed to connect islands of information within industrial environments. The objective was to ensure that even the most isolated areas within an industrial plant can communicate effectively with the central system and create a poor database.

Next, research focused on creating a scalable and cost-effective logical infrastructure. This infrastructure was designed to store large volumes of industrial data, with a focus on rapid implementation and traceability of modifications made, using concepts of containers and infrastructure as code.

Finally, the capacity of the data infrastructure, both logical and physical, to provide information on monitored elements was guaranteed so that third-party systems can consume this data for analysis. This ability to provide quality data with good response time is crucial to transforming the industry's maintenance profile from a reactive model to a proactive model.

The results obtained demonstrate the solutions and effectiveness of the proposed solution, offering a robust and low-cost infrastructure that can be easily integrated and scaled in different industrial environments. The dissertation concludes that the implementation of such systems can bring significant operational and economic improvements to the industry, promoting proactive maintenance and more efficient data management.

Keywords: IIOT, IaC, LoRa, Containers, Kubernetes

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ANEXOS

LISTA DE ABREVIATURAS

3GPP - Third-Generation Partnership Project: Organização que desenvolve padrões para tecnologias móveis, como 3G, 4G e 5G.

6LoWPAN: IPv6 over Low power Wireless Personal Area Networks

AES-128: Advanced Encryption Standard with 128-bit key

AMQP - Advanced Message Queuing Protocol: Protocolo de mensagens aberto e padronizado para sistemas de mensagens assíncronas.

C4.5 - Classification and Regression Tree (versão anterior ao CART): Algoritmo de árvore de decisão para classificação e regressão.

CAN - Controller Area Network: Protocolo de comunicação serial usado principalmente em sistemas embarcados e automotivos.

CART - Classification and Regression Trees: Método de aprendizado de máquina baseado em árvores de decisão.

CBM - Condition-based Maintenance: Estratégia de manutenção que monitora o estado de um equipamento para determinar quando a manutenção é necessária.

FFT - Fast Fourier Transform: Algoritmo para transformar sinais de domínio temporal em domínio de frequência.

IaC - Infrastructure as Code: Prática de gerenciar infraestruturas de TI usando código e recursos de automação.

ID3 - Iterative Dichotomiser 3: Algoritmo de árvore de decisão usado em aprendizado de máquina para classificação.

IIoT - Internet of Things Industrial: Uma subcategoria do IoT que se concentra na aplicação de tecnologias IoT em ambientes industriais.

IoT - Internet of Things: Refere-se à interconexão de dispositivos físicos por meio da Internet.

IPv6: Internet Protocol version 6

ISM - Industrial, Scientific, and Medical: Banda de frequência reservada para uso industrial, científico e médico.

KAFKA - Apache Kafka: Plataforma de streaming distribuída para processamento de dados em tempo real.

KNN - k-nearest neighbors: Algoritmo de classificação e regressão baseado na proximidade de instâncias.

LoRaWAN: Protocolo de rede de baixa potência e longo alcance (LPWAN) projetado para comunicação entre dispositivos IoT.

LPWA - Low Power Wide Area: Tecnologia de comunicação de longo alcance e baixo consumo de energia.

LPWAN - Low Power Wide Area Network: Rede de longo alcance e baixo consumo de energia, ideal para dispositivos IoT.

MQTT - Message Queuing Telemetry Transport: Protocolo de mensagens leve e baseado em assinatura, adequado para comunicação entre dispositivos IoT.

NB-IoT - Narrowband Internet of Things: Tecnologia de comunicação de baixa potência e largura de banda estreita para dispositivos IoT.

OASIS - Organization for the Advancement of Structured Information Standards: Consórcio que desenvolve padrões abertos para tecnologias de informação e comunicação.

OPC DA - OPC Data Access: Padrão de comunicação para troca de dados em sistemas de automação.

OPC UA - OPC Unified Architecture: Arquitetura de comunicação aberta e padronizada para sistemas industriais.

O-QPSK - Offset Quadrature Phase-Shift Keying: Modulação de amplitude em quadratura deslocada utilizada em comunicações digitais.

QoS - Quality of Service: Medida da qualidade de transmissão de dados em uma rede.

SOA - Arquitetura orientada a serviços: Modelo de arquitetura de software em que os componentes são projetados para serem serviços independentes.

TLS - Transport Layer Security: Protocolo de segurança que fornece comunicação segura na Internet.

VMs - Máquinas Virtuais: Ambiente de computação virtual que opera como uma instância de sistema computacional completo.

Wi-Fi - Wireless Fidelity: Tecnologia de rede sem fio que permite a conexão de dispositivos à Internet ou a outras redes.

Zigbee: Protocolo de comunicação sem fio de baixo consumo de energia, usado principalmente em aplicações de controle e monitoramento.

SUMÁRIO

1 – Introdução

No ambiente industrial possuímos cenários bem diversos em relação à manutenção e conectividade. O cenário industrial brasileiro ainda possui a manutenção muito reativa, sendo que a maior parte das ações da manutenção são corretivas, aumentando assim o custo da manutenção e impactando o processo produtivo.

Com relação à conectividade, existem indústrias que possuem ilhas de informação, onde muitas vezes os equipamentos possuem um automatismo para executar a ação para aquela etapa do processo e diagnósticos básicos, mas efetivamente são isolados do ponto de vista de comunicação e transferência de dados para outros elementos, impossibilitando um tratamento global e sistêmico das informações deste ativo.

1.1 Problema e justificativas

Em um cenário onde a manutenção é, em sua maioria, reativa e existem muitas informações pouco integradas, o monitoramento de saúde de bombas industriais irá impactar positivamente o negócio dessas indústrias. Este trabalho propõe uma infraestrutura física e lógica de dados para possibilitar que a manutenção desses ativos se torne proativa.

Essa infraestrutura irá possibilitar a coleta de informações de bombas industriais para uma estrutura de dados que possibilite capacidade analítica, de forma assertiva e segura, estejam elas isoladas em ilhas de informação ou pertencentes à uma rede de automação existente. Com isso, os clientes dessa infraestrutura poderão utilizar de algoritmos de análise de dados para trabalhar com os dados coletados para atribuir características qualitativas ao ativo e definir o status de saúde e funcionamento dele, o que tornaria a manutenção mais eficiente, podendo até sair de um padrão de manutenção corretiva para preditiva.

1.2 Objetivos

Implementar uma infraestrutura física e lógica de dados que possibilite uma tratativa para que a manutenção de bombas industriais se torne proativa. O uso de técnicas de análise de dados e aprendizado de máquinas, através de uma abordagem de manutenção baseada em condição, permitirão prognósticos com indicadores de saúde das bombas e mudará o patamar da manutenção de bombas para proativa, diminuindo assim os custos de manutenção e impacto na produção. Essa solução de monitoramento de bombas industriais é denominada Mobi.

1.2.1 Objetivos Específicos

Os objetivos específicos deste trabalho abrangem três grandes âmbitos:

- **Infraestrutura Física de Baixo Custo**: Desenvolver uma infraestrutura física econômica para conectar ilhas de informação, utilizando um dispositivo final conectado a um *gateway*, proporcionando conectividade para ambientes industriais.
- **Infraestrutura Lógica Escalável e Econômica**: Criar uma infraestrutura lógica escalável, de baixo custo, com rápida implementação e rastreabilidade de modificações, destinada ao armazenamento de dados industriais.
- **Integração e Análise de Dados**: Assegurar que a infraestrutura de dados, tanto lógica quanto física, tenha a capacidade de fornecer informações a um sistema analítico. Isso permitirá a transformação do perfil de manutenção da indústria para um modelo proativo.

1.3 Organização do trabalho

O Capítulo 2 apresenta uma revisão bibliográfica e revisão teórica baseada em artigos científicos de revistas especializadas e livros sobre os temas abordados, tratando, respectivamente, conectividade, arquitetura de sistemas, manutenção e análise de dados.

O Capítulo 3 descreve os procedimentos metodológicos que foram utilizados neste trabalho, desde a definição de hardwares até a camada de visualização das informações para o mantenedor.

O Capítulo 4 apresenta a estruturação da solução em camadas, suas características e seus diferenciais em relação ao mercado.

O Capítulo 5 descreve as validações da nova infraestrutura de dados para um ambiente analítico.

O Capítulo 6 trata das conclusões e considerações gerais, bem como apresenta sugestões para trabalhos futuros.

2 – Revisão da literatura

Este capítulo apresenta a revisão da literatura utilizado para a realização deste trabalho. Objetivando tratar de aspectos voltados aos objetivos desta dissertação foi realizada uma revisão bibliográfica sistemática nas bases de dados de jornais, revistas técnicas, artigos e teses.

Com isso, foram obtidos artigos, teses de doutorado e livros onde foi realizada triagem, classificação e excluídos conteúdos repetidos e pouco significativos. A partir deste primeiro resultado, os conteúdos alinhados com a proposta deste trabalho embasaram o referencial teórico utilizado.

O objetivo do referencial teórico foi de buscar, inicialmente, como tem sido abordada a conectividade, arquiteturas de sistemas, manutenção e uso de análise de dados para manutenção preditiva. A Figura 1 ilustra as etapas desta revisão bibliográfica estruturada:

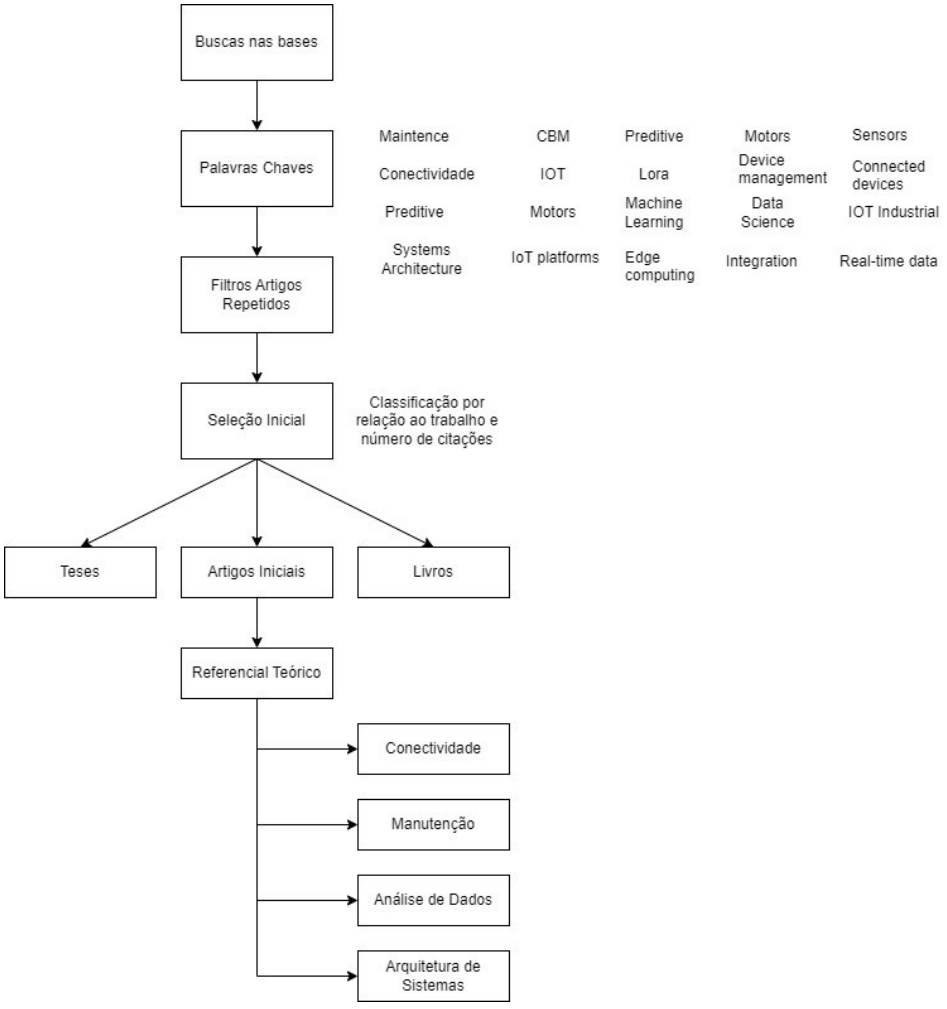


Figura 1 - Estrutura de pesquisa bibliográfica (Fonte: Próprio)

As seções posteriores destinam-se a embasar teoricamente as decisões tecnológicas e estratégicas da solução baseado na revisão da literatura. As seções posteriores deste capítulo irão navegar por conceitos de arquitetura, IoT, conectividade, manutenção e análise de dados.

2.1 - Arquitetura de Sistemas

Quando tratamos de arquitetura de sistemas devemos nos atentar para duas abordagens de arquitetura: lógicas e físicas. A arquitetura lógica de sistemas se refere à organização e interação dos componentes de software, delineando a estrutura conceitual e funcional do sistema, incluindo a divisão em módulos, serviços e a lógica de negócios subjacente. Por outro lado, a arquitetura física trata dos aspectos tangíveis do sistema, como hardware, redes e infraestrutura de implantação, descrevendo a configuração real dos recursos computacionais, a distribuição geográfica dos dispositivos e os requisitos de desempenho. Enquanto a arquitetura lógica se concentra na lógica e na funcionalidade do sistema, a arquitetura física está preocupada com a implementação concreta e os recursos físicos necessários para executar o sistema.

Usualmente, no ambiente industrial, esses conceitos se fundem e principalmente para o grande público, as arquiteturas de sistemas industriais tratam da estrutura organizacional e funcional dos sistemas usados nas industriais para controle de processo, automatização, otimização e gestão do processo e do negócio.

Tradicionalmente, existem diversas arquiteturas de sistemas industriais, tais como: centralizada, distribuída, em camadas, em rede e em nuvem.

Arquitetura Centralizada

Nesta abordagem, todos os dispositivos e sensores em uma instalação industrial estão conectados a um controlador central. Esse controlador é responsável por receber dados, processá-los e enviar comandos de volta para os dispositivos. A arquitetura centralizada é simples de implementar e oferece controle direto sobre todos os dispositivos, mas pode ser limitada em termos de escalabilidade e flexibilidade.

Arquitetura Distribuída

Nesta configuração, o controle e a automação são distribuídos entre vários controladores locais ou células de controle. Cada controlador é responsável por uma parte específica do processo ou da linha de produção. Os controladores podem se comunicar entre si para coordenar operações e trocar dados. A arquitetura distribuída é mais flexível e escalável do que a centralizada e pode oferecer um desempenho melhor em ambientes industriais complexos.

Arquitetura em Camadas

Esta abordagem divide o sistema em várias camadas distintas, cada uma com funções específicas. As camadas podem incluir a interface do usuário, lógica de controle, comunicação de rede e acesso a dispositivos. A arquitetura em camadas facilita a modularidade, manutenção e atualização do sistema, permitindo que as mudanças em uma camada não afetem necessariamente as outras.

Arquitetura em Rede

Nesta configuração, os dispositivos industriais estão interconectados por meio de uma rede de comunicação, como Ethernet industrial, PROFINET, Modbus TCP, entre outras. Isso permite uma comunicação rápida e eficiente entre os dispositivos, facilitando a integração de sistemas e o acesso remoto.

Arquitetura de Automação em Nuvem

Com o avanço da tecnologia de nuvem, muitas empresas estão adotando arquiteturas de automação em nuvem, onde os dados dos dispositivos industriais são enviados para serviços em nuvem para análise, armazenamento e processamento. Isso permite acesso remoto aos dados e insights em tempo real sobre o desempenho do sistema, além de facilitar a implementação de soluções de manutenção preditiva e otimização de processos.

Para sistemas de IoT Industrial, podemos destacar uma combinação dessas arquiteturas, aproveitando ao máximo suas capacidades de distribuição, operação em camadas e utilização eficiente dos recursos disponíveis.

2.2 - IoT

Quando se trata de IoT, temos inúmeras possibilidades de conectar os dispositivos, recuperar e inserir informações nesses dispositivos por meio de uma rede, temos duas vertentes a serem consideradas: IoT para a sociedade e IoT Industrial.

Essas duas vertentes, apesar de muitas vezes utilizarem tecnologias similares, possuem requisitos bem distintos, sejam eles funcionais ou não funcionais.

Sistemas de IoT para a sociedade muitas vezes focam em requisitos funcionais e não possuem missão crítica, ou seja, uma indisponibilidade não impactaria o negócio ou a vida de forma significativa. Sistemas de IoT Industriais possuem requisitos muitas vezes intimamente ligados aos indicadores de negócio da Industria, ou seja, a indisponibilidade de um desses sistemas poderia acarretar perda de produtividade, qualidade ou eficiência impactando negativamente os indicadores das indústrias.

IoT Industrial

Dificilmente conseguiremos identificar a primeira vez que o termo IoT Industrial (IIoT) foi cunhado, porém, conceitualmente, o termo ganhou notoriedade a partir da quarta revolução industrial iniciada na Alemanha, na feira de Hannover em 2011, onde a Alemanha lançou um programa de Industria 4.0 com o objetivo estratégico de tornar a indústria mais eficiente e competitiva, apoiada em ferramentas tecnológicas como IoT, computação em nuvem, inteligência artificial e outras.

Com isso, a IoT ganha o protagonismo importante no contexto de digitalização das indústrias, possibilitando conectividade de diversas formas diferentes e inclusive, quebrando o paradigma industrial da pirâmide de automação tradicional, possibilitando uma horizontalidade da informação.

2.3 – Conectividade

2.3.1 – Arquitetura IIoT

Podemos tratar a arquitetura IIoT em uma arquitetura composta por três camadas, cada uma desempenhando um papel no ecossistema da IIoT (T. HAFEEZ, 2020), conforme representado na Figura 2.

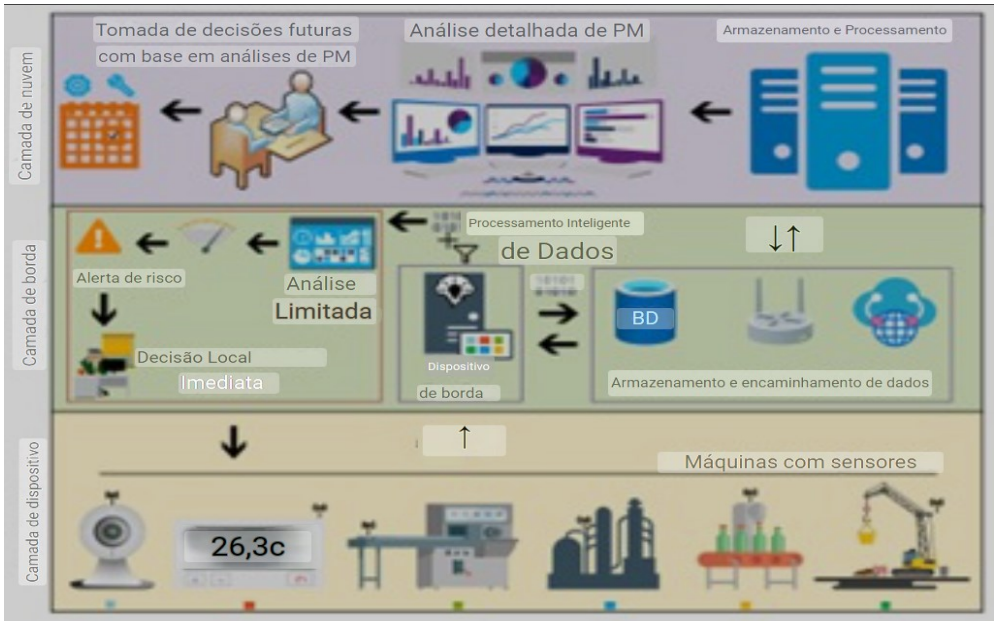


Figura 2 - Arquitetura baseada em camadas. (Fonte: T. HAFEEZ, 2020)

2.3.2 – Camada de Dispositivos

Esta é a camada mais baixa da arquitetura IIoT, composta por dispositivos físicos que coletam dados do ambiente. Estes dispositivos podem variar desde sensores simples, como temperatura, umidade e luminosidade, até dispositivos mais complexos, como câmeras, drones ou sistemas de monitoramento de saúde. A principal função desses dispositivos é capturar dados do mundo real e transmiti-los para a próxima camada da arquitetura.

2.3.3 – Camada de Comunicação

A camada de comunicação é responsável por transmitir os dados coletados pelos dispositivos IoT para a nuvem ou para uma infraestrutura de processamento e análise. Esta camada pode envolver uma variedade de tecnologias de comunicação, incluindo redes sem fio (como Wi-Fi, Bluetooth, Zigbee, LoRaWAN) ou redes cabeadas (como Ethernet, CAN). Seu principal objetivo é garantir uma comunicação confiável e segura entre os dispositivos e a próxima camada da arquitetura.

Na camada de comunicação, entre alguns elementos importantes a serem considerados, dois em específico merecem destaque: tecnologia de comunicação e protocolo. Esses itens estão intimamente ligados ao tema conectividade. Esse tema é essencial para propiciar um ambiente de dados que propicie uma capacidade analítica no futuro.

Tecnologias de Conectividade

Nos últimos anos, o avanço acelerado da tecnologia tem impulsionado a proliferação de dispositivos conectados à Internet, tornando a comunicação sem fio uma parte indispensável de nossas vidas cotidianas. As tecnologias sem fio desempenham um papel vital na habilitação da Internet das Coisas (IoT), permitindo a troca de dados entre dispositivos e sistemas de forma eficiente e conveniente.

Desde a evolução dos padrões de Wi-Fi, que possibilitaram a conectividade de dispositivos em redes locais, até tecnologias de longo alcance como LoRaWAN que viabilizam a comunicação em amplas áreas geográficas, o espectro de tecnologias sem fio oferece uma gama diversificada de opções para diferentes necessidades e cenários de aplicação. Essas tecnologias não apenas conectam dispositivos entre si, mas também capacitam uma ampla variedade de aplicações, desde automação residencial e industrial até cidades inteligentes e saúde conectada.

No entanto, com a diversidade de tecnologias disponíveis, surge a necessidade de compreender suas características, vantagens e limitações para selecionar a solução mais adequada para cada aplicação específica. Neste contexto, esta seção visa fornecer uma visão geral das tecnologias sem fio de comunicação mais utilizadas no âmbito de IoT. As seções posteriores descrevem conceitos e aplicações destas tecnologias.

Zigbee

O Zigbee é um padrão de comunicação sem fio, de baixo consumo de energia e curto alcance, projetado especificamente para aplicações de redes de sensores sem fio (RSSF).

Em 2002 foi criada a ZigBee Alliance, uma associação de empresas, universidades e agências governamentais, que tinham o intuito de desenvolver o protocolo ZigBee.

ZigBee é um padrão que implementa uma pilha de protocolos sobre duas camadas, a camada de aplicação APL (Aplication Layer) e a camada de rede NWK (Network Layer), conforme apresentado na Figura 3. Esse padrão estabelece uma implementação simplificada, focada em 3 principais características: baixo custo, baixo consumo de energia e reduzida taxa de transferência de dados (250 kbps).

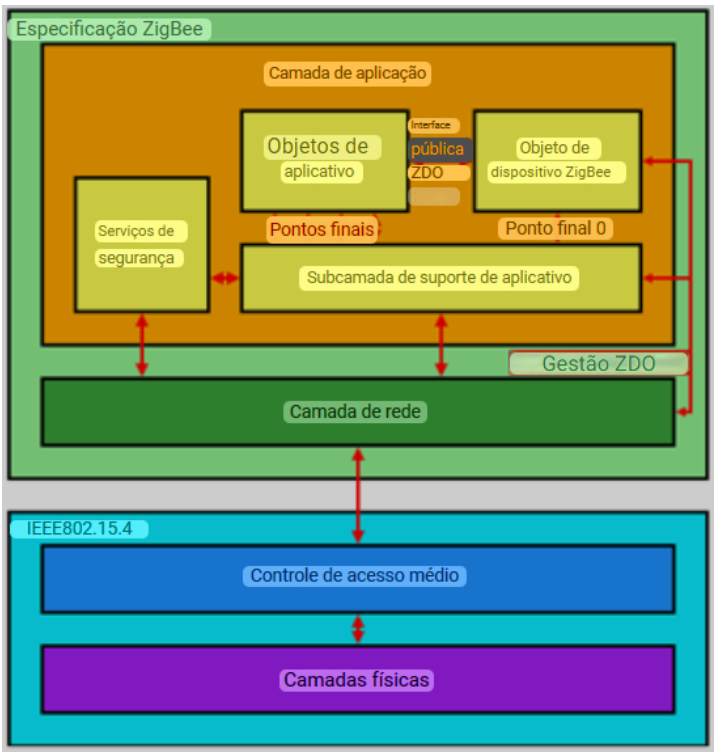


Figura 3 - Arquitetura Zigbee

A tecnologia Zigbee é amplamente utilizada em ambientes de monitoramento e controle industrial, automação residencial e sistemas de energia, destacando-se pelo seu baixo consumo de energia e comunicação sem fio confiável e eficiente (ZIGBEE ALLIANCE, 2021)

O Zigbee opera na faixa de frequência de 2,4 GHz e utiliza a tecnologia de modulação (O-QPSK - *Offset Quadrature Phase Shift Keying*) para garantir uma comunicação confiável e imune a interferências. A arquitetura de rede permite que os dispositivos comuniquem entre si permitindo assim uma rede auto organizável e escalável.

O Zigbee pode ser caracterizado por quatro serviços: encriptação, associação e autenticação, roteamento, serviços de aplicação.

