Alberi Medeiros Santos

Projeto de sistema para interconexão e mapeamento dinâmico entre dispositivos em um cenário de Indústria 4.0

Campina Grande, Paraíba

setembro de 2024

Alberi Medeiros Santos

Projeto de sistema para interconexão e mapeamento dinâmico entre dispositivos em um cenário de Indústria 4.0

Trabalho de Conclusão de Curso submetido à Coordenadoria do Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Engenheiro Eletricista.

Universidade Federal de Campina Grande – UFCG

Centro de Engenharia Elétrica e Informática – CEEI

Departamento de Engenharia Elétrica – DEE

Orientador: Danilo Freire de Souza Santos, Dr.

Campina Grande, Paraíba

setembro de 2024

Alberi Medeiros Santos

Projeto de sistema para interconexão e mapeamento dinâmico entre dispositivos em um cenário de Indústria 4.0

Trabalho de Conclusão de Curso submetido à Coordenadoria do Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Engenheiro Eletricista.

Danilo