- Serviço de Encriptação – os serviços garantem que as chaves de aplicações e de rede implementem a criptografia AES (Advanced Encryption Standard) de 128 bits, trazendo para o protocolo segurança e integridade da informação.
- Serviço de associação e autenticação - esse serviço permite somente que nós válidos podem se juntar à rede, garantindo autorização nomeada a camada de segurança do protocolo.
- Serviço de roteamento: o serviço implementa um roteamento baseado no protocolo AODV (Ad Hoc On-Demand Distance Vector). Esse protocolo é adaptativo a cenários de alta mobilidade, que tende a evitar desperdício de banda, minimizando o processamento nos nós da rede.
- Serviços de Aplicação: esse serviço aplica um conceito de "cluster" onde cada nó pertence a um cluster e possui permissões especificas de ações predefinidas.

O Zigbee funcionalmente possui 3 atores principais, esses tipos de nós são:

- Coordenador: é o dispositivo "mestre", governa toda a rede
- Roteadores: roteiam as informações enviadas pelos dispositivos finais
- Dispositivo final: são os nós sensores, eles coletam os dados das plantas industriais.

A interação entre os elementos e serviços do protocolo e sequenciamento desde fluxo seguem as seguintes etapas.

A primeira etapa é quando um nó quer se juntar à rede, para isso deve solicitar ao coordenador um endereço de rede. Esse endereço possui todas as informações da rede em 16 bits, executando os serviços de autenticação e a criptografia.

Após o aceite do nó na rede, ele tem a capacidade de enviar pacotes aos outros nós da rede através dos roteadores. O roteador, por sua vez, ao obter o pacote e o destino, verifica se o dispositivo final está acordado ou dormindo. Se acordado, o roteador envia o pacote ao destino e, caso esteja dormindo, o roteador retém o pacote aguarda que o dispositivo final solicite ingresso na rede novamente e então retransmite o pacote.

Os modos de operação do Zigbee, “beaconing” e o “non-beaconing”, contribuem para baixo consumo de energia

O “beaconing” trata-se de um modo que os roteadores enviam periodicamente “beacon frames”, mensagens sinalizadoras que confirmam sua presença na rede. Assim os nós podem se manter inativos durante esse intervalo entre o envio de uma mensagem e outra, poupando energia.

O “non-beaconing” não utiliza os “beacon frames”. Dependendo do cenário, pode ser custoso manter essa sincronia, assim é melhor que os dispositivos permaneçam ativos todo o tempo, sacrificando o consumo de energia.

O Zigbee é uma tecnologia importante e que traz uma boa combinação de baixo consumo de energia, alcance estendido e interoperabilidade. Portanto, deve ser avaliada na escolha de redes de sensores sem fio avaliando suas características e cenários de aplicação.

LoRaWAN

“LoRaWAN® é um protocolo de rede de baixo consumo de energia e área ampla (LPWA) projetado para conectar coisas sem fio operadas por bateria à Internet em redes regionais, nacionais ou globais e atende aos principais requisitos da Internet das Coisas (IoT), como comunicação bidirecional, segurança ponta a ponta, serviços de mobilidade e localização.” (LoRa Alliance, 2024).

O LoRaWAN (Long Range Wilde Area Network) é uma tecnologia desenvolvida pela empresa Semtech Corporation em parceria com a LoRa Alliance, uma organização de colaboração de empresas que trabalham para desenvolver e promover o padrão LoRaWAN. A Semtech Corporation é a empresa que desenvolveu a tecnologia LoRa (Long Range), que serve de base para o protocolo LoRaWAN (SEMTECH CORPORATION, 2024).

O LoRaWAN é resultado de uma colaboração entre várias empresas e organizações empenhadas no desenvolvimento e na adoção de tecnologias de comunicação sem fio para IoT.

De forma geral a LoRaWAN em sua arquitetura possui camadas que estruturam a comunicação entre dispositivo final, gateway e aplicação, conforme apresentado na Figura 4. Porém, a utilização do LoRaWAN implica em utilizar dispositivos finais e gateways homologados pela LoRa Alliance.

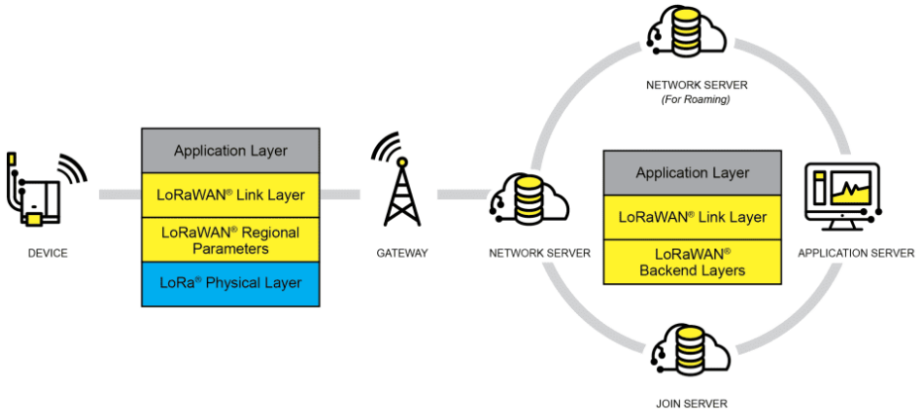


Figura 4 - Arquitetura LoRaWAN.

O LoRaWAN é um protocolo de comunicação projetado para dispositivos de baixo consumo de energia em redes de longa distância. O protocolo opera em uma arquitetura estrela, onde dispositivos finais (sensores ou atuadores) enviam dados por modulação LoRa para gateways. Esses gateways retransmitem os dados para um servidor de rede usando uma conexão IP padrão. O servidor de rede gerencia a comunicação, assegura a segurança com criptografia AES-128 e encaminha os dados para servidores de aplicação que processam as informações para diversos usos.

O LoRaWAN é eficaz para aplicações que exigem transmissão de pequenos pacotes de dados com baixa frequência, garantindo longa duração da bateria dos dispositivos conectados.

Algumas caraterísticas do LoRa a coloca em posição de destaque em relação as possibilidades das tecnologias sem fio aplicáveis a IoT Industrial. Estas características são:

- Baixo Consumo de Energia: Essa característica permite que os dispositivos operem por longos períodos com uma única bateria ou fonte de energia.
- Comunicação de Longa Distância: O LoRa utiliza modulação de espalhamento espectral, o que lhe permite transmitir sinais em distâncias muito maiores do que outras tecnologias sem fio convencionais, mesmo em ambientes urbanos densos e com obstáculos. O LoRa possui uma capacidade de alcançar distâncias longas podendo atingir vários quilômetros em áreas urbanas e até 15 quilômetros em ambientes rurais, dependendo das condições específicas do ambiente.

- Baixo Custo de Implementação: A infraestrutura necessária para implantar uma rede LoRa é relativamente simples e de baixo custo em comparação com outras tecnologias de comunicação sem fio de longo alcance.
- Flexibilidade e Escalabilidade: O LoRa é altamente flexível e pode ser configurado para atender a uma variedade de requisitos de aplicação, desde monitoramento remoto de sensores até aplicações industriais complexas.

Uma forma de utilizar a arquitetura de sistemas de IIoT usando a tecnologia LoRa, é um dispositivo final simples coletando informações da planta e comunicando via LoRa com o gateway. O gateway, por sua vez, comunicando com a camada de aplicação usando um protocolo de comunicação como o MQTT, esse exemplo de aplicação pode ser observado na Figura 5.

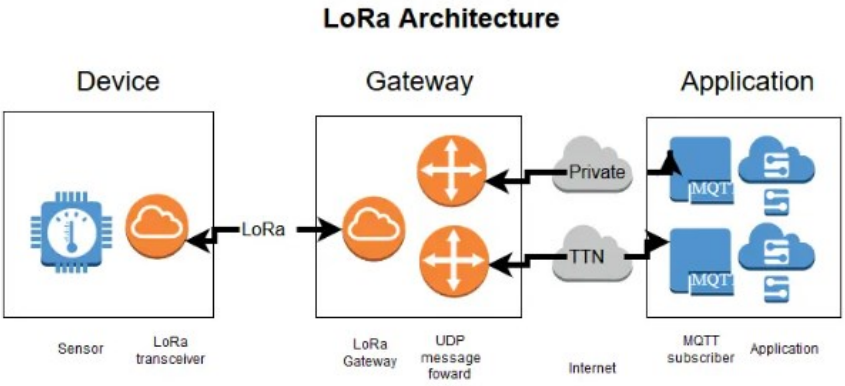


Figura 5 - Arquitetura LoRa.

O LoRa tem se destacado como uma tecnologia de comunicação sem fio versátil e altamente eficiente para uma variedade de aplicações de IIoT. Sua combinação única de longo alcance, baixo consumo de energia, baixo custo e flexibilidade o torna uma escolha atraente para uma ampla gama de cenários, desde monitoramento remoto de sensores em áreas rurais até aplicações urbanas em cidades inteligentes.

Segundo Centenaro, essa tecnologia apresenta grandes oportunidades e benefícios para aplicações na IIoT com potencial de oferecer uma cobertura abrangente e eficiente, permitindo a implementação de soluções de monitoramento e controle em larga escala (CENTENARO, 2016).

Thread

O protocolo Thread foi desenvolvido pela Thread Group, uma organização sem fins lucrativos composta por várias empresas líderes do setor de tecnologia. As

empresas fundadoras do Thread Group incluem grandes nomes da indústria de tecnologia, como Google, Samsung, Qualcomm, ARM, NXP Semiconductors e outras. Essas empresas colaboraram para desenvolver o protocolo Thread e promover sua adoção em uma ampla variedade de dispositivos IoT (THREAD GROUP, 2024).

O protocolo utiliza de padrões abertos e foi projetado para oferecer uma solução robusta e escalável para redes IoT, com foco em baixo consumo de energia, segurança e interoperabilidade.

Thread é um protocolo de rede sem fio em malha de baixo consumo de energia e baixa latência, construído usando padrões abertos e comprovados, baseado no Protocolo da Internet versão 6 (IPv6). Thread resolve as complexidades da IoT, abordando desafios como interoperabilidade, alcance, segurança, energia e confiabilidade.

Uma das principais vantagens desse protocolo é a capacidade de se integrar a outras redes IP e não precisa de gateways proprietários, a Figura 6 exemplifica esse tipo de integração. Essa característica reduz o investimento e a complexidade da infraestrutura, eliminando potenciais pontos de falha e reduzindo encargos de manutenção. Thread também conecta dispositivos à nuvem com segurança, facilitando o controle de produtos e sistemas IoT a partir de dispositivos como telefones celulares e tablets.

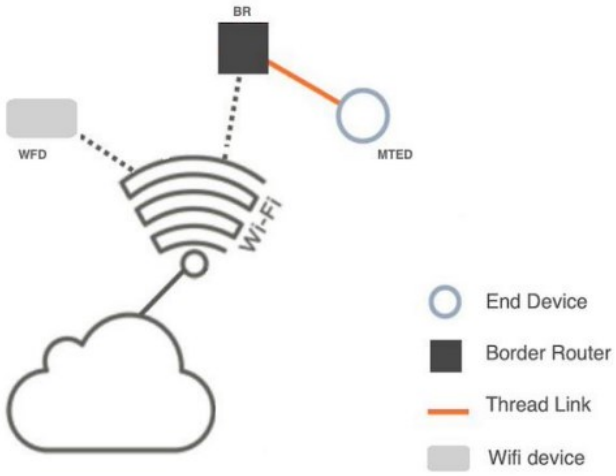


Figura 6 -Diagrama Físico de Interligação Thread.

A base IP da Thread é independente da camada de aplicação, oferecendo aos fabricantes a flexibilidade em escolher uma ou múltiplas camadas de aplicação para seu caso de uso para conectar dispositivos em múltiplas redes, como exemplificado na Figura 7.

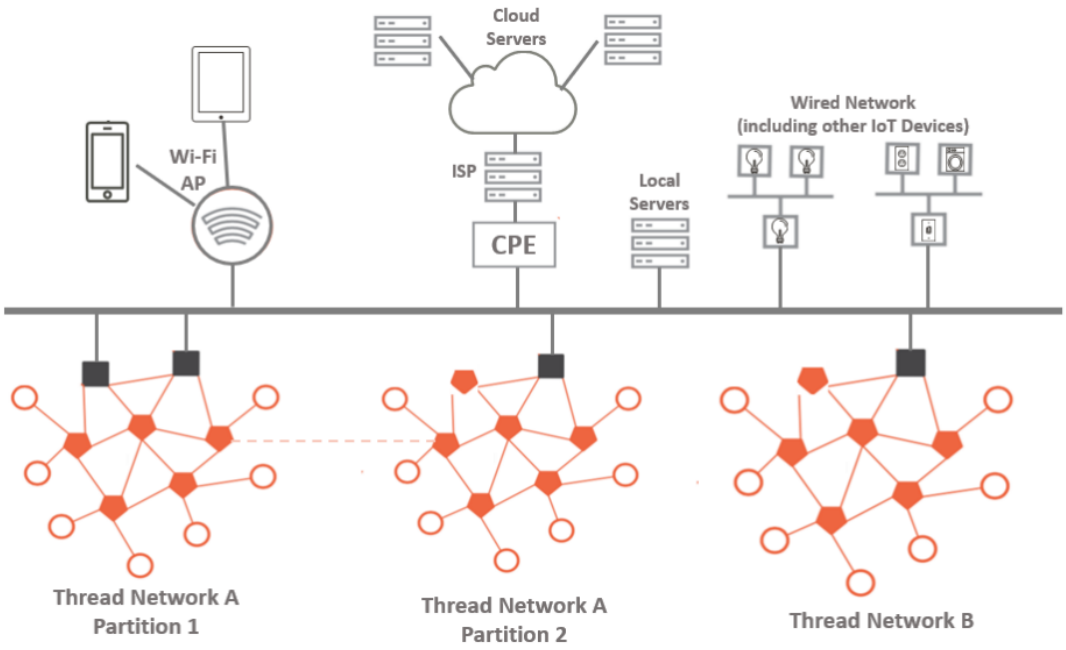


Figura 7 - Múltiplas Threads Networks compartilhando usando um backbone link.

Com o Thread, os desenvolvedores podem lançar seus aplicativos, dispositivos, sistemas e serviços no mercado com mais rapidez, pois usam o mesmo conjunto de ferramentas disponíveis para a Internet, uma vez que usa o IP como base de comunicação. Além do mais, a camada de aplicação e os serviços de nuvem em dispositivos Thread podem ser alterados ao longo do tempo, porque o protocolo é independente da camada de aplicação.

Thread é baseado em tecnologias existentes em todas as suas camadas: desde roteamento, pacotes e segurança até sua tecnologia de rádio sem fio, sendo isso um enorme diferencial a outras tecnologias sem fio para IoT. Semelhante ao Wi-Fi, o Thread é um padrão aberto que não está vinculado a um fabricante específico, o que minimiza o risco de incompatibilidades (THREAD GROUP, 2024).

O protocolo Thread usa várias tecnologias existentes, incluindo IEEE 802.15.4 para comunicação de rádio, IPv6 e 6LoWPAN para rede e roteamento, e AES-128 para segurança, proporcionando uma rede de malha robusta e eficiente para dispositivos IoT (THREAD GROUP, 2024; IEEE COMPUTER SOCIETY, 2011).

Isto permite a utilização em larga escala e a facilidade de instalação, já que os mesmos dispositivos e tecnologias de rede podem ser usados em todas as regiões.

Narrow Band

A Narrow Band, ou NB-IoT (Narrowband Internet of Things), é uma tecnologia de LPWAN (Low Power Wide Area Network) e tem se apresentado como uma solução eficiente e econômica para redes de IoT (GSMA, 2024).

A tecnologia NB-IoT foi inicialmente apresentada e padronizada em 13 de junho de 2016 pela Third-Generation Partnership Project (3GPP) como uma tecnologia de comunicação sem fio. Já em 2019, a rede de operação foi lançada por 140 operadores em 69 países (3GPP, 2016).

O NB-IoT busca otimizar a banda passante e de energia durante a comunicação, para isso utiliza da infraestrutura de antenas e sistema de comunicação da tecnologia 4G e 5G. Isso é um fato relevante e uma vantagem para a tecnologia, pois está alinhado à estratégia das operadoras de celular de ampliar a cobertura de 5G para que os celulares tenham acesso a redes de altas velocidades, assim, amplia também a cobertura disponível para o protocolo NB-IoT.

Em sua arquitetura existem três principais componentes:

- Estação base (eNb) – coordena a comunicação com dispositivos na área de cobertura;
- Gateway BN-IoT (NW-GW) – é o ponto de acesso entre a rede Nb-IoT e a infraestrutura de rede principal (4G ou 5G)
- Servidor centralizado – gerencia o registro, autenticação e provisionamento de dispositivos, bem como a entrega de dados entre os dispositivos e os aplicativos finais.

Entre as vantagens da tecnologia pode-se destacar:

- Custo baixo do módulo de comunicação;
- Longa duração de bateria de mais de 10 anos;
- Grande quantidade de dispositivos por célula (55.000 dispositivos por célula);
- NB-IoT suporte a comunicação Half-Duplex;

A NB-IoT oferece um potencial significativo para ser uma tecnologia utilizada largamente em cenários de IoT. Por possuir uma boa eficiência espectral, baixo consumo de energia e capacidade de penetração em estruturas densas, o NB-IoT está posicionado como uma solução robusta e econômica para uma variedade de aplicações IoT em diversos setores.

Comparativo entre as tecnologias

A Tabela 1 apresenta as tecnologias sem fio mais comumente utilizadas para soluções IoT.

Tabela 1 - Comparativo protocolos rede sem fio.

Tecnologia	**Zigbee**	**LoRaWAN**	**Thread**	**Narrow Band**
Alcance	Curto	Longo	Curto a Médio	Longo
Consumo de energia	Baixo a Médio	Baixo a Médio	Baixo a Médio	Baixo a Médio
Topologia de rede	Malha	Estrela	Malha	Estrela
Segurança	Média	Alta	Alta	Alta
Velocidade de transmissão	Baixa	Baixa	Média	Baixa
Interoperabilidade	Boa	Baixa	Boa	Boa
Frequência de operação	2,4 GHz	915Mhz, 2,4GHz e 5 GHZ	2,4 GHz	300Hz a 3,4GHz
Aplicações comuns	Automação residencial, industrial, iluminação inteligente	Agricultura inteligente, monitoramento ambiental, rastreamento de ativos	Dispositivos de automação residencial, iluminação, segurança	Monitoramento remoto, sensores industriais, redes de baixa potência

Essa tabela apresenta um destaque para LoRa e NarrowBand em relação às outras tecnologias, quando analisado principalmente os requisitos de baixo consumo de energia e longo alcance.

Segundo Carullo, Montori e Tarchi (2023), a decisão sobre qual tecnologia utilizar é embasada em seu artigo "Long range and low power networks for the internet of things: open issues and comparison between LoRaWAN and NB-IoT", onde fazem uma comparação entre LoRaWAN e NB-IoT em termos de alcance, eficiência energética, custo e outros aspectos importantes para aplicações de IoT, apresentando significativas do LoRaWAN em relação ao NB-IoT.

O primeiro elemento que qualifica o LoRa é o alcance estendido do LoRaWAN, que permite a comunicação por vários quilômetros, mesmo em ambientes urbanos densos, sem a necessidade de repetidores contra um alcance mais limitado e uma possível necessidade maior de células do NarrowBand.

Outro ponto de relevância que traz destaque para o LoRa é o baixo consumo de energia do LoRaWAN, o que é uma vantagem significativa em comparação com o NB-IoT.

Por fim o custo de infraestrutura do LoRa é vantajoso em comparação com o NB-IoT. A implantação de uma rede LoRa requer menos infraestrutura e é menos dispendiosa, tornando-a uma opção mais acessível para implantações em grande escala.

Essas vantagens sugerem que o LoRa pode ser uma escolha mais adequada para certos cenários de aplicação de IIoT.

2.3.3.2 – Protocolos de Comunicação para IoT

Neste trabalho, foram considerados os protocolos de comunicação que apresentaram maior relevância bibliográfica em pesquisas de IoT Industrial. Dentre esses protocolos se destacam: AMQP, MQTT, OPC-UA e Kafka. As características desses protocolos podem ser observadas na Tabela 2.

Tabela 2 – Comparativos de protocolos de comunicação para IoT.

Aspecto/Protocolo	**MQTT**	**AMQP**	**OPC-UA**	**Kafka**
Uso	Comunicação entre dispositivos IoT	Mensagens entre aplicações	Comunicação em sistemas industriais	Mensageria e streaming de eventos
Modelo de comunicação	Publish/subscribe	Publish/subscribe e filas de mensagens	Arquitetura de servidor/cliente	Publish/subscribe e filas de mensagens
Aplicação principal	IoT, telemetria, monitoramento	Mensageria em sistemas distribuídos	Controle de processos industriais	Integração de sistemas, análise de dados
Overhead	Baixo	Moderado	Moderado	Moderado a alto
Latência	Baixa	Variável, dependendo da rede e da carga	Baixa a moderada	Baixa a moderada
Segurança	Possui recursos de segurança, mas requer configuração adequada	Possui recursos de segurança, suporta SSL/TLS	Possui recursos de segurança, suporta SSL/TLS	Pode ser seguro com configuração adequada
Padrões	MQTT v5.0, MQTT-SN	AMQP 1.0, AMQP 0-10, AMQP 0-9-1	OPC UA 1.04	Kafka API, Kafka Streams API
Escalabilidade	Alta	Alta	Alta	Alta

As seções posteriores irão descrever melhor cada um desses protocolos de comunicação.

MQTT

O MQTT foi desenvolvido pela IBM em 1999 por Andy Stanford-Clark e Arlen Nipper para monitorar oleodutos de petróleo através da transmissão de dados por satélite. Foi desenvolvido para ser um protocolo de mensagens leve e eficiente, ideal para comunicações em redes com velocidade de transmissão limitada e dispositivos com recursos limitados como sensores e dispositivos IoT que operam com fonte de energia limitada.

Por ser um protocolo leve, ele possibilita a implementação em dispositivos com restrições significativas de hardware. Além disso, por ser muito adaptável, possibilita a aplicação em diversos cenários e dispositivos e, obviamente, em serviços de IoT (SENGUL; KIRBY, 2021).

A comunicação via MQTT é formada por, principalmente, três elementos: publicadores, inscritos ou assinantes e broker, como exemplificado na Figura 8.

Os publicadores são responsáveis por enviar um certo dado para um servidor Broker, o qual tem a tarefa de redirecionar esta mesma mensagem para os dispositivos assinantes. Para que essa tarefa seja realizada com segurança e sem a mistura de dados, cada mensagem utiliza um tópico para trafegar. Dessa forma, por exemplo, apenas os assinantes que se inscreveram em um determinado tópico podem ter acesso às mensagens enviadas pelo publicador nesse mesmo tópico.

Este modelo permite uma comunicação assíncrona eficiente entre dispositivos, onde os publicadores e inscritos não precisam estar ativos ao mesmo tempo, e a estrutura hierárquica dos tópicos permite uma organização e subscrição individualizada das mensagens.

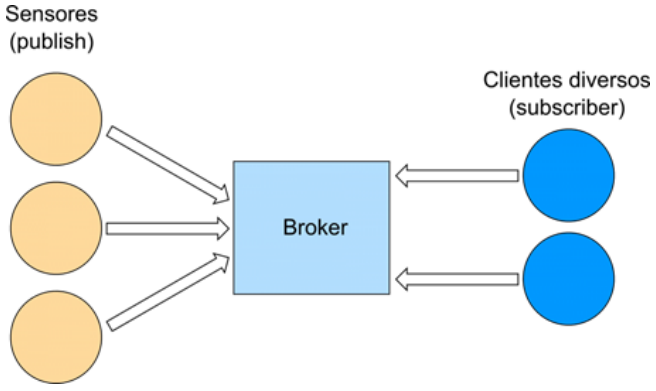


Figura 8 – Esquema de comunicação MQTT.

Outra característica do MQTT é suportar diferentes níveis de qualidade de serviço (QoS) e garantir a entrega das mensagens de acordo com os requisitos de aplicação.

Existem três modos de QoS de entrega de dados:

- QoS0 (no máximo uma vez): é uma entrega de melhor esforço;
- QoS1 (pelo menos uma vez): duplicações podem ocorrer;
- QoS2 (exatamente uma vez): uma entrega confiável.

O broker é o orquestrador desta comunicação e podem existir diversas arquiteturas possíveis, como apresentado na Figura 9. Inclusive, pode ser instalado em um ambiente em nuvem ou em um servidor local. Existe ainda a possibilidade de comunicação entre brokers expandindo ainda mais as possibilidades e capacidade de escalar. Um servidor que suporta a função de Broker pode atuar simultaneamente como publicador e assinante de outro Broker, possibilitando implementações escaláveis e hierárquicas (IMANE; TOMADER; NABIL, 2018).

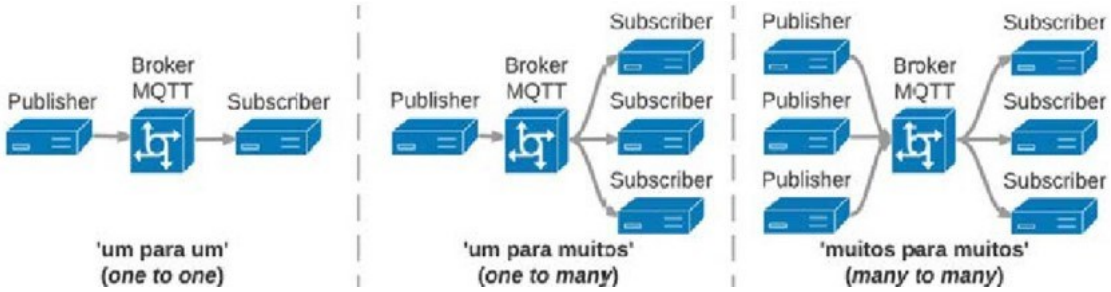


Figura 9 - Exemplos de arquiteturas MQTT.

A versão 5 é a mais recente do MQTT, lançada em 2019, que evoluiu significativamente, inclusive com uso de criptografia para segurança usando o TLS (*Transport Layer Security*) (SENGUL; KIRBY, 2021). O TLS com o uso de “tokens” tem finalidade de conceder permissão para a conexão entre um cliente e o Broker para publicar ou subscrever um tópico específico.

O MQTT apresenta-se como uma capacidade de atender uma enorme gama de cenários de IoT em especial IIoT.

AMQP

O AMQP (Advanced Message Queuing Protocol) é um protocolo de mensageria aberto e padronizado, que oferece uma solução robusta e flexível para comunicação entre sistemas distribuídos. Esse protocolo é regulamentado e mantido pela OASIS (*Organization for the Advancement of Structured Information Standards*), organização sem fins lucrativos que desenvolve e promove padrões abertos para a comunicação entre sistemas.

O foco de utilização do AMQP é em sistemas distribuídos para facilitar a troca de mensagens entre aplicativos e serviços. Neste cenário, muitas vezes a comunicação assíncrona e com confiabilidade são requisitos primordiais que são entregues, graças a características da solução de processamento de eventos e implementação de filas de mensagens.

O modelo de comunicação adotado pelo AMQP utiliza-se de filas de mensagens e exchanges. O exchange, neste contexto, é o componente que atua como ponto de entrada para mensagens. Publicadores publicam mensagens em exchanges e as mensagens são roteadas para as filas especificas, de acordo com as regras de roteamento. Para fechar o ciclo de comunicação, os consumidores se conectam às filas para receber as mensagens. A Figura 10 representa o modelo de comunicação do protocolo.

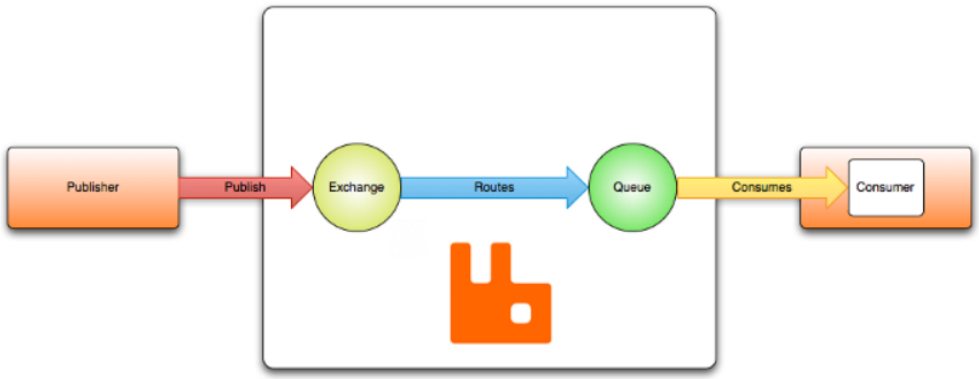


Figura 10 - Digrama de modelo comunicação AMQP (Fonte: https://www.rabbitmq.com/tutorials/amqp-concepts).

Pode-se considerar que os exchanges são os principais elementos deste protocolo e as regras de roteamento os diferenciam em tipos. Eles podem ser subdivididos em:

- Direct Exchange: As mensagens são roteadas para as filas com base em uma chave de roteamento que é especificada pelo publicador. A mensagem é entregue à(s) fila(s), cuja chave de roteamento corresponde exatamente à chave especificada pelo Publisher, como apresentado no diagrama da Figura 11.

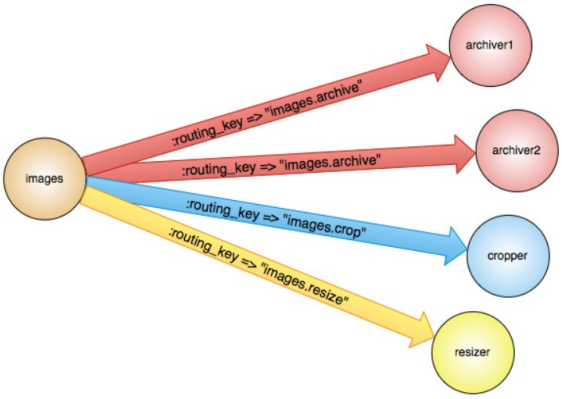


Figura 11 - Direct Exchange Routing (Fonte: https://www.rabbitmq.com/tutorials/amqp-concepts).

- Fanout Exchange: As mensagens são roteadas para todas as filas vinculadas a ela. Ignora qualquer chave de roteamento especificada pelo Publisher e entrega a mensagem a todas as filas vinculadas, como apresentado no diagrama da Figura 12.

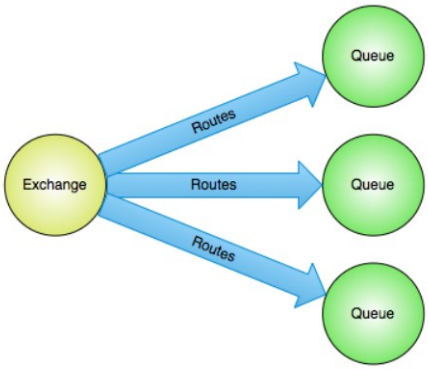


Figura 12 - Fanout Exchange Routing (Fonte: https://www.rabbitmq.com/tutorials/amqp-concepts).

- Topic Exchange: As mensagens são roteadas para filas com base em padrões de chave de roteamento (wildcards) especificados pelo Publisher.
- Headers Exchange: As mensagens são roteadas com base em cabeçalhos (headers) definidos pelo produtor. A exchange examina os cabeçalhos da mensagem e faz correspondência com os critérios de roteamento especificados pelas filas.

Com foco no atendimento de sistemas industriais de alta criticidade e acurácia, requisitos como escalabilidade, overhead, latência e segurança são essenciais e, por isso, serão detalhados no contexto deste protocolo.

O AMQP é altamente escalável, pois foi projetado para suportar sistemas distribuídos de grande escala. Para conseguir essa escalabilidade existem diversos recursos que podem ser utilizados como:

- Replicação e Clustering – distribuição de cargas de trabalho entre vários nós do cluster.
- Particionamento de Filas – Permite que o processamento de mensagens seja distribuído de forma eficiente entre nós.
- Balanceamento de carga dinâmico – Distribuição automática de carga nos nós do cluster com base na carga de trabalho e capacidade de cada nó.
- Escalabilidade horizontal – Capacidade de adicionar novos nós ao cluster para aumentar a capacidade, o processamento e o armazenamento.
- Modelo de assinatura de mensagens – O AMQP permite múltiplos consumidores para uma mesma fila para processar mensagens concorrentemente, facilitando assim a distribuição de carga e paralelismo no processamento das mensagens, o que impacta significativamente a eficiência e desempenho do sistema.

A necessidade de carga de processamento e recursos computacionais, o overhead, é um aspecto importante na decisão de qual protocolo utilizar.

A AMQP possui um overhead moderado, uma vez que possui boa flexibilidade e confiabilidade. Porém, quando comparado a protocolos mais simples como o MQTT, ele é mais oneroso, pois possui uma camada de abstração de roteamento de mensagens e garantia de entrega que necessita de mais poder computacional. (VIDELA e WILLIANS) discorrem que, com boas práticas de design e configuração do sistema, incluindo o uso de exchanges, filas e políticas de roteamento podem garantir escalabilidade e desempenho do sistema minimizando o overhead.

Latência, assim como overhead, podem inviabilizar o uso do protocolo para um tipo de aplicação específica, (Li e Zhang) listam como os principais aspectos que influenciam a latência o processamento no Broker, a latência da rede, processamento no consumidor e congestionamento da rede e do Broker e propõe que, com uso de estratégias e técnicas corretas, o AMQP garante entregas de mensagens confiáveis e com baixa latência em ambientes distribuídos.

Por fim, mas não menos importante, um aspecto essencial de qualquer protocolo de comunicação, em especial para protocolos baseados em mensagens, é a segurança. Garantir integridade, confidencialidade e disponibilidade das mensagens é requisito pétreo em praticamente todos os sistemas industriais.

Maffeis, Mancini e Martinelli, abordam diversas questões relacionadas à segurança em sistemas baseados em AMQP e seus desafios, incluindo autenticação e autorização, criptografia, controle de acesso e auditoria e monitoramento. Os autores afirmam que todos esses desafios são possíveis de serem solucionados, mantendo o protocolo seguro (MAFFEIS; MANCINI; MARTINELLI, 2021).

OPC UA

O OPC (OLE for Process Control) para o meio industrial talvez seja o protocolo mais conhecido e disseminado, mas na grande maioria das indústrias do Brasil ainda vigora o OPC DA (OPC Data Access). Segundo Leitão, Colombo e Jammes, o OPC UA (OPC Unified Architecture) supera as limitações do seu antecessor oferecendo recursos avançados de segurança, escalabilidade e interoperabilidade com uma arquitetura orientada a serviços (SOA) permitindo assim uma comunicação flexível e extensível entre sistemas e dispositivos.

A Figura 13 exemplifica essa arquitetura que pode suportar uma grande variedade de plataformas.

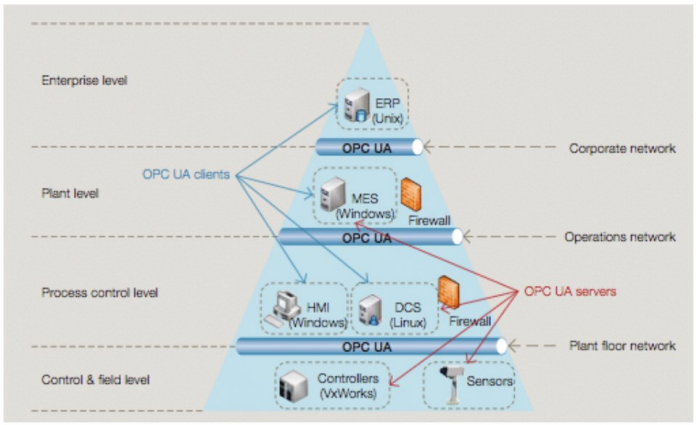


Figura 13 - Aplicação de OPC UA na Pirâmide de Automação.

O OPC UA segue as especificações da OPC Foundation, uma organização sem fins lucrativos que se dedica ao desenvolvimento e promoção de padrões de comunicação para automação industrial. Estas especificações são abertas e padronizadas, a fim de promover uma padronização da utilização do protocolo por fabricantes.

Assim como o OPC DA no passado, o OPC UA tem se tornado amplamente utilizado em ambientes industriais para atividades de controle de processo, monitoramento de ativos, integrações entre sistemas de automação e comunicação com sistemas terceiros.

Apesar de OPC UA ter overhead e ter uma complexidade moderada em sistemas de IoT industriais, o overhead do OPC UA é um elemento importante devido às limitações de recursos, como poder de processamento, memória e velocidade de transmissão de rede, comum em alguns dispositivos.

Essa carga de recursos computacionais para implementar o protocolo estão intimamente ligadas às definições de projeto e configuração, tais como recursos utilizados do protocolo, quantidade de dados transmitidos, segurança e criptografia.

A latência no OPC UA não é o maior desafio do protocolo, que geralmente está entre baixa e média, atendendo os requisitos dos sistemas industriais de tempo real. Porém, assim como em qualquer protocolo, essa latência sofre interferência direta da rede e da carga de trabalho.

Esse protocolo também possui uma característica de ser altamente escalável tanto quando se leva em consideração o número de dispositivos quanto a velocidade de transmissão e tráfego de rede. A escalabilidade está intimamente relacionada ao overhead e à latência, então para garantir um desempenho consistente e confiável de IoT Industrial, deve-se sempre se preocupar em manter baixo o overhead e a latência, enquanto o sistema lida com uma grande quantidade de dispositivos e volume de dados.

O OPC UA oferece recursos avançados de segurança, incluindo autenticação, autorização, criptografia e integridade de dados que devem ser implementados para proteger o sistema contra ameaças cibernéticas e também garante a integridade e confiabilidade das operações de automação industrial.

KAFKA

O Kafka é um produto da Apache que tem como objetivo ser uma plataforma de streaming distribuída, a ferramenta se destaca e é amplamente utilizada em cenário de troca de mensagens e streaming de eventos.

De forma geral, no Apache Kafka, os dados são modelados como fluxos de eventos e organizados em tópicos, algo similar ao protocolo MQTT. Os produtores publicam eventos e enviam dados para essa ação denomina-se "Publish" e os consumidores se inscrevem nos tópicos para receber eventos/dados, essa ação denomina-se "Subscribe", como apresentado na Figura 14.

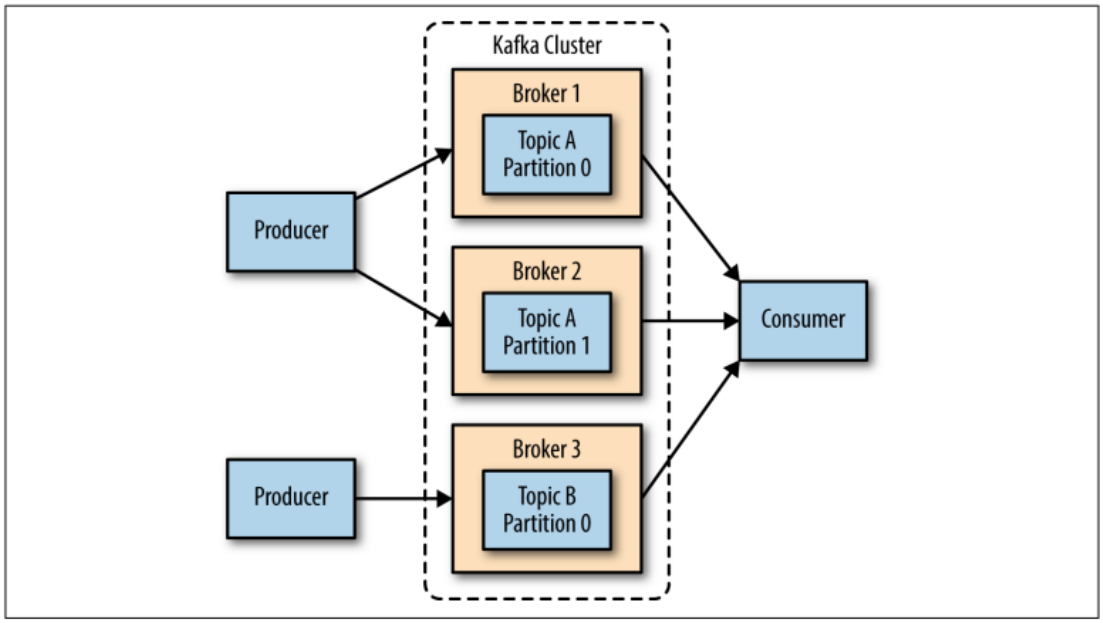


Figura 14 - Cluster Kafka.

A infraestrutura e o modelo de dados do Kafka possibilitam algumas características importantes como escalabilidade, tolerância a falhas e persistência da informação.

Quando ser trata de escalabilidade, a plataforma permite lidar com grandes volumes de dados e suporta uma enorme quantidade de produtores e consumidores de dados. Conforme Narkhede, Shapira e Palino (2017) a escalabilidade flexível do Kafka facilita o gerenciamento de qualquer quantidade de dados. Os usuários podem começar com um único broker como prova de conceito, expandir para um pequeno cluster de desenvolvimento de três brokers, e passar para a produção com um grupo maior de dezenas ou mesmo centenas de brokers, a depender de como a demanda de dados aumente. A Figura 15 apresenta um cluster Kafka com uma estrutura de 2 brokers de uma estrutura de mensagens entre produtores e consumidores.

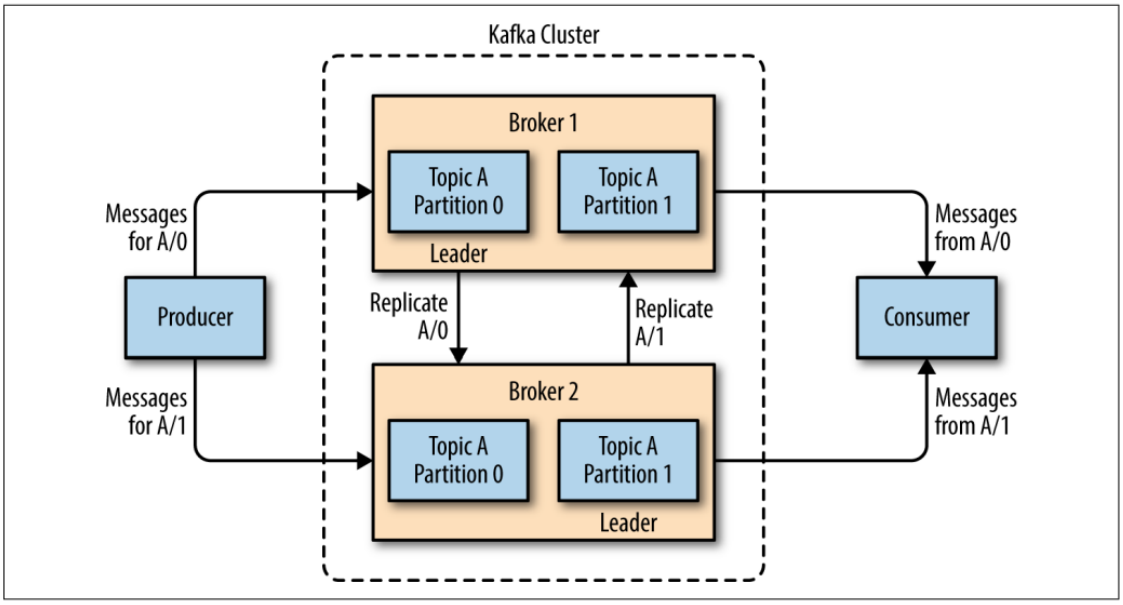


Figura 15 - Replicação de brokers em um Cluster Kafka.

Para a tolerância a falhas e persistência da informação, essas características vêm do aspecto arquitetural da plataforma, ou seja, os dados são replicados e armazenados em cada nó do cluster e podem ser recuperados em caso de falha.

O overhead do Kafka diz respeito à carga de processamento e recursos computacionais necessários para operar. Ele é influenciado pela arquitetura distribuída devido à necessidade de coordenação entre os nós, gerenciamento das partições e replicação dos dados que, apesar de ser um elemento importante para a tolerância de falhas, essa característica impacta o processamento, a quantidade de dados transmitidos e a latência devido à necessidade de sincronização e transferência de dados entre os nós.

A latência no Kafka é o tempo dispensado para que os eventos sejam processados e entregues aos consumidores. Essa latência pode ser influenciada principalmente por: carga de trabalho, configuração do cluster e topologia de rede. A carga de trabalho influencia diretamente, pois o volume de dados e a taxa de eventos a ser direcionado e replicado influencia diretamente no tempo de conclusão de uma transação. A forma como o Kafka é configurado também interfere na latência, pois o número de partições implica em maior replicação e orquestração, assim como quantidade e tamanhos de segmentos de logs. Por fim, a topologia de rede naturalmente cria uma latência entre os nós do cluster que deve ser considerado para cada tipo de aplicação.

2.3.4 – Camada de Aplicação

Esta é a camada superior da arquitetura IoT, onde os dados coletados são processados, analisados e utilizados para gerar valor para os usuários finais. Aqui, os dados podem ser armazenados, analisados em tempo real, visualizados em painéis de controle ou utilizados para alimentar aplicativos específicos. Esta camada pode envolver uma variedade de serviços, como plataformas de IoT na nuvem, sistemas de gerenciamento de dados, aplicativos móveis ou soluções de análise avançada. O objetivo principal é fornecer insights acionáveis e funcionalidades úteis com base nos dados provenientes dos dispositivos IoT.

Para a camada de aplicação temos diversas tecnologias e abordagens que podem ser consideradas, as seções posteriores irão detalhar as tecnologias de contêiners e orquestradores.

Containers

Em sistemas convencionais normalmente é instalado um software em um servidor físico ou virtual, para isso é necessário configurar todas as dependências de bibliotecas, frameworks e atualizações dos sistemas operacionais. Atualmente, nos ambientes de TI, tem ficado cada vez mais comum o uso de uma arquitetura de contêiners.

A Figura 16 exemplifica as principais diferenças de uso de containers em relação ao uso de máquinas virtuais. Uma das principais vantagens é a eficiência no uso de recursos. Containers compartilham o mesmo sistema operacional (OS) do host, o que reduz a sobrecarga e melhora a utilização de recursos em comparação com VMs (Máquinas Virtuais), que requerem um sistema operacional completo para cada instância.

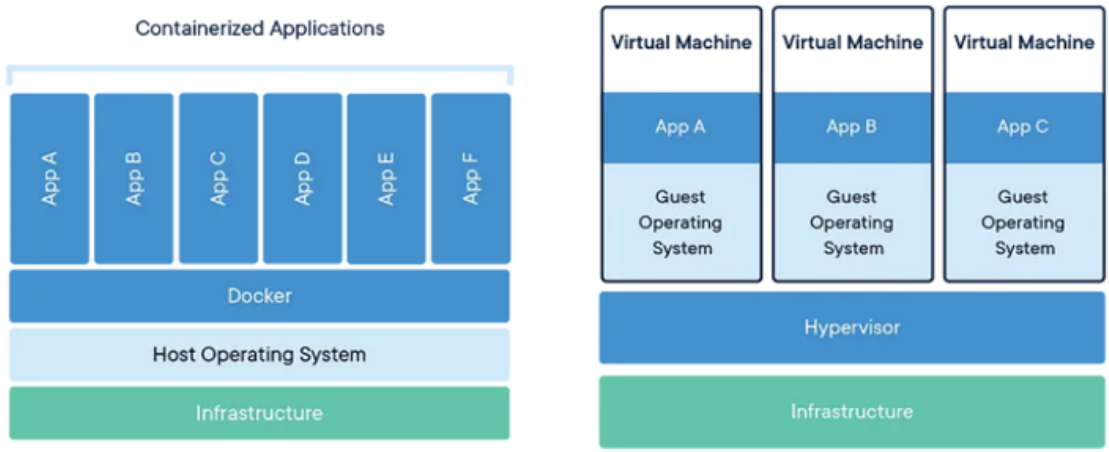


Figura 16 - Containers x VMs

A arquitetura de containers é uma abordagem de virtualização leve que ganhou grande popularidade na última década, impulsionada principalmente pela ascensão de tecnologias como Docker. Docker é um software de código aberto usado para implantar aplicativos dentro de containers virtuais. Um contêiner Docker é um formato de empacotamento que empacota todo o código e dependências de um aplicativo em um formato padrão que permite sua execução rápida e confiável em ambientes de computação. A Figura 17 exemplifica os elementos que compõe a criação de um container Docker. O elemento base é um Docker File que após uma construção (“build”) gera uma imagem Docker e por sua vez roda em um orquestrador formando então o container Docker.

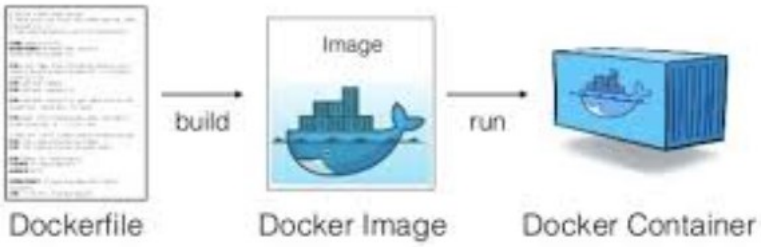


Figura 17 - Elementos Docker

Essa abordagem permite empacotar e distribuir aplicativos, juntamente com suas dependências e configurações, em unidades chamadas containers. Esses containers são isolados uns dos outros e podem ser executados consistentemente em diferentes ambientes, proporcionando portabilidade e eficiência.

Apesar de no ambiente de TI tradicional essa tecnologia tem se tornado cada vez mais difundida e adotada como uma boa prática em sistemas coorporativos, na indústria não é muito difundido. O ambiente industrial por sua natureza é mais conservador e mais lento para adoção de novas tecnologias.

Uma arquitetura de contêiners possui alguns componentes diferentes dos convencionais de sistemas até então massivamente utilizados na indústria. Esses componentes são:

- Container Engine - O componente central que gerencia a criação, execução e destruição de containers. Exemplos: Docker Engine, containerd.
- Imagens de Containers - Pacotes leves que contêm o código do aplicativo, bibliotecas e dependências necessárias. Imagens são usadas para criar containers

em tempo de execução. Cada imagem é composta por camadas, facilitando a reutilização e distribuição eficiente.

- Orquestradores - Ferramentas que automatizam a implantação, o escalonamento e a gestão de containers em larga escala. Exemplos: Kubernetes, Docker Swarm, OpenShift.
- Registries - Repositórios para armazenar e distribuir imagens de containers. Exemplos: Registries públicos (por exemplo, Docker Hub) e privados são comuns.

Entre os benefícios de uma arquitetura de contêiners podemos destacar:

- Isolamento: Cada container é isolado, garantindo que as mudanças em um não afetem outros.
- Portabilidade: Containers podem ser executados de maneira consistente em diferentes ambientes, desde o desenvolvimento até a produção.
- Eficiência: Uso eficiente de recursos, pois os containers compartilham o mesmo kernel do sistema operacional host.
- Escalabilidade: Os orquestradores facilitam a escalabilidade horizontal, permitindo a gestão eficaz de grandes cargas de trabalho.

Quando se trata de uma arquitetura baseada em containers pode-se considerá-la para sistemas “on-premise” (servidores locais) e “on cloud” (sistema em nuvem), porém, independente da infraestrutura selecionada, um ponto comum desta arquitetura é a necessidade de um orquestrador.

Orquestrador

Os orquestradores de containers desempenham um papel fundamental na gestão eficiente de ambientes distribuídos de containers, automatizando tarefas como implantação, escalonamento, balanceamento de carga e monitoramento.

Para entender melhor os aspectos e características dos orquestradores serão descritos alguns papéis deles e o impacto inovador desta tecnologia na arquitetura de sistemas.

Implantação Automatizada:

Os orquestradores automatizam o processo de implantação de containers em clusters de servidores. Eles facilitam a distribuição de containers em diferentes nós do cluster, levando em consideração fatores como recursos disponíveis, requisitos de aplicativos e políticas de tolerância a falhas.

Escalonamento Dinâmico:

Monitoram a carga de trabalho e ajustam automaticamente o número de instâncias de containers conforme necessário. Isso permite que as aplicações se adaptem dinamicamente a picos de tráfego e otimizem a utilização de recursos.

Balanceamento de Carga:

Distribuem o tráfego de rede de forma equitativa entre as instâncias de containers em execução. Isso garante uma utilização eficiente dos recursos e mantém a alta disponibilidade das aplicações.

Gerenciamento de Recursos:

Alocam recursos computacionais, como CPU e memória, de forma eficiente entre os containers. Isso evita conflitos de recursos e maximiza o desempenho do sistema.

Auto recuperação:

Detectam e lidam automaticamente com falhas de containers ou de nós do cluster. Eles reiniciam containers com falha em outros nós disponíveis para garantir a continuidade do serviço.

Gestão de Configuração e Armazenamento de Dados:

Facilitam a gestão centralizada da configuração de containers e o armazenamento persistente de dados. Isso inclui o provisionamento de volumes de armazenamento e a configuração dinâmica de variáveis de ambiente.

Monitoramento e Logging:

Coletam métricas de desempenho, logs e eventos de containers e nós do cluster. Essas informações são essenciais para monitorar a saúde das aplicações e diagnosticar problemas de operação.

Segurança:

Implementam políticas de segurança, como isolamento de rede e controle de acesso, para proteger os containers e os dados que eles manipulam. Isso ajuda a mitigar riscos de segurança e conformidade regulatória.

Integração com Ecossistema de Ferramentas:

Integram-se com outras ferramentas e serviços, como sistemas de CI (Integração Contínua) e CD (Implantação Contínua), registros de imagens de containers e serviços de monitoramento. Isso proporciona uma experiência de desenvolvimento e operações mais integrada e eficiente.

Extensibilidade e Customização:

Oferecem APIs (Interfaces de Programação de Aplicativos) e mecanismos de extensibilidade que permitem estender e personalizar o comportamento do orquestrador para atender a requisitos específicos de negócios e de aplicações.

Esses papéis destacam a importância dos orquestradores de containers na simplificação e na automação de tarefas relacionadas à gestão de ambientes de containers, permitindo que as organizações desenvolvam e operem aplicações de forma mais eficiente e escalável.

As seções posteriores irão descrever algumas características dos principais orquestradores de mercado.

Kubernetes

Kubernetes é um orquestrador de containers de código aberto desenvolvido pelo Google, agora mantido pela CNCF (“*Cloud Native Computing Foundation”*). Ele oferece recursos poderosos para automação de implantação, dimensionamento e operações de containers.

Entre os benefícios desta ferramenta estão:

- Automatização avançada de implantações e escalonamento.
- Gerenciamento eficiente de recursos.
- Alta disponibilidade e tolerância a falhas.
- Ecossistema robusto de ferramentas e integrações.

Docker Swarm

Docker Swarm é a solução de orquestração de containers integrada ao ecossistema Docker. Ele simplifica a implantação e o gerenciamento de clusters de containers Docker.

Entre os benefícios desta ferramenta estão:

- Fácil de configurar e usar, especialmente para usuários familiarizados com Docker.
- Integração perfeita com o Docker Engine e outras ferramentas do ecossistema Docker.
- Escalabilidade horizontal e alta disponibilidade.

Apache Mesos

Apache Mesos é um sistema de gerenciamento de recursos distribuídos que oferece recursos de orquestração de containers, entre outras funcionalidades. Ele fornece uma abstração de recursos de baixo nível para aplicativos distribuídos.

Entre os benefícios desta ferramenta estão:

- Utilização eficiente de recursos em clusters heterogêneos.
- Suporte para várias estruturas de aplicativos, incluindo containers Docker.
- Escalabilidade e tolerância a falhas.

Amazon ECS (Elastic Container Service)

Amazon ECS é um serviço de orquestração de containers oferecido pela Amazon Web Services (AWS). Ele simplifica a execução de containers Docker em infraestrutura gerenciada pela AWS.

Entre os benefícios desta ferramenta estão:

- Integração nativa com a plataforma AWS, incluindo ferramentas de monitoramento, escalabilidade e segurança.
- Facilidade de uso e configuração.
- Escalabilidade e alta disponibilidade.

Com esses conceitos sobre arquitetura de containers, um novo conceito se torna realidade para configuração de arquitetura, a infraestrutura baseada em código.

2.3.4.3 – Infraestrutura baseada em código

A infraestrutura baseada em código (Infrastructure as Code - IaC) é uma abordagem para gerenciar e provisionar infraestrutura de TI usando código, em vez de

processos manuais ou interfaces gráficas. Ela permite que as equipes definam, implementem e gerenciem a infraestrutura de forma consistente e escalável, tratando a infraestrutura como código de software.

Esse código envolve a descrição da infraestrutura desejada em arquivos de texto, geralmente usando uma linguagem específica ou ferramenta, como Terraform, AWS Cloud Formation, Ansible ou Chef. Esses arquivos descrevem os recursos de infraestrutura necessários, como servidores, redes, armazenamento e configurações de segurança, de forma declarativa.

Essa tratativa possui alguns benefícios como consistência, rastreabilidade, automatização e agilidade.

Consistência: Possibilita que a infraestrutura seja definida e replicada de forma consistente em diferentes ambientes, como desenvolvimento, teste e produção.

Rastreabilidade: Facilita o controle de versão e o rastreamento de mudanças na infraestrutura, proporcionando transparência e auditabilidade.

Automatização: Automatiza a implantação e o gerenciamento da infraestrutura, reduzindo erros humanos e aumentando a eficiência operacional.

Agilidade: Permite que as equipes de desenvolvimento e operações colaborem de forma mais estreita e ágil, acelerando o ciclo de vida do desenvolvimento e a entrega de software.

Usar esse tipo de abordagem possibilita trazer ao meio industrial inovações, ganho de eficiência, performance na implantação e comissionamento de sistemas industriais**.**

A aplicação desta arquitetura e sistema na indústria muitas vezes é aplicado a disciplina de manutenção de ativos industriais e o contexto de manutenção é detalhado na próxima seção.

2.4 - Manutenção

A manutenção é uma disciplina industrial essencial e pode tornar uma empresa mais ou menos competitiva no mercado industrial. Existem diversas abordagens de manutenção. Neste trabalho consideramos as definições baseadas na NBR 5462:1994.

Segundo FLOYD (FLOYD, 1998), mesmo que um equipamento seja concebido da maneira mais eficiente possível, sua confiabilidade e seu desempenho, ao longo do ciclo de vida, estão associados à qualidade de operação e eficiência de manutenção.

Assim, a manutenção, realizada de maneira otimizada, leva a menores custos operacionais.

A manutenção tem o papel de minimizar o risco quantificado grave de segurança ambiental nos ativos ou, ainda, a perda de produção que pode reduzir a viabilidade e rentabilidade de uma organização, tanto a curto e longo prazo, e fazê-lo com o menor custo total (NARAYAN, 2012).

A principal responsabilidade da manutenção é oferecer um serviço que permita a uma organização atingir seus objetivos (HAROUN; DUFFUAA, 2009).

Existem alguns tipos de manutenção, mas de forma geral eles subdividem em reativas e proativas, onde as reativas são manutenção após uma avaria ou defeito, classificadas como corretivas, e manutenção proativa onde se enquadram a preventiva e preditiva. A manutenção preditiva é como o nome sugere, as ações são proativas, não havendo a espera que o equipamento falhe antes de iniciar as operações de manutenção (KOTHAMASU; HUANG; VERDUIN, 2009). A Figura 18 ilustra os tipos de manutenção.

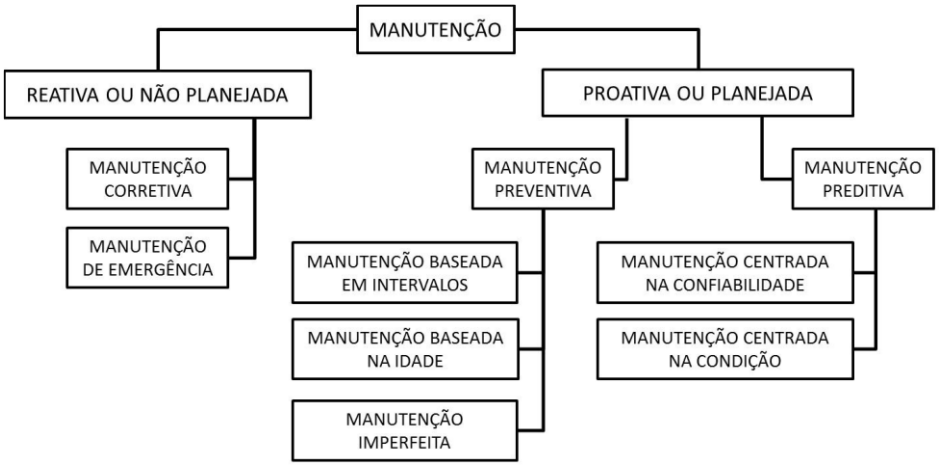


Figura 18 - Filosofias de Manutenção (Fonte: KOTHAMASU, HUANG e VERDUIN (2009)).

Os tipos de manutenção:

• **Manutenção preventiva:** Manutenção efetuada em intervalos predeterminados, ou de acordo com critérios prescritos, destinada a reduzir a probabilidade de falha ou a degradação do funcionamento de um item. A inspeção é realizada antes da falha funcional que, por sua vez, é realizada com base na idade de serviço e no tempo previsto de falha. Assim, se a estimativa é pessimista, a manutenção é feita mesmo quando o equipamento está em perfeito estado de funcionamento. As revisões ou substituições programadas, as tarefas de detecção de falhas baseadas no tempo ou as tarefas em condição fazem parte do programa de manutenção preventiva (NARAYAN, 2012).

• **Manutenção corretiva:** Manutenção efetuada após a ocorrência de uma pane, destinada a recolocar um item em condições de executar uma função requerida. Em muitas organizações, a manutenção corretiva é chamada de manutenção de reparo, quando a orientação para correção de deficiências ocorre somente depois que ele falhou ou parou de funcionar (GULATI, 2012).

• **Manutenção preditiva:** Manutenção controlada que permite garantir uma qualidade de serviço desejada, com base na aplicação sistemática de técnicas de análise, utilizando-se meios de supervisão centralizados ou amostragem para reduzir ao mínimo a manutenção preventiva e corretiva.

Entre as manutenções citadas a corretiva é a que possui o maior custo e tem maior impacto na produção e no negócio. Apesar de melhor que a corretiva, a preventiva, dificulta a otimização do uso do equipamento, pois utiliza de parâmetros como horímetros para disparar o evento de manutenção. A manutenção preditiva é aquela baseada em informações do equipamento e limites de aceitação. Uma técnica muito utilizada na indústria hoje é a manutenção baseada em condições - CBM ("*Condition-based maintenance"*), mas, para isso, é necessária uma instrumentação específica ou um conhecimento muito bom do processo e do funcionamento do equipamento.

A CBM, assim como a manutenção preventiva, tem o objetivo de prevenir quebras e paradas de equipamentos antes que elas ocorram. Porém, com a CBM pode-se otimizar o processo de manutenção e utilização do equipamento, uma vez que as anomalias e degradações ocorrem de forma gradual e não instantaneamente. Com isso, é possível postergar ou anteceder a manutenção baseada no comportamento do equipamento e não por horas de utilização. Diferente da corretiva e da preventiva, a CBM não foca na identificação de falhas e diagnóstico de componentes, mas no monitoramento da degradação e predição de falhas do equipamento (SHIN; JUN, 2015).
Atualmente existem 2 abordagens principais para uso de CBM nas indústrias:

- Utilização de sensores que medem vibração de equipamentos. Com o decorrer do tempo, a deterioração do equipamento aumenta e o padrão de vibração muda, isso pode ser detectado por uma análise de "*Fast Fourier Transform*" (FFT) da vibração de um determinado componente. O ponto que é uma instrumentação específica e alto custo de soluções de mercado acabam limitando o uso para equipamentos que possuem maior custo associado, como por exemplo, moinhos e britadores.

- A utilização de CBM tradicional, onde são criadas regras e notificações para avaliar limites de algumas variáveis associadas ao equipamento, leva à necessidade de conhecimento do processo ou equipamento, impossibilitando escalar a solução.

Tendo em vista que qualquer problema em equipamentos industriais pode resultar em grandes perdas, a CBM é um método bastante atrativo para as indústrias em geral, mas que pode ser ainda mais efetivo e disseminado na indústria de processos se for extrapolada para técnicas de *Data Science* e *Machine Learning*, como proposto pelo projeto.

2.5 - Análise de Dados

Para tratar predição existem na literatura diversos algoritmos possíveis de serem aplicados e combinados para buscar uma melhor capacidade preditiva do modelo.

Os subitens posteriores desta seção descrevem conceitualmente alguns algoritmos e modelos de análise de dados que podem ser aplicados ao universo do problema de predição para bombas industriais. De forma geral estes algoritmos podem ser classificados em tipo de tarefas que são projetados para resolver, sendo elas: regressão, classificação e agrupamento.

2.5.2 – Regressão

Os algoritmos de regressão são técnicas estatísticas utilizadas para prever ou estimar um valor numérico com base em um conjunto de variáveis independentes. Eles são amplamente utilizados em análise de dados, aprendizado de máquina e estatística para resolver problemas de previsão e modelagem.

Existem vários algoritmos de regressão disponíveis, cada um com suas próprias suposições e características. A seguir serão descritos os algoritmos mais amplamente utilizados.

Regressão Linear: A regressão linear é uma técnica básica de regressão que assume uma relação linear entre a variável dependente e as variáveis independentes. O algoritmo busca encontrar a melhor linha reta que se ajusta aos dados, minimizando a soma dos erros quadrados entre as previsões e os valores reais. (WEISBERG, 2005).

Regressão de Árvore de Decisão: A regressão de árvore de decisão é um algoritmo de aprendizado de máquina que segmenta o espaço de entrada em regiões

retangulares e, em cada região, estima um valor para a variável dependente. Ele constrói uma árvore de decisão hierárquica, onde cada nó interno representa uma regra de divisão baseada em uma variável preditora e cada folha representa uma estimativa de valor. (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

Regressão de Floresta Aleatória: A regressão de floresta aleatória é uma extensão da regressão de árvore de decisão. Nesse algoritmo, várias árvores de decisão são construídas com diferentes amostras de dados e variáveis preditoras. As previsões são feitas pela média das previsões de todas as árvores. As florestas aleatórias são conhecidas por sua capacidade de lidar com dados de alta dimensionalidade e lidar com sobreajuste (BREIMAN, 2001).

Existem muitos outros algoritmos de regressão, como regressão logística, regressão de suporte de vetores, regressão de mínimos quadrados parciais, regressão com redes neurais, entre outros. A escolha do algoritmo depende do problema específico e das características dos dados.

2.5.1 – Classificação

A classificação é uma técnica de aprendizado de máquina usada para categorizar dados em classes predefinidas. Ela é fundamental em áreas como reconhecimento de imagem e análise de texto, onde o modelo é treinado com dados rotulados para identificar padrões e distinguir entre diferentes categorias. Após o treinamento, o modelo pode prever a classe de novos dados, auxiliando na tomada de decisões automatizadas. Um dos algoritmos classificação muito comum de ser utilizado nestes cenários é o algoritmo de árvore de decisão.

O algoritmo de Árvore de Decisão é um método de aprendizado de máquina supervisionado utilizado para resolver problemas de classificação e regressão. Ele constrói um modelo em forma de árvore, na qual cada nó interno representa uma decisão baseada em um atributo e cada folha representa uma classe ou um valor numérico.

A construção da árvore de decisão é baseada na divisão recursiva do conjunto de dados de treinamento, utilizando medidas de impureza ou ganho de informação para selecionar o atributo mais relevante em cada etapa. Os algoritmos mais comuns para a construção de árvores de decisão são o ID3 (*Iterative Dichotomiser 3*), o C4.5 e o CART (*Classification and Regression Trees*).

No ID3, por exemplo, as decisões são feitas com base na entropia de um conjunto de dados. A entropia mede a impureza de uma coleção de exemplos e busca minimizá-la ao dividir o conjunto de dados em subconjuntos cada vez mais puros. O

algoritmo ID3 usa a entropia e o ganho de informação para selecionar o melhor atributo a ser utilizado na divisão em cada nó da árvore.

Já o algoritmo C4.5 é uma extensão do ID3 que permite o tratamento de atributos com valores contínuos, além de lidar com dados ausentes. Ele utiliza a razão de ganho em vez do ganho de informação para selecionar os atributos. A razão de ganho normaliza o ganho de informação pelo valor intrínseco da divisão, o que evita o viés em relação a atributos com muitos valores possíveis.

O CART é um algoritmo que pode ser usado tanto para problemas de classificação quanto para regressão. Ele constrói árvores binárias, dividindo recursivamente o conjunto de dados em duas partes com base em uma regra de divisão, como um limite de valor para um atributo contínuo. O CART utiliza o critério de Gini para medir a impureza de um conjunto de exemplos e busca minimizá-lo ao fazer as divisões na árvore. O critério de Gini é uma medida estatística da dispersão que varia de 0 a 1, onde 0 indica perfeita igualdade e 1 perfeita desigualdade.

Existem outras variações e extensões do algoritmo de Árvore de Decisão, como o Random Forest, Gradient Boosting e XGBoost, que combinam várias árvores para obter melhores resultados.

2.5.3 – Agrupamento

Agrupamento é uma técnica de aprendizado não supervisionado que organiza dados em grupos com base em semelhanças, sem rótulos pré-definidos. É usado para identificar padrões e estruturas naturais nos dados, como em segmentação de mercado e análise de padrões. O KNN é um típico algoritmo de agrupamento.

O algoritmo KNN (*k-nearest neighbors*) é um método de aprendizado de máquina supervisionado usado para classificação e regressão. Nesse algoritmo, a previsão de um novo exemplo é baseada na proximidade com os exemplos de treinamento mais próximos. O KNN calcula a distância entre o exemplo de teste e todos os exemplos de treinamento, e os k exemplos mais próximos são usados para prever o rótulo ou valor da variável dependente. O pseudo-código a seguir exemplifica o funcionamento do algoritmo.

```
Algoritmo K-Nearest Neighbors (KNN):

Entrada:
    - Dados de treinamento: T = {(x1, y1), (x2, y2), ..., (xn, yn)}
    - Novo ponto de dados: x
    - Número de vizinhos próximos: K

Saída:
    - Classe ou valor previsto para o novo ponto de dados

Passos:

1. Para cada ponto de dados (xi, yi) em T:
    1.1. Calcule a distância entre x e xi
    1.2. Armazene a distância e o rótulo yi

2. Ordene as distâncias calculadas em ordem crescente

3. Selecione os K pontos mais próximos (menores distâncias)

4. Para classificação:
    4.1. Conte o número de vezes que cada classe aparece entre os K vizinhos mais próximos
    4.2. Atribua a classe mais comum ao novo ponto de dados

   Para regressão:
    4.1. Calcule a média dos valores alvo dos K vizinhos mais próximos
    4.2. Atribua a média ao novo ponto de dados

5. Retorne a classe ou valor previsto
```

Explicação dos Passos

1. **Calcular Distâncias**: Para cada ponto de dados no conjunto de treinamento, o algoritmo calcula a distância entre o ponto de teste e cada ponto de treinamento. A distância comumente usada é a distância euclidiana.
2. **Ordenar Distâncias**: Os pontos de dados são então ordenados com base na distância ao ponto de teste.
3. **Selecionar K Vizinhos Mais Próximos**: Os K pontos de dados mais próximos (ou com as menores distâncias) são selecionados.
4. **Determinar a Classe ou Valor Previsto**:
    - **Classificação**: O rótulo mais comum entre os K vizinhos é escolhido como a previsão.
    - **Regressão**: A média dos valores alvo dos K vizinhos é usada como a previsão.

O KNN é uma abordagem intuitiva e poderosa, mas sua eficácia depende de uma boa escolha dos parâmetros e de um manejo cuidadoso dos dados para agrupar conjunto de dados por classes ou valores previstos.

3 - Metodologia

A metodologia proposta para o projeto está baseada em uma sequência de etapas para o desenvolvimento da solução que são intrínsecas à arquitetura definida. As etapas que compõem o projeto e objeto deste trabalho estão descritas nos itens a seguir.

Definição da Arquitetura

Este item destina-se a definição de arquitetura da solução. Esta definição é uma importante etapa que pode influenciar as definições das etapas posteriores e consequentemente, adoção de uma tecnologia ou outra.

Levantamento e Aquisição de Hardwares

Este item destina-se ao levantamento dos hardwares e sensores que irão compor a arquitetura do projeto. Esta etapa do projeto é composta por dois passos principais: especificação e aquisição. A aquisição dos itens foi baseada nas especificações, menor custo e prazo de entrega.

Prototipação

Todo o processo de montagem do projeto baseia-se no conceito de prototipagem rápida, onde será utilizado um elemento a ser monitorado (ativo), no caso um motor de teste, e serão utilizados elementos de eletrônica de prototipagem rápida e o elemento de processamento. Após testes e validação, o circuito será projetado e produzido em uma placa de circuito impresso.

Ambiente de testes

Para fins de testes e validação do funcionamento do nosso circuito de aquisição de dados e comunicação, utilizaremos um motor de uso residencial que possamos avaliar algumas variáveis.

Definição, projeto e desenvolvimento do hardware

Na definição do hardware, além de atender os requisitos como baixo custo e consumo de energia, é importante que as placas de dispositivo final e gateway sejam

idênticas. Para isso, ambas terão os bornes de interligação dos sensores digitais e analógicos, mas somente o dispositivo final será utilizado para este fim. A utilização do mesmo hardware traz ao projeto uma flexibilidade na utilização e padronização, que irá facilitar o processo de produtização deste elemento.

Desenvolvimento de Software Embarcado

Existem dois elementos na arquitetura do projeto que irão possuir software embarcado, o dispositivo final e o gateway, porém os dois possuem a mesma configuração de hardware. O dispositivo será responsável por duas funcionalidades principais: coleta de dados dos sensores e envio dos dados para o gateway através de uma comunicação sem fio.

O Gateway será responsável por duas funcionalidades principais: recebimento das mensagens e envio das mensagens para o ambiente de dados (servidor).

Configuração do servidor

Nesta etapa será configurado um servidor na nuvem para ser o servidor de aplicações e nele instalado e configurado uma aplicação orquestradora das mensagens que será o broker desta comunicação. O broker MQTT que será utilizado no projeto inicialmente será o Eclipse Mosquitto.

Configuração do cliente MQTT

Nesta etapa será configurado um cliente MQTT que irá subscrever as mensagens do tópico para testes funcionais da solução e validar o recebimento das mensagens dos ativos monitorados.

Extração dos dados de sistemas legados

No caso de instrumentação já existente e integrado aos sistemas de automação industrial ou outras bases de dados, será feita uma extração de dados e enviado para a base de dados da solução na nuvem. Essa etapa é importante para fins de validação, pois servirá como base de testes para validação da capacidade da infraestrutura de fornecer dados ao ambiente analítico.

Criação da Base de Dados

Nesta etapa será criada e configurada uma base de dados em um serviço de banco da nuvem para armazenar estas informações dos ativos monitorados. A base de dados do sistema será uma base temporal com a capacidade de armazenar dados com informações de tag, valor, timestamp e demais itens necessários para contextualizar a informação.

Driver do Assinante MQTT

Nesta etapa será projetado e desenvolvido um aplicativo (driver) que irá subscrever os tópicos MQTT dos ativos monitorados e inserir os dados recebidos na base de dados.

Armazenamento dos dados

Os dados serão armazenados em ambiente de nuvem e com a capacidade escalável e com boa performance para armazenar e recuperar dados temporais. Nesta etapa cria-se a condição de ter uma base de dados de qualidade, com a possibilidade de fornecê-los ao ambiente analítico.

Apresentação dos dados

Os usuários do sistema poderão consumir as informações armazenadas dos ativos monitorados e ter capacidade de, através de uma tela, analisar o comportamento das bombas. É importante que nessa etapa a usabilidade e capacidade de navegar por informações importantes dos dados coletados sejam considerados requisitos primordiais para a solução.

Inicialmente, foi pensado em dashboards em portal web que irão dar ao analista a capacidade de navegar pelas variáveis dos ativos monitorados e, também, uma tendência desta saúde de forma prática e acessível.

4 – Desenvolvimento do Sistema de Monitoramento de Bombas Industriais Inteligente

Esta seção destina-se à aplicação do conhecimento obtido no referencial teórico e as possíveis evoluções aplicadas às etapas e áreas de conhecimentos de cada camada da solução de IoT Industrial para o desenvolvimento do sistema de monitoramento de bombas industriais inteligente.

4.1 - Arquitetura

A definição de melhor arquitetura IIoT para a solução baseou-se nas melhores práticas descritas na bibliografia e muitos aspectos associados a camadas bem distintas da aplicação.

As camadas de dispositivo, de borda e de nuvem têm caraterísticas funcionais importantes de serem consideradas, conforme descrito em HAFEEZ. Essas camadas são também caracterizadas comumente como camada de dispositivo, camada de comunicação e camada de aplicação.

Para HAFEEZ, um papel importante da camada de borda (edge) é ser capaz de trazer uma redução dos dados, minimizando os custos de comunicação. A Figura 19 exemplifica conceitualmente as camadas da arquitetura e onde as funções estão alocadas.

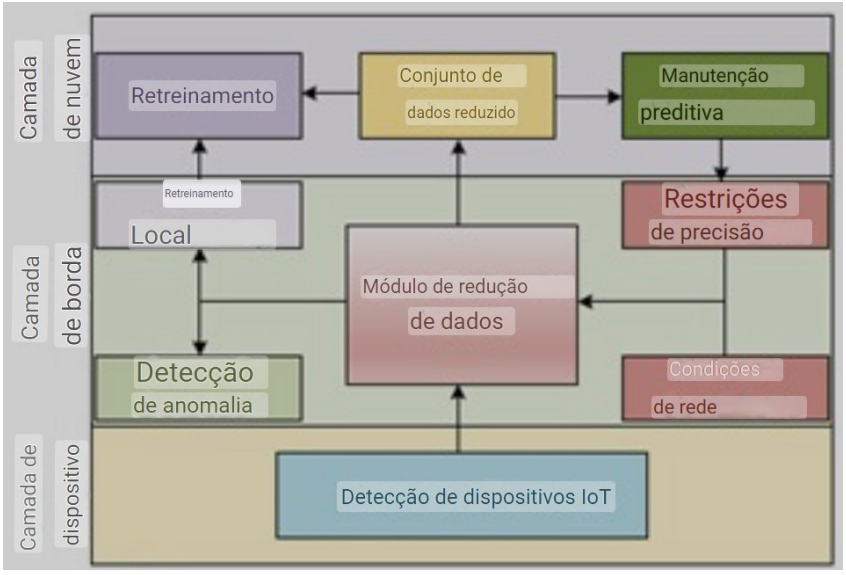


Figura 19 - Arquitetura referência HAFEEZ.

Muitas das funções do modelo proposto por HAFEEZ foi considerado na proposição da solução, porém algumas delas foram consideradas na camada da aplicação, a Figura 20 apresenta a arquitetura da solução.

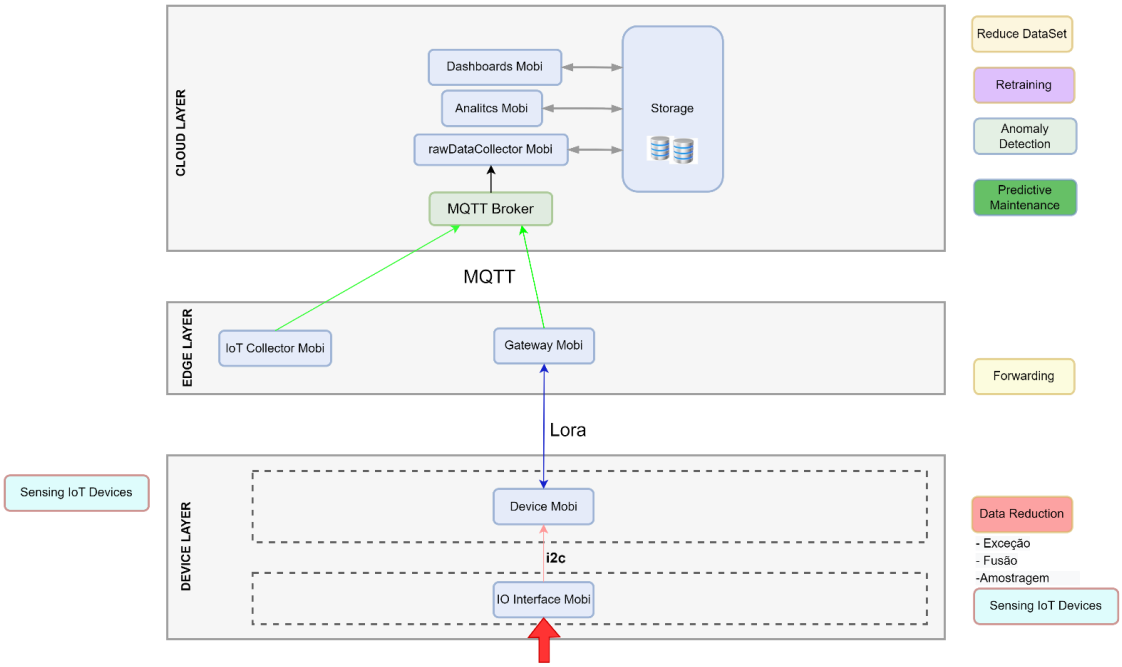


Figura 20 -Arquitetura da Solução (Fonte: Autoria Própria).

No caso da arquitetura da solução, a redução de dados está localizada na camada de dispositivo na etapa de exceção de dados. A camada de borda se encarrega de ser o orquestrador do autogerenciamento e sincronismo dos dispositivos e principalmente, disponibilizar os dados de forma segura para a camada de nuvem.

Na camada de aplicação, que é em nuvem, é onde é feita a tratativa do fluxo de armazenamento dos dados, desde a preparação dos dados, segurança até visualização e disponibilização desses dados para um ambiente analítico.

4.2 – Camada de Dispositivo

Essa camada é composta por dispositivos físicos da arquitetura da solução, onde o dispositivo final possui sensores que estão medindo as variáveis das bombas industriais e transmitindo para as camadas superiores até que os dados sejam armazenados em nuvem.

Portanto, faz parte desta camada os hardwares desenvolvidos (dispositivo final e gateway) e seus respectivos firmwares para aquisição e encaminhamento dos dados das bombas.

Além dos aspectos construtivos da solução eletrônica, como o desenvolvimento do protótipo e a seleção das tecnologias de comunicação, há uma considerável ênfase

na conectividade e na escolha da tecnologia adequada para reduzir lacunas de informação entre as diferentes ilhas de dados. Também se destaca a preocupação com o firmware, que lida com questões de segurança da informação, autogerenciamento e otimização do fluxo de dados.

As seções posteriores trazem informações sobre as decisões de projeto em relação à camada de dispositivo e aplicação prática, culminando em um protótipo.

4.2.1 – Conectividade

Existem diversas tecnologias de comunicação que podem ser utilizadas em cenários de IIoT. Com o intuito de atender os requisitos dos cenários onde a solução poderá ser aplicada, foi feita uma criteriosa comparação entre elementos de hardware da solução para compor o dispositivo final e o gateway.

Pensando no microcontrolador do sistema (elemento computacional). ele deveria possuir um baixo custo, possibilitar intercambiar entre dispositivo final e gateway e atender os requisitos de eletricidade e ambiente. A tabela 3 apresenta os elementos computacionais e o mais aderente à solução proposta foi o ESP-WROOM_32

Tabela 3 - Comparação de controladores (Fonte: Autoria Própria).

Microcontrolador/ Características	ESP-WROOM_32 (Node MCU)	ESP-8266 (Node MCU)	ESP32 LoRaWan	Arduino Uno	Arduino Nano
Custo	R$ 70,00	R$ 30,00	R$ 220,00	R$ 59,00	R$ 30,00
Bluetooth	Contém	Não contém	Contém	Não contém	Não contém
LoRa	Não contém	Não contém	Contém	Não contém	Não contém
Wifi	Contém	Contém	Contém	Não contém	Não contém
Conector do Módulo	Micro-USB	Micro-USB	Micro-USB	USB (tipo B)	Mini-USB
Tensão	2,7 a 3,6 V	4.5 a 9 V	2,7 a 3,6 V	7 a 12V (Recomendado)	7 a 12V (Recomendado)
Nível lógico	3.3V	5V	3.3V	5V	5V
Temperatura de trabalho	-40 a 85 ° C	-40 a 125 ° C	-40 a 85 ° C	-40 a 85 ° C	-40 a 85 ° C
Corrente de consumo (máx)	500mA	200mA	350mA	200mA	19mA
GPIO (Digitais)	25	11	36	14	14
GPIO (Analógicas)	15	1	18	6	8

Para comunicação de longa distância entre o dispositivo final e o gateway, era preciso uma solução sem fio de longa distância e com baixo custo de implementação. Para isso, foi selecionado a tecnologia LoRa. Porém, além da tecnologia, devemos avaliar qual equipamento de mercado será utilizado na solução e um dos requisitos é que ele seja homologado pela Anatel. A tabela 4 e a tabela 5 apresentam alguns dos principais módulos LoRa e uma comparação segundo os critérios definidos para a aplicação, a saber: custo e homologação pela Anatel.

Tabela 4 - Comparativo de preços módulos LoRa.

Modelo	Homologação	Preço médio (R$)
RN2903A	SUSPENSO EM 03/03/2022	88
eLR100-UL-00	Valido até 19/03/2023	64 (Produto indisponível na maioria dos casos)
E32-915T30D	Valido até 23/09/2024	100
RFM95W	Valido até 26/12/2024	75
RFM95W-915S2	Valido até 18/06/2023	75
E220-900T30D	Valido até 14/02/2023	50

Tabela 5 - Módulos LoRa – Regulamentação.

Nome do Fabricante	Modelo	Tipo Produto	Número de Homologação	Certificado de Conformidade Técnica	Data do Certificado de Conformidade Técnica	Data de Validade CCT	CNPJ do Solicitante	Nome do Solicitante	Situação Requerimento Anatel
Microchip Technology Inc.	RN2903A	Transceptor de Radiação Restrita	8021908759	NCC 16733/19	21/02/2019	21/02/2021	61.549.010.000.175	APLICACOES ELETRONICAS ARTIMAR LTDA.	Homologação Suspensa
Shenzhen Hope Microelectronics Co. - Ltd.	RFM95W-915S2	Transceptor de Radiação Restrita	46221905508	29419	19/06/2019	18/06/2023	30.221.647.000.104	Prediza Tecnologia da Informação Ltda	Homologação Emitida
eWBM Corporation Ltd.	eLR100-UL-00	Transceptor de Radiação Restrita	68691912482	118603	13/09/2019	13/09/2023	11.576.445.000.130	Smart Modular Tech do Brasil Ind e Com de Comp Ltd	Homologação Emitida
Microchip Technology Inc.	RN2903A	Transceptor de Radiação Restrita	69982108759	OCP 49921	24/05/2021	23/05/2025	6.084.766.000.188	RSY Hardware e Software EIRELI	Homologação Emitida
Chengdu Ebyte Electronic Technology Co., Ltd.	E220-900T30D	Transceptor de Radiação Restrita	1,85912E+11	NCC 22864/21	14/12/2021	14/12/2023	3.021.334.000.130	FOCKINK INDUSTRIAS ELETRICAS LTDA	Homologação Emitida
Shenzhen Hope Microelectronics Co., Ltd.	RFM95W	Transceptor de Radiação Restrita	61642005129	120581	20/05/2020	20/05/2022	45.561.404.000.192	Superior Industries do Brasil LTDA	Homologação Suspensa
Shenzhen Hope Microelectronics Co. - Ltd.	RFM95W	Transceptor de Radiação Restrita	52261605508	ICC 08.030/2022.1	14/12/2022	26/12/2024	23.395.686.000.162	SMART COMPOSE COMERCIAL LTDA - ME	Homologação Emitida

O modulo selecionado, que atente aos requisitos da solução em relação ao custo e à conformidade com a legislação brasileira, é o RFM95W.

4.2.2 – Prototipação

O dispositivo final e o Gateway da solução devem ser validados quanto a suas características. Para isso, a prototipação dos hardwares passou por etapas de projeto e implementação de placa de circuito impresso até a montagem das caixas.

A

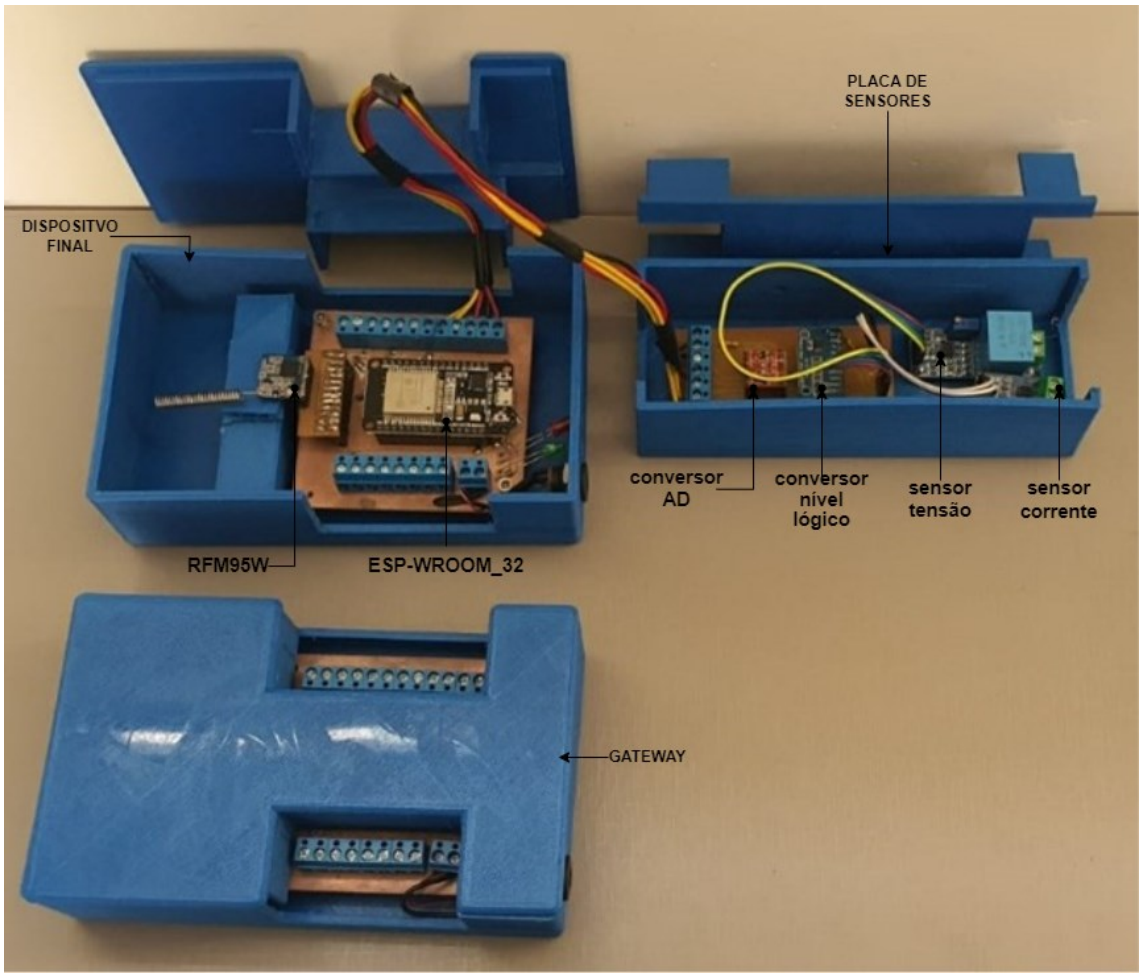


Figura 21 apresenta o protótipo do sistema que será utilizado para conectividade com ilhas de informação.

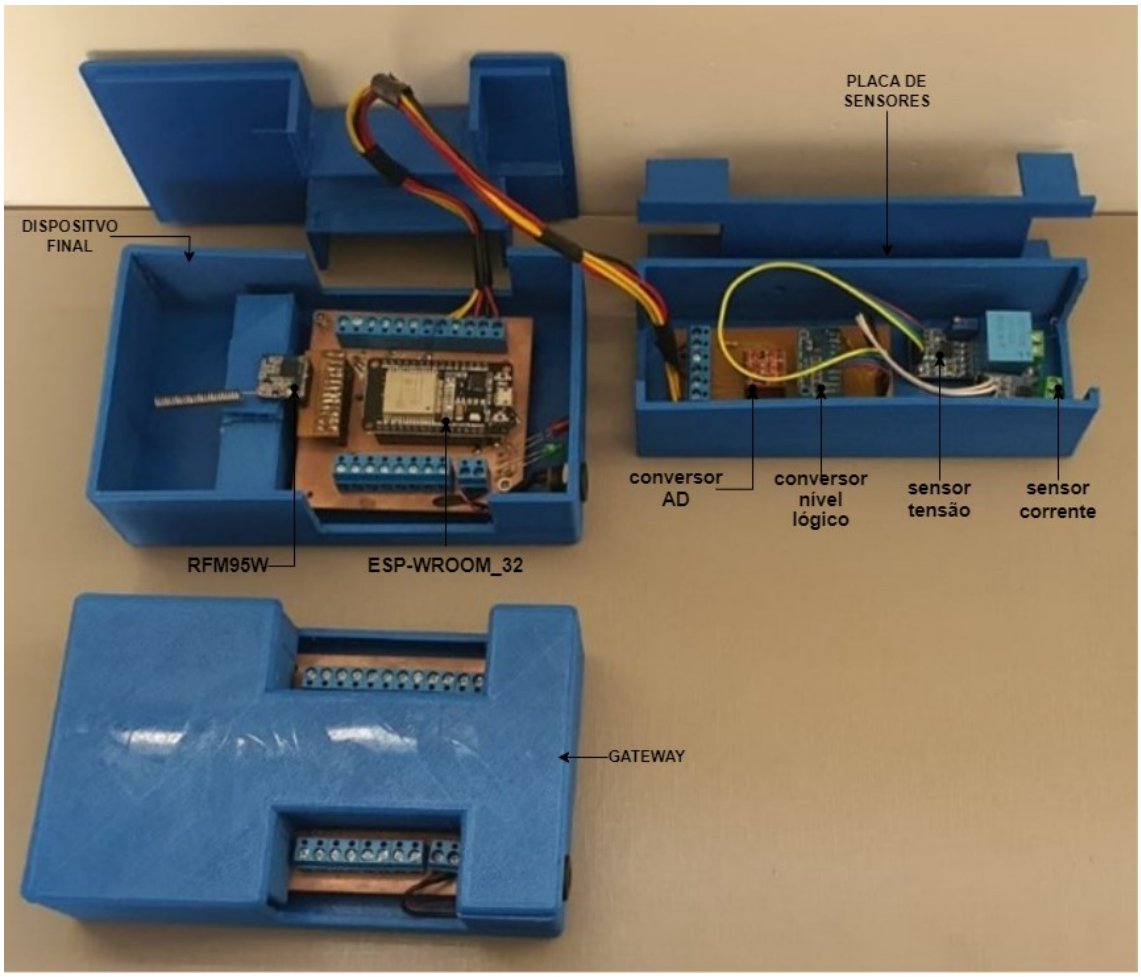


Figura 21 - Protótipo Gateway e dispositivo final.

O protótipo foi desenvolvido seguindo os requisitos do projeto em que o hardware desenvolvido possa comportar tanto como dispositivo final quanto como gateway. O que diferencia o dispositivo final do gateway é somente pelo firmware aplicado dependendo no aspecto funcional que o hardware irá atuar na solução.

Outro elemento da solução é uma placa de sensores que se interliga com dispositivo final a partir de uma comunicação I2C.

As seções posteriores irão apresentar o projeto de placas e componentes utilizados para criação dos protótipos.

Placa multifuncional - Dispositivo / Gateway

Como descrito anteriormente, esse elemento será uma placa multifuncional, que poderá ser utilizada como dispositivo final ou gateway da solução, a depender do firmware instalado. O primeiro passo do desenvolvimento do protótipo da placa multifuncional é a definição dos componentes que irão compô-la.

A placa será composta essencialmente por módulos de computação ESP-WROOM_32 e comunicação LoRa, além de bornes de conexão KRE. A Figura 22 e a Figura 23 apresentam, respectivamente, o diagrama esquemático e o layout da placa multifuncional.

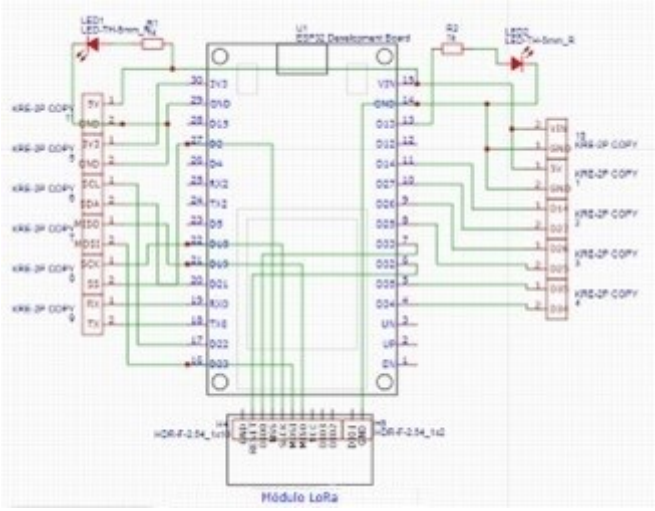


Figura 22 - Esquema de interligação da placa multifuncional.

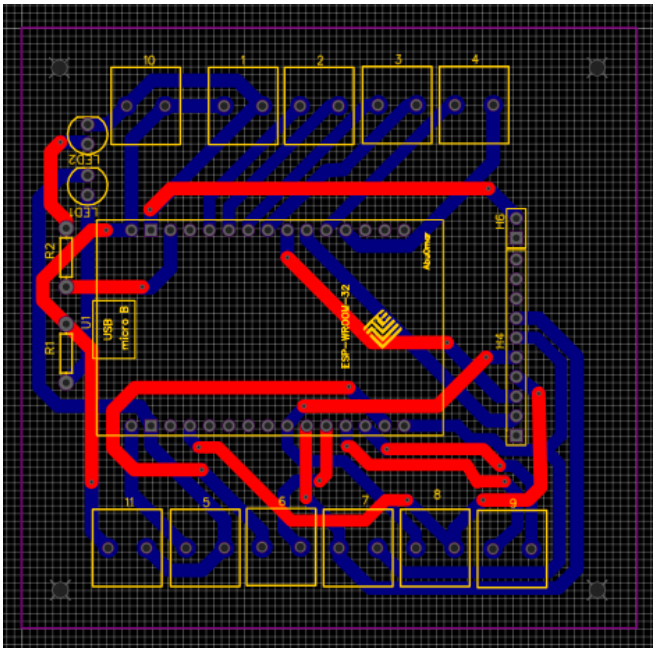


Figura 23 - Projeto de circuito impresso da placa multifuncional.

Placa de sensores

Esse elemento do protótipo será responsável por conter os sensores que irão ser conectados à placa multifuncional. A placa de sensores do protótipo é composta

pelos seguintes módulos sensores: corrente AC, tensão AC e acelerômetro de três eixos (MPU6050). Além disso, a placa contém um conversor A/D de 16 bits (ADS1115) e um conversor de nível lógico (CNL35). A Figura 24 e a Figura 25 apresentam, respectivamente, o diagrama esquemático e o layout da placa de sensores.

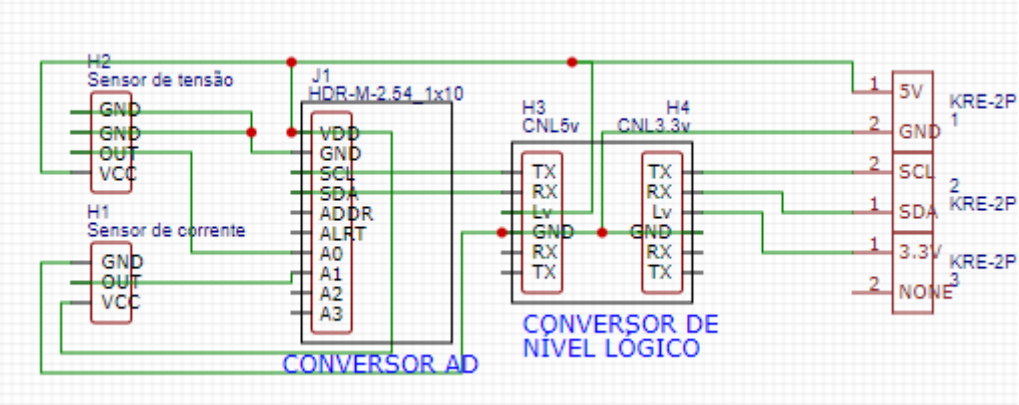


Figura 24 – Diagrama esquemático da placa de sensores.

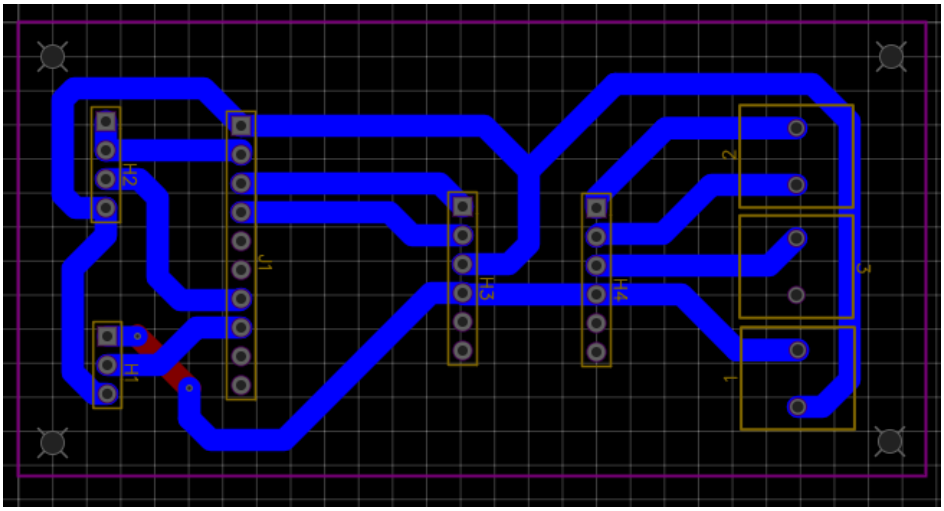


Figura 25 - Projeto de circuito impresso da placa de sensores.

4.2.3 – Desenvolvimento do Firmware

Esta seção destina-se a apresentar o fluxograma do software embarcado do sistema. O fluxograma é apresentado na Figura 26 e representa as etapas que foram consideradas no desenvolvimento e suas interações. Como descrito na seção de protótipo da placa multifuncional, o firmware é o elemento que irá diferenciar os dispositivos com relação à sua função no processo (dispositivo final ou gateway).

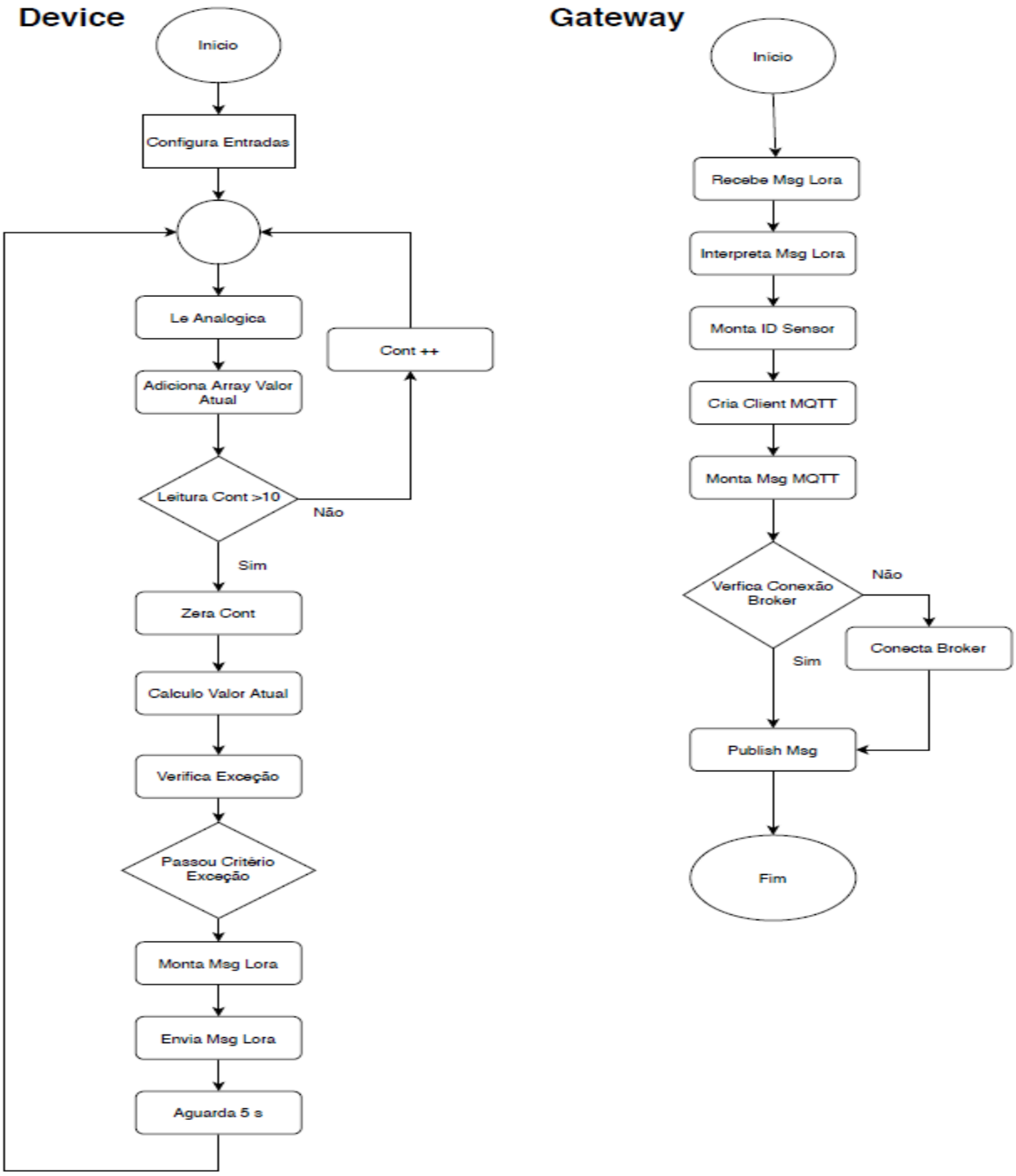


Figura 26 - Fluxograma do dispositivo final e do Gateway.

Os subitens desta seção apresentam, de forma mais detalhada, as etapas deste processo que merecem destaque pela agregação de valor em redução de dados e confiabilidade.

Os firmwares desenvolvidos possuem propriedade intelectual, como registro de programas de computador, identificados pelos seguintes certificados digitais:

Tabela 6 - Registros de Programas de Computador - Camada de Dispositivo.

Número	Título
20210022	Firmware de Device LoRaWAN para monitoramento de ativos industriais.
20210023	Firmware de Gateway MQTT para monitoramento de ativos industriais

Os documentos dos registros podem ser encontrados nas seções **ANEXO A – Registro de Programa de Computador – Firmware Gateway** e **ANEXO B – Registro de Programa de Computador – Firmware Device**.

Exceção de Dados

Na indústria, a maior parte dos processos e elementos monitorados são variáveis analógicas que são amostradas em intervalos constantes e enviadas para armazenamento. Porém, se não houver nenhum tratamento sobre esses dados, qualquer valor da variável lida do campo será enviada na rede e, com isso, o tráfego poderá ser muito alto. Para minimizar o tráfego e consumo de energia do dispositivo final será utilizado o algoritmo de exceção de dados (AVEVA, 2023).

A exceção ocorre com a eliminação de alguns pontos coletados por meio do algoritmo de exceção, que irá classificar a amostra por basicamente dois parâmetros: tempo da última coleta e variabilidade dos últimos valores amostrados.

O algoritmo de exceção é responsável por minimizar a quantidade de dados enviados para o Gateway, reduzindo o volume de dados na comunicação LoRa. Além disso, é responsável, também, por minimizar as perdas de dados. Isso somente é possível, pois tal algoritmo deixa que os dados sejam enviados em apenas 2 casos.

Caso 1: Quando alguma das medições feitas pelo sensor ultrapassa uma variação de “ExcDev” em relação à medida anterior. Dessa forma, evita-se enviar todos os valores iguais ou muito próximos em um pequeno intervalo de tempo, o que pode ser observado na Figura 27 (AVEVA, 2023).

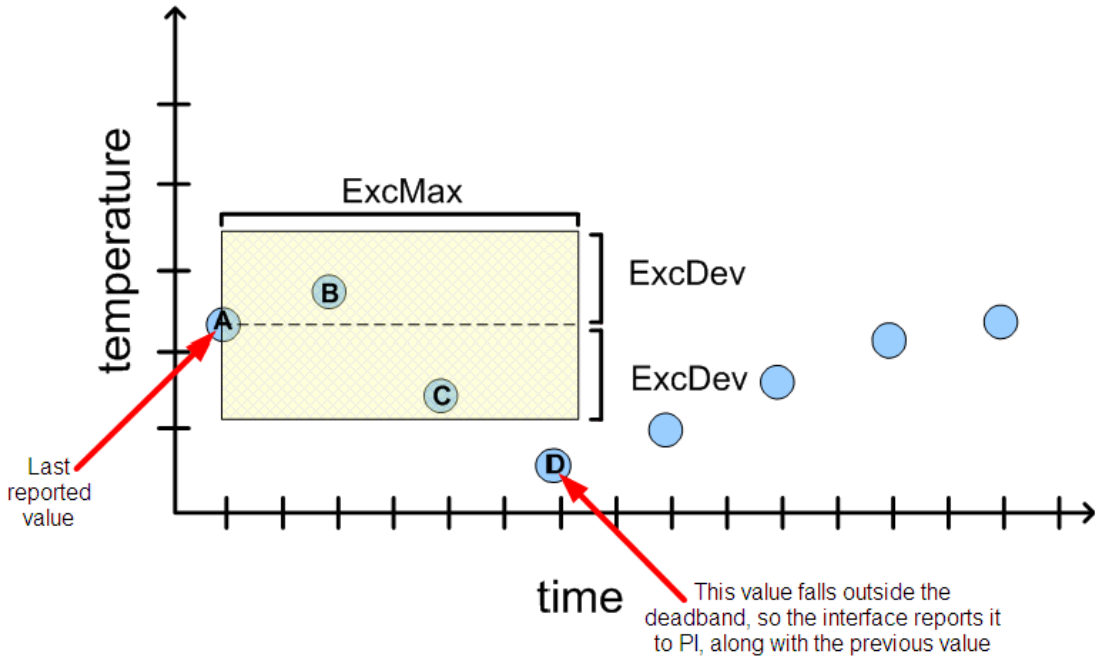


Figura 27 - Demonstração do algoritmo de exceção por variação. (AVEVA, 2023).

Caso 2: Quando se passa um período específico entre a medição atual e a última medição enviada representado na Figura 27 como "*ExcMax*". Por exemplo, se esse intervalo de tempo, que será especificado na instalação do dispositivo final, for de 5 (cinco) minutos, o maior intervalo entre duas mensagens enviadas para o Gateway será de 5 (cinco) minutos, sendo que poderá haver intervalos menores devido ao caso 1. Essa abordagem na arquitetura da solução compõe o elemento associado à redução de dados.

Funções de Autogerenciamento

Visando trazer maior confiabilidade na solução, foram implementadas algumas funções de autogerenciamento, conforme descrito por CUNHA e BRAGA.

A função de autocura consiste na identificação da falta de conexão das comunicações (LoRa, WiFi e MQTT) utilizadas e sua reconexão posteriormente. Dessa maneira, foram elaboradas funções que reconhecem a falta de conexão e tentam reconectar os dispositivos antes de enviar os dados, que são armazenados até que possam ser enviados, de modo que não haja perdas de dados.

Para solucionar o problema de identificação e reconexão dos módulos LoRa, foi desenvolvido um algoritmo que consiste em um "handshake" entre o dispositivo final e o gateway, onde o dispositivo final pergunta se o gateway recebeu a mensagem e o

gateway responde à pergunta feita pelo dispositivo final, assegurando assim a comunicação e possibilitando o envio das mensagens. Essa verificação é feita todas as vezes em que há alguma mensagem a ser enviada.

Nesse contexto, caso não haja a plena comunicação entre os dispositivos, o dispositivo final armazena as medições em um buffer que serve como uma memória para esses dados. Quando a conexão volta, todos os dados armazenados no buffer são enviados. Em seguida, o buffer tem todas as suas posições zeradas. A Figura 28 apresenta o fluxograma desse processo.

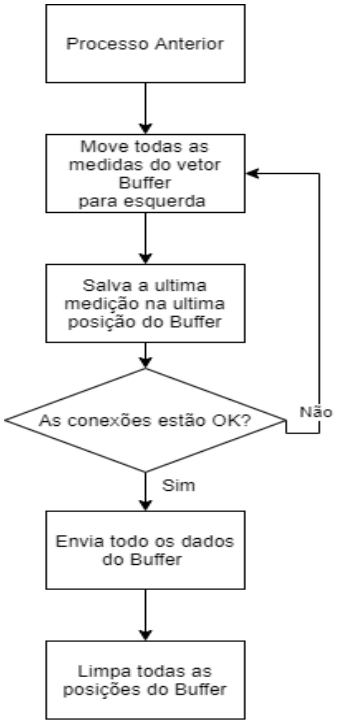


Figura 28 - Buffer de comunicação.

Um problema que nos deparamos durante a execução do programa é que a reconexão com a rede WiFi e com o servidor Broker demandavam um tempo elevado, o que atrapalhava o recebimento das mensagens LoRa no gateway. Para solucionar este problema, que impacta diretamente no desempenho da solução, foi implementado um algoritmo que utiliza o segundo núcleo de processamento do ESP32, separando assim funcionalmente processos de comunicação das demais funções do firmware.

4.3 – Camada de Comunicação

Essa camada é responsável pela comunicação no âmbito da solução de IIoT. Essa comunicação possui duas tratativas, a primeira comunicação interna entre

dispositivos finais e gateway, usando a tecnologia LoRa, e a segunda a comunicação para camada de aplicação na nuvem utilizando Wifi e o protocolo MQTT.

Entre o dispositivo final e o gateway a solução utiliza a tecnologia LoRa, para isso, foi implementado na solução um protocolo para permitir uma comunicação multi-ponto – ponto, validação da comunicação, funcionalidades de autocura e sincronismo. Um ponto de destaque da solução é a decisão de não usar o LoRaWAN e sim implementar um protocolo sobre a tecnologia LoRa visando a minimização dos custos.

Uma vez que o gateway recebeu o pacote LoRa, ele deve interpretar os dados e criar uma mensagem MQTT para publicar no Broker, alcançando assim a camada de aplicação. Um item importante de ser considerado desta solução é com relação à arquitetura e locação das camadas, pois a camada de aplicação da solução está em nuvem.

Um fator importante a ser considerado nas soluções de iIoT é a segurança da informação e, por isso, existe um olhar específico sobre o tema.

4.3.1 – Segurança da informação

No contexto da solução, a cibersegurança é de extrema importância devido à grande quantidade de dispositivos interconectados e à natureza dos dados transmitidos. São dados industriais que podem inferir o funcionamento correto ou não de uma bomba, então a qualidade e assertividade desta informação é essencial para criar um base de dados confiável para um ambiente analítico.

Para a comunicação entre o *Gateway* e o ambiente de dados na nuvem utiliza-se o protocolo MQTT. Como pré-requisito da solução o ambiente deve possuir uma conexão Wifi segura com a internet.

Um típico problema de segurança neste tipo de comunicação é o ataque do "homem do meio" (Man-in-the-Middle, MitM), que ocorre quando um atacante se posiciona entre a comunicação de duas partes legítimas, interceptando e possivelmente modificando as informações transmitidas. Esse ataque permite que o atacante possa ler, alterar ou até mesmo injetar dados maliciosos na comunicação, sem que as partes envolvidas percebam. Este tipo de ataque compromete rigorosamente o objetivo da solução de fornecer um ambiente de dados de qualidade e confiável para servir elementos de análise.

Para garantir a segurança da informação existem algumas ações realizadas na solução para manter a segurança da comunicação e uma delas é a utilização de criptografia e autenticação com o uso do protocolo TLS (Transport Layer Security). O

TLS é um protocolo criptografado projetado para fornecer comunicação segura pela internet. Ele é amplamente utilizado para proteger a transferência de dados entre clientes e servidores em uma rede, garantindo três principais aspectos de segurança: confidencialidade, integridade e autenticidade.

O uso deste protocolo foi implementado no firmware do gateway e no broker da nuvem para proteger a comunicação e criptografar os dados transmitidos. Entre o dispositivo final e o gateway existe um protocolo de identificação entre consumidor e produtor do dado que garante que o produtor dos dados (dispositivo final) somente continue a comunicação uma vez que ele recebe uma confirmação identificada do gateway.

Com essa abordagem, a camada de comunicação garante a autenticação, a confidencialidade e a integridade das comunicações entre o gateway e o ambiente de dados da camada de aplicação.

4.4 – Camada de Aplicação

A camada de aplicação na arquitetura de sistemas IoT é a camada superior desta infraestrutura. Nela é executada a ação de receber dados da camada de comunicação, contextualizar, tratar, armazenar e prover dashboards que possibilitem insights sobre os ativos. Além disso, como objetivo principal, essa camada se torna uma estrutura de dados robusta e confiável para fornecer dados para uma estrutura analítica.

A camada de aplicação pode ser local ou em nuvem, pode conter aplicações tradicionais ou utilização de contêiners e orquestradores.

Conforme observamos nas revisões bibliográficas em relação a arquiteturas de sistemas, a utilização de uma infraestrutura de contêiners com orquestrador em nuvem é um salto para a indústria e traz ganhos em escalabilidade, produtividade e desempenho computacional.

Portanto, a solução para a camada de aplicação, buscando inovar e colher os frutos desta inovação para a indústria, foi a utilização de contêiners em nuvem.

A solução utiliza a nuvem da Azure como ferramenta cloud e para orquestrar os contêiners é utilizado o Kubernetes. A solução possui quatro contêiners e cada um possui especificidade funcional para a solução. Os serviços (contêiners ou pods) do Kubernetes são:

- Mosquitto – Broker MQTT
- Node-Red – Aplicação para Driver MQTT
- InfluxDb – Base de dados – “Time series db”
- Grafana – Plataforma de visualização de dados

Para organizar melhor a arquitetura, existem alguns passos anteriores à criação do cluster e criação dos serviços, para organizar melhor a estrutura e ter maior controle e gestão dos recursos.

- Definir a subscrição

  (az account set --subscription <subscription_id>)

- Criar um grupo de recursos

  (az group create --name MOBI --location eastus)

Uma boa prática é qualquer elemento a ser criado e configurado para essa nova arquitetura estar associado a esse grupo de recursos.

O cluster Kubernetes é dividido em dois componentes, o painel de controle que irá fornecer serviços e orquestração das cargas de trabalho e os nós que executam as cargas de trabalho, como exemplificado na Figura 29.

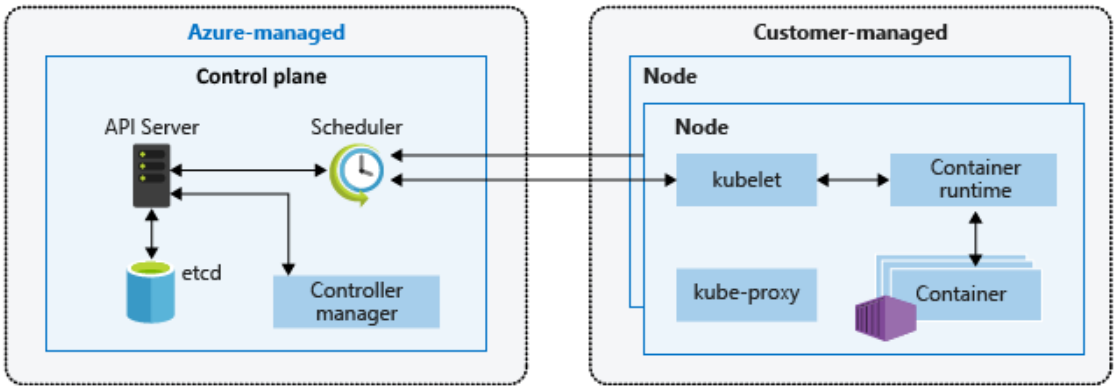


Figura 29 - Estrutura do Cluster Kubernetes.

Por sua vez, para esses nós se comunicarem e compartilharem arquivos, eles necessitam de uma infraestrutura de rede (Azure Virtual network Interface) e de arquivos (Azure Disk e Azure Files) assim com apresentado na Figura 30.

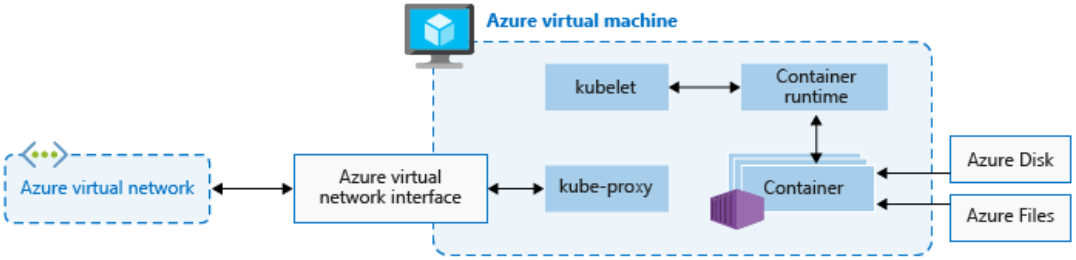


Figura 30 - Infra nó Azure.

O fluxograma da Figura 31 exemplifica as etapas posteriores para criação da infraestrutura seguindo a premissa de criar estruturação e elementos compartilhados inicialmente.

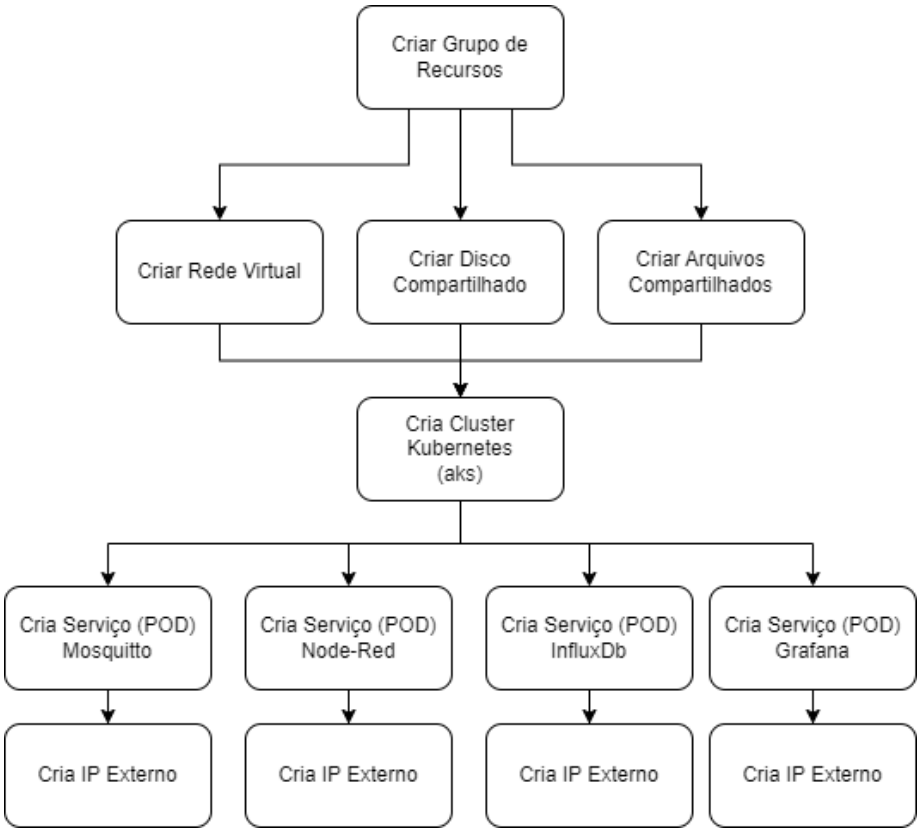


Figura 31 - Fluxograma Etapas Infraestrutura IaC.

Esta arquitetura possui uma característica que é disruptiva em relação às demais arquiteturas anteriormente utilizadas na indústria de processo, em que utilizamos a capacidade de uso da infraestrutura como código, minimizando o tempo de setup, maximizando a replicabilidade da solução e ainda possibilitando maior controle de alterações e versionamento da infraestrutura definida.

A arquitetura desenvolvida para a camada de aplicação é um código e possui propriedade intelectual registrada, como programa de computador, no seguinte certificado digital:

Número	Título
20240010	Arquitetura de Kubernetes na nuvem para implementação do Monitoramento de Bombas Industriais Inteligente.

O documento do registro pode ser encontrado na seção **ANEXO C – Registro de Programa de Computador – Arquitetura Mobi**.

4.4.1 – Broker MQTT

Na arquitetura da solução o broker MQTT é um contêiner, que na estrutura do Kubernetes, é inicializado como serviço. O uso de containers traz um ganho significativo para a solução, pois é utilizado um contêiner oficial do fabricante do software e na configuração do serviço, o mesmo é parametrizado. A solução utilizou a imagem oficial do Mosquitto como broker da comunicação MQTT, como apresentado na Figura 32.

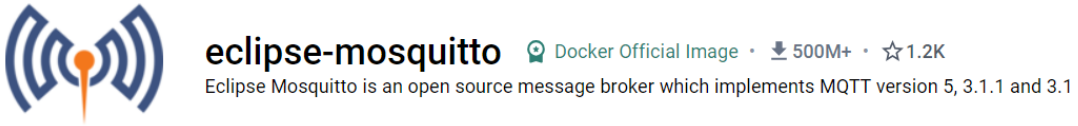


Figura 32 – Imagem Oficial do contêiner do eclipse-mosquitto (fonte: Docker Hub).

Um elemento de destaque para esta aplicação em específico é a configuração do serviço, que deve utilizar as configurações do TLS para garantir uma comunicação com o gateway da camada de comunicação de forma segura.

No Kubernetes é necessário criar um volume virtual, que irá conter a estrutura de pastas padrão para o Mosquitto e onde serão feitas as configurações de segurança, log e dados. Para conectar o volume virtual do Kubernetes às pastas especificas do serviço do Mosquitto, na inicialização do serviço, o mesmo deve ser apontado.

4.4.2 –Driver MQTT

A ferramenta escolhida para o desenvolvimento deste driver foi o Node-RED, primariamente devido à sua facilidade de desenvolvimento utilizando conceitos de low-code e à sua ampla adoção em aplicações de IoT (Internet das Coisas). O Node-RED emprega uma abordagem baseada em nós para conectar uma variedade de elementos, incluindo dispositivos, blocos de código, integrações e outros recursos. Esta solução utiliza de um contêiner do Node-RED instanciado no Kubernetes para servir a solução, como apresentado na Figura 33.

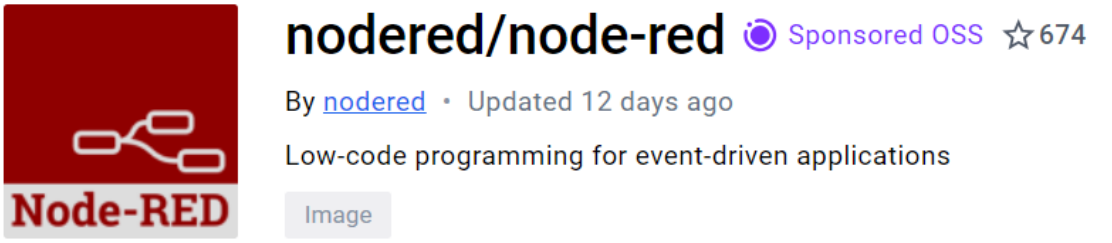


Figura 33 - Imagem Docker do Node-RED.

De forma simplificada, o propósito desta aplicação é ler os dados das mensagens enviadas pela camada de comunicação e persisti-los em uma base de dados. Para alcançar este objetivo, diversos pontos devem ser considerados na solução, os quais serão detalhados na Figura 34.

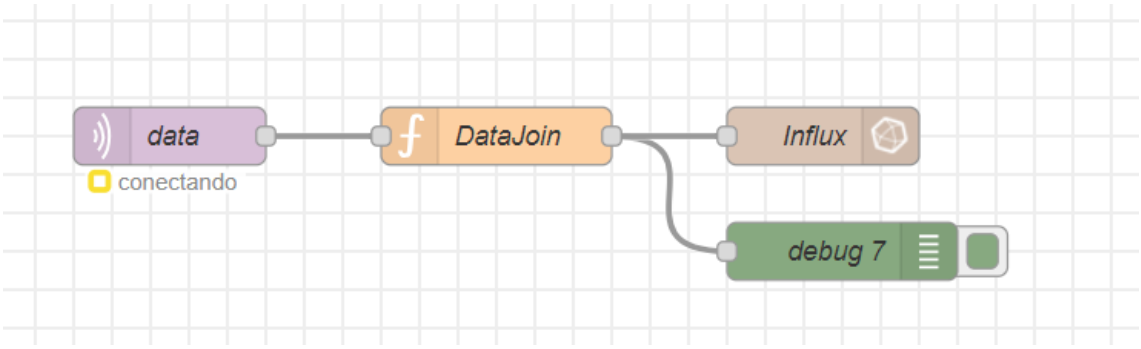


Figura 34 - Fluxo leitura de dados – Driver MQTT.

Primeiramente, para ler as mensagens enviadas pela camada de comunicação, a primeira etapa deste driver é se registrar como assinante (subscriber) do tipo específico de mensagem no broker MQTT. Sempre que o broker (servidor de mensagens) receber esse tipo de mensagem, ele a encaminhará para quem assinou o tópico correspondente.

A segunda etapa do driver, exemplificado na Figura 35, consiste em interpretar a mensagem, que está no formato JSON, e estruturá-la para então salvá-la no bucket da base de dados, que neste caso é o InfluxDb. A estrutura de dados implementada assegura que cada dado armazenado contenha um carimbo de tempo, o valor e características que permitem sua contextualização no ambiente industrial, tais como o asset (equipamento), tag (descrição da variável), tipo (tipo daquela variável, por exemplo corrente) e a unidade de engenharia do valor medido.

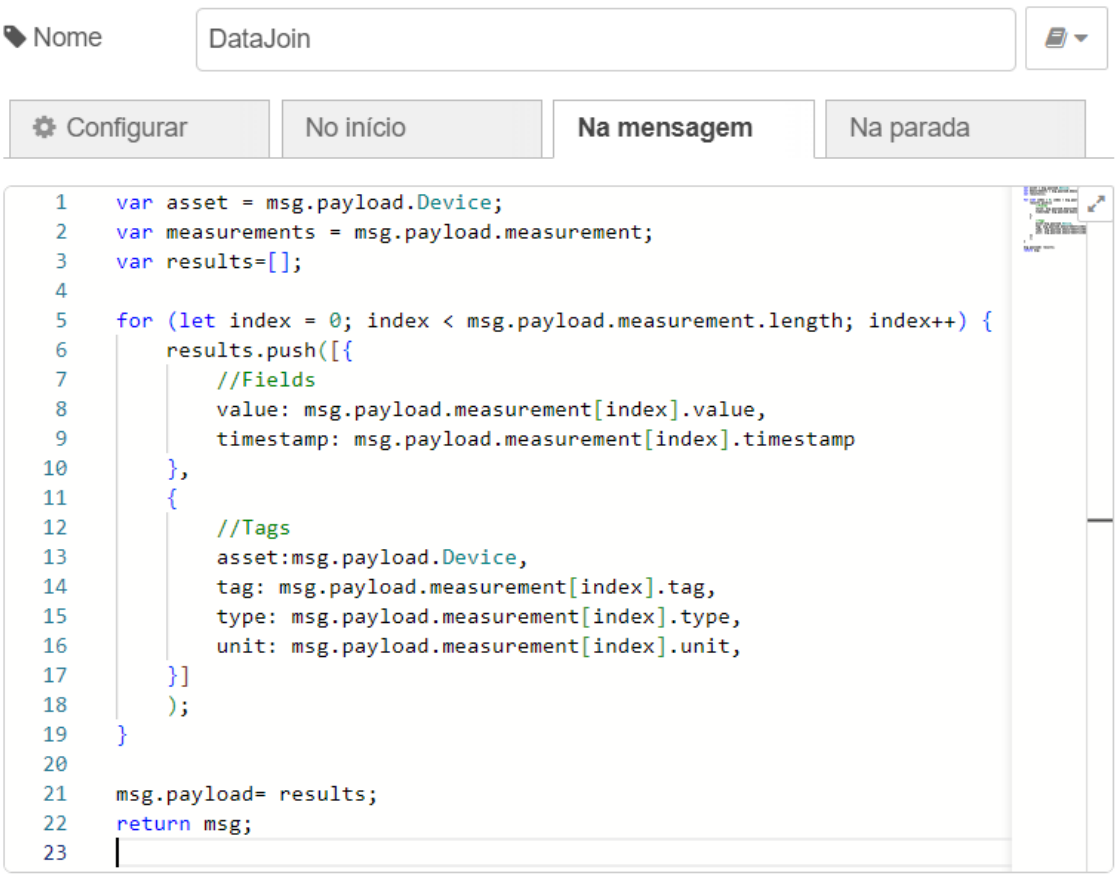


Figura 35 - Script de interpretação da mensagem.

Assim como na comunicação entre as camadas, também há uma preocupação com a segurança da informação, utilizando autenticação para a comunicação tanto com o MQTT Broker quanto com o InfluxDB. Dessa forma, a solução não permite que ataques cibernéticos comprometam a veracidade das informações ou as utilizem para outros fins ilícitos.

O driver MQTT em Node-RED desenvolvido possui propriedade intelectual registrada como programa de computador no seguinte certificado digital:

Número	Título
20240010	Driver MQTT Low-Code para o Monitoramento de Bombas Industriais Inteligente (MoBI IoT)

O documento do registro pode ser encontrado na seção **ANEXO D – Registro de Programa de Computador – Driver MQTT**.

4.4.3 – Camada de Aplicação – Armazenamento dos dados

Para atender à necessidade de armazenamento de dados de variáveis de processo industrial ao longo do tempo, optou-se pelo uso do InfluxDB. A seleção foi motivada pelas características próprias de sistemas IoT industriais, tais como escalabilidade, baixa latência e facilidade de integração, critérios essenciais que foram apresentados na Tabela 7 que subsidiou essa decisão.

Tabela 7 - Comparativo entre bancos de dados temporais.

Recurso	**TimescaleDB**	**InfluxDB**	**TempoDB**	**KDB+**
Modelo de dados	SQL	NoSQL	NoSQL	NoSQL
Linguagem	SQL	InfluxQL	Não especificado	Q
Escalabilidade	Alta	Alta	Alta	Muito alta
Suporte a SQL	Sim	Não	Não	Não
Agregações	Sim	Sim	Sim	Sim
Replicação	Sim	Sim	Sim	Sim
Latência	Baixa	Baixa	Não especificado	Baixa
Suporte à alta disponibilidade	Sim	Sim	Sim	Sim
Integração	Postgres, Prometheus	Grafana	AWS, Grafana	Não especificado
Licença	Apache 2.0	MIT	Comercial	Comercial
Documentação	Boa	Boa	Boa	Boa

Como citado anteriormente, a arquitetura da camada de aplicação é baseada em contêiners orquestrados pelo Kubernetes na Azure e, por isso, utilizou uma imagem Docker oficial do InfluxDB armazenada no Docker Hub, como apresentado na Figura 36.

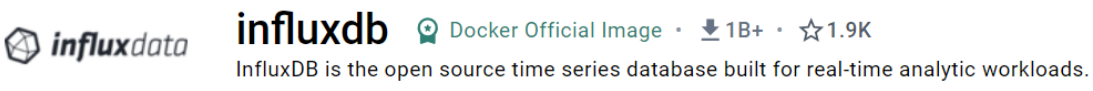


Figura 36 - Imagem Docker oficial – InfluxDb.

Uma característica importante desse banco é que ele é um banco NoSQL (Not Only SQL) em seu modelo de dados, oferecendo assim para os usuários a capacidade de criar modelos mais flexíveis para armazenar e recuperar dados. Por isso, o cliente que armazena dados poderá fazê-lo de maneira que atenda as aplicações que irão consumi-lo, podendo inclusive semi-estruturar e criar contexto como no caso do Driver MQTT da seção anterior.

No InfluxDB foi criado um bucket chamado “data” que contém todos os dados recebidos dos ativos dessa solução através do driver MQTT da camada de aplicação. Por sua vez, esse cliente criou para solução uma estrutura de dados com propósito de contextualizar os dados industriais contendo, para cada “measurement”, além do timestamp e valor, atributos de “asset”, “tag”, “type” e “unit”. Essa estrutura pode ser observada Figura 37 da ferramenta de query do InfluxDB.

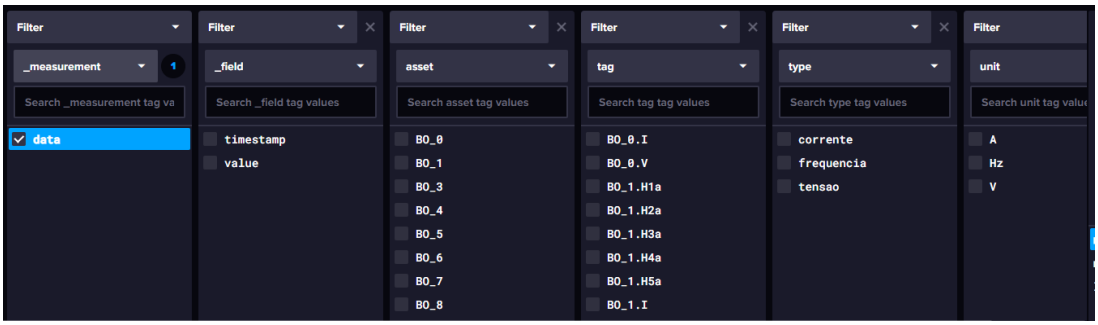


Figura 37 -Exemplo de Query Builder do InfluxDb.

A segurança é um elemento que é primordial e tratada em todas as camadas e aplicações da solução e, portanto, também foi considerada para o banco de dados. Na base de dados tanto para ingestão de dados como para consumo, todas as transações são autenticadas através de um token de API. Nesta aplicação, existem tokens de comunicação para a inserção de dados pelo driver MQTT e para o consumo de dados pela camada de visualização e as permissões são configuradas de acordo com suas necessidades funcionais, conforme apresentado na Figura 38.

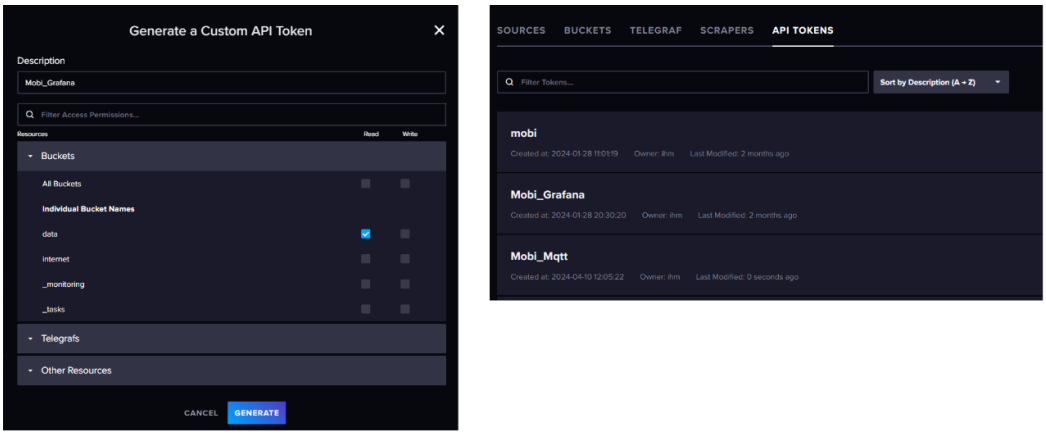


Figura 38 - Configuração de API Tokens – Segurança.

4.4.4 – Visualização de Painéis (Dashboards) de Bombas Industriais

Para atender à necessidade de visualização dos dados dos elementos monitorados na solução, optou-se pelo uso do Grafana. A escolha foi devida as características como gratuidade (código aberto), suporte de integração com diversas bases de dados e usabilidade. A Tabela 8 – Comparativo de visualizadores subsidiou essa decisão.

Tabela 8 – Comparativo de visualizadores.

Visualizador	Descrição	Recursos Principais	Características para Soluções IoT	Outras Características	Preço
Grafana	Plataforma de visualização e monitoramento de dados	Flexibilidade e extensibilidade, suporte a diversos bancos de dados	Suporte a plugins e integrações, painéis dinâmicos, suporte a protocolos IoT	Suporte a monitoramento de dispositivos em tempo real	Gratuito (código aberto)
Tableau	Plataforma líder em visualização de dados	Interface intuitiva, suporte para várias fontes de dados	Recursos avançados de análise, colaboração em tempo real, integração com serviços em nuvem para IoT	Suporte a visualização de dados de sensores e dispositivos IoT	Licença a partir de $70 por usuário/mês
Power BI	Ferramenta da Microsoft para análise de dados	Integração com outras ferramentas Microsoft, AI-powered	Capacidade de embedding em aplicativos, modelagem avançada, integração com Azure IoT	Suporte a streaming de dados para análise em tempo real	Licença a partir de $9,99 por usuário/mês
Google Data Studio	Oferecido pelo Google, focado na colaboração	Conexão com fontes de dados do Google, fácil compartilhamento	Colaboração em tempo real, integração com Google Cloud IoT	Suporte a visualização de dados de dispositivos IoT	Gratuito, com opções de conectores pagos

O Grafana utilizado na solução é uma imagem docker do Docker Hub orquestrado pelo Kubernetes na Azure, conforme referenciado na Figura 39.

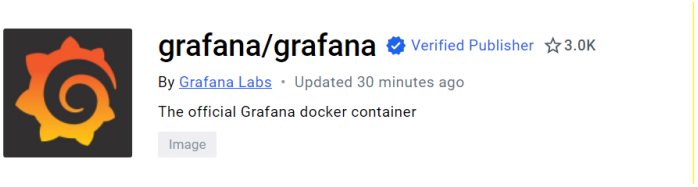


Figura 39 - Imagem Docker – Grafana.

Funcionalmente, a aplicação apresenta uma visualização de uma tela com a capacidade de filtrar o “Asset” (equipamento) e “type” (tipo de variável monitorada) permitindo assim ao usuário avaliar os diversos dados atuais e históricos das grandezas recebidas da infraestrutura de IoT Industrial da solução. A Figura 40 apresenta a visualização de um dashboard apresentando os dados de últimos valores e tendência da corrente da uma bomba (BO_03).

Figura 40 - Dashboard de monitoramento de bombas industriais.

Um ponto de destaque deste dashboard é que os filtros são carregados dinamicamente e a inserção de mais um ativo a ser monitorado ou mais um tipo de variável é transparente para o usuário final e não requer novos desenvolvimentos.

O dashboard desenvolvido no Grafana possui propriedade intelectual registrada, como programa de computador, no seguinte certificado digital:

Número	Título
20240009	Visualização dos dados (representações gráficas de lista, gauge e tendências) do Monitoramento de Bombas Industriais Inteligente (MoBI IoT)

O documento do registro pode ser encontrado na seção **ANEXO E – Registro de Programa de Computador – Dashboard Mobi**.

5 – Validação do ambiente de dados

Para validar a solução proposta deve-se considerar que o sistema atenda a objetivos específicos associados a custo, solução escalável, possibilidade de integração com sistemas de terceiro e uso dos dados para análise.

Um dos objetivos desta solução é que a infraestrutura física de coleta e envio de dados tenha um custo baixo. Para isso foi desenvolvido um hardware e um protocolo visando custo baixo e performance suficiente. O dispositivo final e o gateway foram desenvolvidos com o requisito de custo muito baixo usando dispositivos eletrônicos que atendessem estes requisitos como ESP32 (ESP-WROOM_32) e modulo LoRa (RFM95W), também, ainda visando diminuição dos custos, foi desenvolvido em cima da tecnologia LoRa um protocolo que permite comunicação multi-ponto – ponto sem a necessidade de usar dispositivos homologados na LoRaWAN.

Outro elemento crucial para a solução é a capacidade de escalar a solução, ou seja, a infraestrutura lógica deve ser capaz de crescer de acordo as demandas do negócio e suportar volumes crescentes de dados sem comprometer o desempenho.
Essa capacidade fica a cargo da arquitetura de lógica definida e suas características usando Kubernetes. Escalonar pods no Kubernetes (Figura 41) é um processo que envolve aumentar ou diminuir o número de réplicas de um pod para garantir que a aplicação tenha recursos suficientes para atender à demanda atual. A solução adotou a VPA (Vertical Pod Autoscaler), onde o orquestrador ajusta automaticamente o incremento dos pods baseado nas solicitações e limites de recursos dos contêiners.

Nome	Namespace	Status	Tipo	IP do Cluster	IP Externo	Portas
kubernetes	default	Ok	ClusterIP	10.0.0.1		443/TCP
kube-dns	kube-system	Ok	ClusterIP	10.0.0.10		53/UDP,53/TCP
metrics-server	kube-system	Ok	ClusterIP	10.0.71.98		443/TCP
ama-metrics-ksm	kube-system	Ok	ClusterIP	10.0.70.180		8080/TCP
node-red-service	default-1707327919123	Ok	LoadBalancer	10.0.228.93	172.214.115.137	80:30728/TCP,1...
influxdb-service	default-1707328273584	Ok	LoadBalancer	10.0.111.221	172.214.112.248	80:32679/TCP,8...
grafana-service	default-1707331331295	Ok	LoadBalancer	10.0.253.127	172.212.122.114	80:31485/TCP,3...
ama-metrics-operator-targets	kube-system	Ok	ClusterIP	10.0.43.179		80/TCP

Figura 41- Pods Kubernetes

A integração de dados de diversas fontes e a análise eficaz dessas informações são componentes fundamentais para extrair insights valiosos e apoiar a tomada de decisões estratégicas. Para isso, os dados armazenados nos "*buckets*" do InfluxDB possuem necessidade de ser acessado por ferramentas diversas como dashboards tipo o Grafana (Figura 42) ou mesmo scripts em Python para utilizar bibliotecas de análise de dados.

Figura 42 - Gráfico plotado no Dashboard Grafana.

Para acessar a base de dados usando Python foi utilizada a biblioteca “influxdb-client” instalada no comando (pip install influxdb-client). Para validar a capacidade de recuperar dados do InfluxDB com o Python foi utilizado o código representado na Figura 43 utilizando o Jupyter Notebook.

```python
from influxdb_client import InfluxDBClient, Point, WritePrecision
from influxdb_client.client.write_api import SYNCHRONOUS
import datetime


# Variáveis de configuração
url = "http://localhost:8086"  # URL do InfluxDB
token = "DInHNrhWc-NjwrcoqNOjRXNM76xBtTsoVtzcQlR_OGyG-TKE4FhlHh_GN1SuCpHZ5cqNyaoaysAzAmhRYMwMeA=="       # Token de autenticação
org = "mobi"                # Organização
bucket = "data"          # Bucket


# Conectar ao InfluxDB
client = InfluxDBClient(url=url, token=token)


# Obter a API de consulta
query_api = client.query_api()


#Consultar dados
query = f'from(bucket: "data") |> range(start: -1h) |> filter(fn: (r) => r._measurement == "data")'

tables = query_api.query(org=org, query=query)


# Imprimir resultados
for table in tables:
    for record in table.records:
        print(f'Time: {record.get_time()}, Location: {record["location"]}, Temperature: {record["_value"]}')


# Fechar a conexão
client.close()
```

Figura 43 - Python consumindo dados do InfluxDb.

Com esses objetivos atendidos, entende-se que a solução possui capacidade de mover a manutenção das bombas industriais para um cenário de proatividade com ações assertivas baseadas em dados analíticos e, portanto, minimizando as perdas produtivas.

6 - Conclusões

No Brasil as indústrias possuem cenários bem heterogêneos, nelas encontramos sistemas de controle novos e outros com mais de 15 anos de atuação sem atualizações, porém, independente do seu grau de automação, na maioria das indústrias, a manutenção tem a mesma tratativa, uso massivo de manutenções corretivas e utilização de rotinas de manutenção preventiva.

O sistema de monitoramento de bombas industriais inteligente objetiva, através de uma infraestrutura de dados eficiente e confiável, fornecer informações para que um ambiente analítico tenha insumos suficientes para mudar o perfil de manutenção de bombas industriais para preditivo.

O sistema conecta bombas de um sistema de automação existente ou bombas isoladas através de uma rede física de baixo custo e com conexão de longa distância utilizando o dispositivo final e o gateway LoRa propostos no projeto.

Além de viabilizar a coleta de dados de sistemas dispersos, a solução incorpora uma camada de inteligência diretamente no dispositivo, economizando o tráfego de dados na camada de comunicação, permitindo assim uma otimização mais eficaz da energia e da velocidade de transmissão da rede. Isso é possível graças à implementação de filtros baseados na dinâmica dos elementos mensurados, conhecidos na solução como regras de exceção de dados. Além disso, garante a segurança da informação por meio de um protocolo de *handshake* identificado entre o gateway e o dispositivo final durante a troca de informações.

Outro aspecto inovador e altamente vantajoso para esse segmento é a adoção de uma arquitetura flexível e escalável por meio de contêineres na nuvem, combinada com a estruturação eficiente e ágil das informações em um banco de dados NoSQL temporal. Essa abordagem oferece uma maior agilidade no desenvolvimento, implantação e escalabilidade do sistema, além de proporcionar uma gestão mais eficiente dos recursos de hardware e garantir uma maior resiliência e disponibilidade do sistema em ambientes dinâmicos e de grande volume de dados.

Para buscar estimar um ganho, podemos exemplificar o uso da CBM tradicional, com custos altos de implementação e pouca escalabilidade, mas, mesmo assim, conseguiu uma economia estimada em $35 bilhões no USA.

Pensando em Brasil, a ABRAMAN – Associação Brasileira de Manutenção e Gestão de Ativos, apresenta um indicador que 5% do faturamento bruto da indústria é gasto com manutenção. Analisando o caso de indústrias de transformação, boa parte

ainda tem sua principal ação de manutenção como corretiva e grande parte destes ativos são bombas industriais.

Em um estudo de caso de uma mineradora brasileira, avaliando produção de minério de ferro, o custo de um minuto de planta parada seria algo em torno de R$30.000,00. Economicamente, a solução proposta se apresenta muito viável, pois possui um custo de 0,06% em relação a 1 minuto de planta parada por bomba.

Com base nos argumentos apresentados, acreditamos que esta solução possui um considerável potencial para impactar positivamente os negócios dos clientes, tanto em termos tecnológicos quanto econômicos. A implementação deste produto oferece uma infraestrutura escalável e confiável para coleta, segurança e armazenamento de dados, tornando-se uma ferramenta essencial para operacionalizar uma abordagem baseada em dados na gestão de manutenção. Neste modelo, as decisões são fundamentadas em dados em vez de vieses ou opiniões, o que ajuda a minimizar ou prevenir perdas no processo produtivo. Além disso, essa solução contribui para a democratização do acesso à informação na indústria.

6.1 – Trabalhos futuros

Este trabalho tratou do contexto dos dados e criar uma infraestrutura de dados escalável e performático para prover dados ao ambiente analítico, capaz de trazer a indústria para manutenção proativa de bombas industriais. Trabalhos futuros serão dedicados à criação de um ambiente analítico utilizando-se das diversas tecnologias de IA, seja ela tradicional ou generativa, para criar predição e insights para os mantenedores.

As duas vertentes de IA, trabalhadas de forma sincronizada, poderiam inclusive dar ao ambiente analítico capacidade prescritiva, apoiando os mantenedores como um copiloto da manutenção.

O primeiro passo para esse trabalho, foi criar a capacidade analítica para uma base de dados e possibilitar que terceiros, com uso de ferramentas de análise de dados, possam entender o comportamento das variáveis e inferir possíveis predições.

O segundo passo seria, nas tendencias das variáveis, habilitar a capacidade de criar anotações. Essas anotações, principalmente aquelas associadas a uma anomalia, poderiam formar uma base de conhecimento na base NoSQL da infraestrutura de dados.

Poderia também ser incluído nessa base de dados, documentos de referência associados a esses ativos, tais como procedimentos operacionais, manuais, troubleshootings e outros.

A terceira parte seria associar a esse ambiente IA, utilizando o LLM ou IA Generativa (como é mais conhecida) a capacidade de interpretação e geração de textos em linguagem natural para funcionar como um assistente do mantenedor e sugerir algumas ações baseadas no conhecimento prévio e contextualização.

Essa abordagem representa uma verdadeira revolução na forma como o conhecimento e a manutenção são gerenciados nas indústrias, introduzindo capacidades prescritivas ao processo. Isso significa que não estamos apenas analisando dados para entender o que aconteceu, mas também estamos capacitados a prever o que pode acontecer e recomendar ações para otimizar o desempenho através de um sistema. Essa capacidade prescritiva representa o mais alto nível da pirâmide analítica clássica, levando a tomada de decisão a um novo patamar de eficiência e precisão.

7 - Referências bibliográficas

3GPP. Release 13 Overview. 2016. Disponível em: https://www.3gpp.org/release-13. Acesso em: 6 jun. 2024.

ABNT. NBR 5462: Confiabilidade e Mantenabilidade. Rio de Janeiro: ABNT, 1994.

ALPAYDIN, E. Introduction to Machine Learning. MIT Press, 2014. (Capítulo 7)

ALVES, S. Análise de Vibração de Máquinas para Manutenção Preditiva. Disponível em: https://www.venturus.org.br/analise-de-vibracao-de-maquinas-para-manutencao-preditiva. Acesso em: 01 de dezembro de 2021.

AMQP 1.0 Specification. OASIS Standard, 2012. Accessível em: https://www.oasis-open.org/committees/amqp/documents/amqp-specification-versions.

AVEVA. PI Server Data Archive Administration Guide. Disponível em: https://docs.aveva.com/bundle/pi-server-s-da-admin/page/1021932.html. Acesso em: 05 jan. 2024.

BANKS, J.; DESJARDINS, T. RabbitMQ in Action: Distributed Messaging for Everyone. Manning Publications, 2014.

BISHOP, C. M. Pattern Recognition and Machine Learning. Springer, 2006. (Capítulo 2.5.4)

BREIMAN, L. et al. Classification and regression trees. CRC press, 1984.

BREIMAN, L. Random Forests. Machine learning, v. 45, n. 1, p. 5-32, 2001.

CARULLO, A. et al. Long range and low power networks for the internet of things: open issues and comparison between LoRaWAN and NB-IoT. Wireless Communications and Mobile Computing, v. 2018, 2018. Disponível em: https://doi.org/10.1155/2018/5983045. Acesso em: 3 out. 2023.

CENTENARO, Michela et al. Long-range communications in unlicensed bands: The rising stars in the IoT and smart city scenarios. IEEE Wireless Communications, v. 23, n. 5, p. 60-67, 2016.

CHELBI, A.; AIT-KADI, D. Inspection and predictive maintenance strategies. In: International Journal of Computer Integrated Manufacturing, v. 11, n. 3, p. 226-231, 1998.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data, 2016.

CUNHA, Adriano B. da; GONTIJO, Alberto de F.; BRAGA, Anísio R.; ARAÚJO FILHO, João E. M. de; LIMA, Danilo P.; BRANDÃO, Dener A. de L.; MACIEL, Pedro H. A.; PINTO, Victor M. M.; MELO, Alexandre V. de; CALIMAN, Edilson H.;

ANDRADE, Johnny S. Resultados da Primeira Rede de Sensores de Auscultação de Barragens Sem Fio, Autonômica e de Baixo Custo da Cemig GT. IX Citenel.

EMBARCADOS. Introdução ao OPC UA. Disponível em: https://embarcados.com.br/introducao-ao-opc-ua/.

GANDA, W. L.; MOUTINHO, M. V. B.; CASTRO, G. B. Aplicação de algoritmo para detecção de falhas em bombas centrífugas multi-estágio de injeção de água em reservatórios de petróleo. In: XIII Simpósio Brasileiro de Automação Inteligente. Porto Alegre – RS, 1o – 4 de Outubro de 2017.

GARG, N. Kafka Streams in Action. Manning Publications, 2018.

GSMA. NB-IoT: Narrowband IoT Market Status. 2024. Disponível em: https://www.gsma.com/iot/narrowband-iot/. Acesso em: 6 jun. 2024.

GULATI, R. Maintenance and Reliability. Best Practices. 2ª ed. New York: Industrial Press, 2013.

HAFEEZ, TAIMUR; XU, LINA; MCARDLE, GAVIN. Edge Intelligence for Data Handling and Predictive Maintenance in IIOT. IEEE Access, Digital Object Identifier 10.1109/ACCESS.2021.3069137

HAN, J.; KAMBER, M.; PEI, J. Data Mining: Concepts and Techniques. Elsevier, 2011.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media, 2009.

HOHPE, G.; WOOLF, B. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Professional, 2003.

IEEE COMPUTER SOCIETY. IEEE Standard for Local and Metropolitan Area Networks--Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). New York: IEEE, 2011. Disponível em: https://standards.ieee.org/standard/802_15_4-2011.html. Acesso em: 1 fev. 2024.

IGBEE ALLIANCE. Zigbee Specification. 2021. Disponível em: https://zigbeealliance.org/solution/zigbee/. Acesso em: 5 nov. 2023.

KREPS, J. et al. Kafka: A Distributed Messaging System for Log Processing. Disponível m: https://kafka.apache.org/documentation/.

Kreps, Jay; Narkhede, Neha; Rao, Jun. Kafka: The Definitive Guide: Real-Time Data and Stream Processing at Scale. O'Reilly Media, 2017.

LAMIM FILHO, P. C. M.; BRITO, J. N.; PEDERIVA, R. Detection of electrical faults in induction motors using vibration analysis. In: Journal of Quality in Maintenance Engineering, v. 19, n. 4, p. 364 – 380, 2013.

LEITAO, P. et al. Industrial Cloud-Based Cyber-Physical Systems: The IMC-AESOP Approach. Springer, 2016.

LI, Q.; ZHANG, L. Reliable and Low Latency Message Delivery in AMQP-Based Publish/Subscribe Systems. In: IEEE Access, v. 6, p. 19099-19114, 2018. DOI: 10.1109/ACCESS.2018.2816389.

LoRa Alliance. LoRaWAN® Overview. Disponível em: https://lora-alliance.org/about-lorawan/. Acesso em: 1 jun. 2024.

LoRaWAN. Disponível em: https://lora-alliance.org/about-lorawan. Acesso em: 01 de junho de 2020.

LU, B.; GUNGOR, V. C. Online and Remote Motor Energy Monitoring and Fault Diagnostics Using Wireless Sensor Networks. In: IEEE Transactions on Industrial Electronics, v. 56, n. 11, p. 4651-4659, 2009.

MAFFEIS, S. et al. AMQP Security Issues. In: 2010 International Conference on Availability, Reliability and Security, p. 53-60, 2010. DOI: 10.1109/ARES.2010.18.

MICROSOFT AZURE. Guia de IA do Azure para soluções de manutenção preditiva. Redmond, WA: Microsoft, 2018. Disponível em: https://docs.microsoft.com/pt-br/azure/architecture/data-science-process/predictive-maintenance-playbook. Acesso em: 30 nov. 2021.

MITCHELL, T. M. Machine learning. McGraw Hill, 1997.

MOKOSMART. How does LoRa sensor send and receive data. Disponível em: https://www.mokosmart.com/how-does-lora-sensor-send-and-receive-data/. Acesso em: 01 de junho de 2020.

MQTT ORG. Documentation MQTT. Disponível em: http://mqtt.org/documentation. Acesso em: 04 de junho de 2020.

NARAYAN, V. Effective Maintenance Management. Risk and Reliability Strategies for Optimizing Performance. 2ª ed. New York: Industrial Press, 2012.

NARKHEDE, N. et al. Kafka: The Definitive Guide: Real-Time Data and Stream Processing at Scale. O'Reilly Media, 2017.

NEPOMUCENO, L. X. Técnicas de manutenção preditiva. 1ª ed. São Paulo: Blucher, 2014.

OPC FOUNDATION. OPC 11030: UA Modelling Best Practices. Disponível em: https://opcfoundation.org/resources/ua-modelling/.

OSI SOFT. A Guidebook to Implementing Condition-Based Maintenance (CBM) Using Real-time Data. Disponível em: https://pisquare.osisoft.com/s/question/0D51I00004UHpkKSAT/cbm-guidebook-version-2. Acesso em: 03 de janeiro de 2022.

PERUCHI, F. L. Análise das causas raízes de falha de uma bomba centrífuga de média consistência aplicada na área de branqueamento de polpa de celulose.

POMORSKY LINESSIO, R. Análise de Vibração de Motores Elétricos Utilizando um Acelerômetro Óptico Biaxial. Curitiba, 2016.

QUINLAN, J. R. C4.5: Programs for machine learning. Morgan Kaufmann, 1993.

QUINLAN, J. R. Induction of decision trees. In: Machine Learning, 1(1), 81-106, 1986.

RABBITMQ. AMQP 0-9-1 Model Explained. Disponível em: https://www.rabbitmq.com/tutorials/amqp-concepts.html.

ROSA, F. L.; JUNG, C. F.; MENGDEN, P. R. Um modelo para avaliação do potencial estratégico de projetos de P&D de Inovação Tecnológica. DOI: 10.14488/1676-1901.v13i3.1421.

SEMTECH CORPORATION. LoRa Technology. 2024. Disponível em: https://www.semtech.com/lora. Acesso em: 5 mai. 2024.

SERRA, L. F. N. Proposta de um plano de manutenção preventiva para bombas de polpa de rejeitos de minério.

SHAW, T.; BARNELL, M. OPC UA: The Basics: An OPC UA Overview for Those Who May Not Have a Degree in Embedded Programming. OPCTI, 2016.

SHI, Y.; JUDD, M. Finding Nearest Neighbors for Multi-Dimensional Data. In: DBKDA 2013: The Fifth International Conference on Advances in Databases, Knowledge, and Data Applications.

SHIN, J.-H.; JUN, H.-B. On condition based maintenance policy. In: Journal of Computational Design and Engineering, v. 2, n. 2, p. 119 – 127, 2015. ISSN 2288-4300.

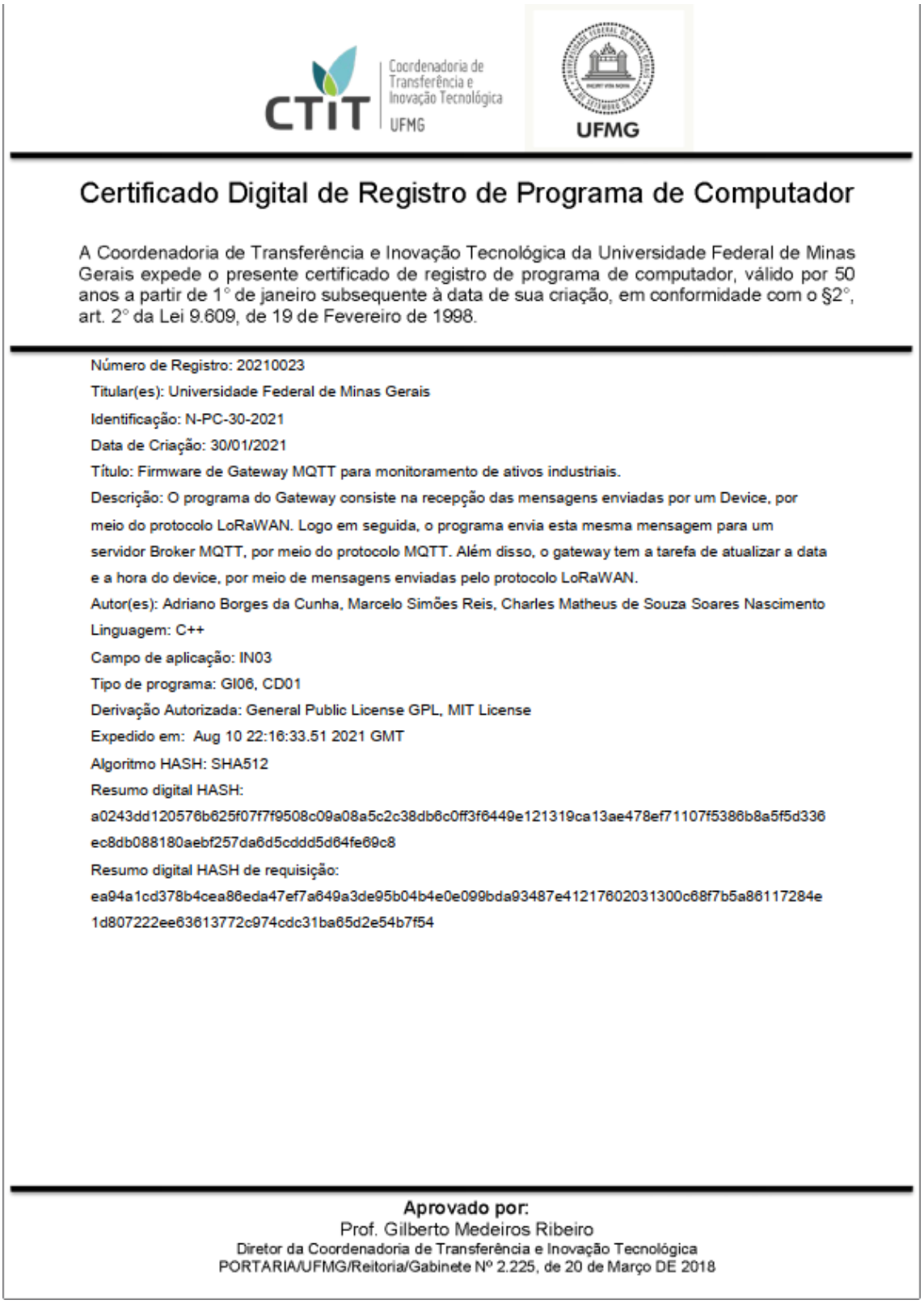
THREAD GROUP. About Thread. Disponível em: https://www.threadgroup.org/About. Acesso em: 15 mai. 2024.

VIDELA, A.; WILLIAMS, J. J. W. RabbitMQ in Action: Distributed Messaging for Everyone. Manning Publications, 2012.

VRBA, P.; STRASSER, T. I. OPC UA in Practice. CRC Press, 2016.

WEISBERG, S. Applied Linear Regression. 3rd ed. Hoboken: John Wiley & Sons, 2005.

ANEXO A – Registro de Programa de Computador – Firmware Gateway



CTIT Coordenadoria de Transferência e Inovação Tecnológica UFMG

UFMG

Certificado Digital de Registro de Programa de Computador

A Coordenadoria de Transferência e Inovação Tecnológica da Universidade Federal de Minas Gerais expede o presente certificado de registro de programa de computador, válido por 50 anos a partir de 1° de janeiro subsequente à data de sua criação, em conformidade com o §2°, art. 2° da Lei 9.609, de 19 de Fevereiro de 1998.

Número de Registro: 20210023

Titular(es): Universidade Federal de Minas Gerais

Identificação: N-PC-30-2021

Data de Criação: 30/01/2021

Título: Firmware de Gateway MQTT para monitoramento de ativos industriais.

Descrição: O programa do Gateway consiste na recepção das mensagens enviadas por um Device, por meio do protocolo LoRaWAN. Logo em seguida, o programa envia esta mesma mensagem para um servidor Broker MQTT, por meio do protocolo MQTT. Além disso, o gateway tem a tarefa de atualizar a data e a hora do device, por meio de mensagens enviadas pelo protocolo LoRaWAN.

Autor(es): Adriano Borges da Cunha, Marcelo Simões Reis, Charles Matheus de Souza Soares Nascimento

Linguagem: C++

Campo de aplicação: IN03

Tipo de programa: GI06, CD01

Derivação Autorizada: General Public License GPL, MIT License

Expedido em: Aug 10 22:16:33.51 2021 GMT

Algoritmo HASH: SHA512

Resumo digital HASH:

a0243dd120576b625f07f7f9508c09a08a5c2c38db6c0ff3f6449e121319ca13ae478ef71107f5386b8a5f5d336ec8db088180aebf257da6d5cddd5d64fe69c8

Resumo digital HASH de requisição:

ea94a1cd378b4cea86eda47ef7a649a3de95b04b4e0e099bda93487e41217602031300c68f7b5a86117284e1d807222ee63613772c974cdc31ba65d2e54b7f54

Aprovado por:

Prof. Gilberto Medeiros Ribeiro

Diretor da Coordenadoria de Transferência e Inovação Tecnológica

PORTARIA/UFMG/Reitoria/Gabinete Nº 2.225, de 20 de Março DE 2018

ANEXO B – Registro de Programa de Computador – Firmware Device

Certificado Digital de Registro de Programa de Computador

A Coordenadoria de Transferência e Inovação Tecnológica da Universidade Federal de Minas Gerais expede o presente certificado de registro de programa de computador, válido por 50 anos a partir de 1° de janeiro subsequente à data de sua criação, em conformidade com o §2°, art. 2° da Lei 9.609, de 19 de Fevereiro de 1998.

Número de Registro: 20210022

Titular(es): Universidade Federal de Minas Gerais

Identificação: N-PC-29-2021

Data de Criação: 30/01/2021

Título: Firmware de Device LoRaWAN para monitoramento de ativos industriais.

Descrição: O programa tem como premissa receber as medições realizadas pelos sensores no campo e selecionar (medidas de tensão e corrente elétrica), dentre elas, quais são relevantes para o sistema. Após selecionadas, as medições são organizadas em uma mensagem que contém, além dos valores medidos, o horário da medição e o status do equipamento. Logo em seguida, a mensagem montada é enviada para outro dispositivo, por meio da tecnologia LoRa. Dessa forma, possibilitando a comunicação entre dispositivos distantes sem o uso de cabos e internet.

Autor(es): Adriano Borges da Cunha, Marcelo Simões Reis, Charles Matheus de Souza Soares Nascimento

Linguagem: C++

Campo de aplicação: IN03

Tipo de programa: GI06, CD01

Derivação Autorizada: General Public License GPL, MIT License, BSD License

Expedido em: Aug 10 21:57:08.513 2021 GMT

Algoritmo HASH: SHA512

Resumo digital HASH:

09a851ac0b33dd8cac6598dd35f7bc9494212f3c6467954146a74b00b25b7aa945007fcae440736e835ae970
16e0509edd41e62ebf27f3947e31fdadad8b5d10

Resumo digital HASH de requisição:

8a5347d13c27ec3647fe78530e3f941d705ad75937b71fed1a3c110b18295fc83002f895a7900e11466bcf593d
4a2e67761cc533e68ec1e543c05d9554bf29ed

Aprovado por:

Prof. Gilberto Medeiros Ribeiro

Diretor da Coordenadoria de Transferência e Inovação Tecnológica

PORTARIA/UFMG/Reitoria/Gabinete Nº 2.225, de 20 de Março DE 2018

ANEXO C – Registro de Programa de Computador – Arquitetura Mobi

CTIT Coordenadoria de Transferência e Inovação Tecnológica UFMG

UFMG

Certificado Digital de Registro de Programa de Computador

A Coordenadoria de Transferência e Inovação Tecnológica da Universidade Federal de Minas Gerais expede o presente certificado de registro de programa de computador, válido por 50 anos a partir de 1° de janeiro subsequente à data de sua criação, em conformidade com o §2°, art. 2° da Lei 9.609, de 19 de Fevereiro de 1998.

Número de Registro :20240010
Titular(es) :Universidade Federal de Minas Gerais (UFMG) , IHM Engenharia
Identificação :N-PC-05-2024
Data de Criação :14/02/2024
Título :Arquitetura de kubernates na nuvem para implementação do Monitoramento de Bombas Industriais Inteligente (MoBI IoT).
Descrição :Este código fonte sobre uma arquitetura de kubernates na nuvem Azure da Microsoft que consta em serviços (containers) de base de dados temporal (influxdb), plataforma lowcode (node-red), um broker MQTT (mosquito) e uma plaforma de visualização de dados lowcode (grafana).
Autor(es) :Marcelo Simões Reis, Cauã Magalhães Pereira, Adriano Borges da Cunha
Linguagem :YAML (serialização de dados legíveis), Notação JSON para representação de nós e fluxos
Campo de aplicação :IN01, IN02, IF01
Tipo de programa :GI08
Derivação Autorizada :Apache License Version 2.0
Expedido em :16/05/2024 21:08:48.188000 UTC (Coordinated Universal Time)
Algoritmo HASH :SHA512
Resumo digital HASH :80abbfc24e1212ac8b50fcb62adb65ac21ec94b0770684344cc35f5732768a82f0930fcc6a6537df442f5faceaac63eee4f30a4a2b143e423a7e337c843561b8
Resumo digital HASH de requisição :8503a96d91d4d48ce46021149d21827aee56f0a79740ae52f611bd7b4712420c8bceb007660b51f3229eb7122b96b3a140101668a4d06f0d0063d92310715c2e

Aprovado por:
Prof. Gilberto Medeiros Ribeiro
Diretor da Coordenadoria de Transferência e Inovação Tecnológica
PORTARIA/UFMG/Reitoria/Gabinete Nº 2.225, de 20 de Março DE 2018

ANEXO D – Registro de Programa de Computador – Driver MQTT

CTIT Coordenadoria de Transferência e Inovação Tecnológica UFMG

UFMG

Certificado Digital de Registro de Programa de Computador

A Coordenadoria de Transferência e Inovação Tecnológica da Universidade Federal de Minas Gerais expede o presente certificado de registro de programa de computador, válido por 50 anos a partir de 1° de janeiro subsequente à data de sua criação, em conformidade com o §2°, art. 2° da Lei 9.609, de 19 de Fevereiro de 1998.

Número de Registro :20240011
Titular(es) :Universidade Federal de Minas Gerais (UFMG) , IHM Engenharia
Identificação :N-PC-06-2024
Data de Criação :01/01/2024
Título :Driver MQTT Low-Code para o Monitoramento de Bombas Industriais Inteligente (MoBI IoT).
Descrição :Programa para receber dados de um broker MQTT de um device que está monitorando um ativo industrial (bomba industrial) e escrever em um banco de dados temporal (influxDb) as variáveis monitoradas.
Autor(es) :Marcelo Simões Reis, Cauã Magalhães Pereira, Adriano Borges da Cunha
Linguagem :Javascript, linguagem de programação visual por fluxo de dados do Node-RED, Notação JSON para representação de nós e fluxos
Campo de aplicação :IN01, IN02, IF01
Tipo de programa :GI08, GI06
Derivação Autorizada :Apache License Version 2.0
Expedido em :16/05/2024 21:15:01.381000 UTC (Coordinated Universal Time)
Algoritmo HASH :SHA512
Resumo digital HASH :6ccda496970898004c1d2901ad995117020f111cdcc15cb9809b9d46819aadb25b d468d59bbfff801535e71d21ba8e517f14fe92b6f4ba8a181ba41d01997355
Resumo digital HASH de requisição :ad7711de6bb4432976ea98ecc3f512a52daccfc70af0a700b6d6 214bf47663999244af3395786fbea3cbe530d58d21239d123ca8c4586088f02ed42489ca3b34

Aprovado por:
Prof. Gilberto Medeiros Ribeiro
Diretor da Coordenadoria de Transferência e Inovação Tecnológica
PORTARIA/UFMG/Reitoria/Gabinete Nº 2.225, de 20 de Março DE 2018

ANEXO E – Registro de Programa de Computador – Dashboard Mobi

CTIT Coordenadoria de Transferência e Inovação Tecnológica UFMG

UFMG

Certificado Digital de Registro de Programa de Computador

A Coordenadoria de Transferência e Inovação Tecnológica da Universidade Federal de Minas Gerais expede o presente certificado de registro de programa de computador, válido por 50 anos a partir de 1° de janeiro subsequente à data de sua criação, em conformidade com o §2°, art. 2° da Lei 9.609, de 19 de Fevereiro de 1998.

Número de Registro :20240009
Titular(es) :Universidade Federal de Minas Gerais (UFMG) , IHM Engenharia
Identificação :N-PC-04-2024
Data de Criação :14/02/2024
Título :Visualização dos dados (representações gráficas de lista, gauge e tendências) do Monitoramento de Bombas Industriais Inteligente (MoBI IoT).
Descrição :Apresentar os dados dos assets (bombas industriais) por tipo de variável em representações gráficas de lista, gauge e tendências.
Autor(es) :Marcelo Simões Reis, Cauã Magalhães Pereira, Adriano Borges da Cunha
Linguagem :Javascript, linguagem de programação visual por fluxo de dados do Node-RED e GRAFANA, GRAFANA configuration language River, Notação JSON para representação de nós e fluxos, Influx Query Language (InfluxQL), FLUX (InfluxDB functional data scripting language), TICKscript language
Campo de aplicação :IN01, IN02, IF01
Tipo de programa :GI08
Derivação Autorizada :MIT License, GNU Affero General Public License Version 3 (AGPL)
Expedido em :16/05/2024 21:04:13.756000 UTC (Coordinated Universal Time)
Algoritmo HASH :SHA512
Resumo digital HASH :77b0d115239b7fcf919317c7e906ed6f55b1e5ca119e6ae013df779646503bcf28c8fdf693e46472d8547716d1177abc931bfaaf8d00530eda9dd8d59c79e35c
Resumo digital HASH de requisição :7367579c45148a478c6931cba9acc54c96e2b74060bce07c9dd35be8c8ec4117a662e239bcbb565f0faaacdd98f3181d6471153eb74df6db37133021ee119be0

Aprovado por:
Prof. Gilberto Medeiros Ribeiro
Diretor da Coordenadoria de Transferência e Inovação Tecnológica
PORTARIA/UFMG/Reitoria/Gabinete Nº 2.225, de 20 de Março DE 2018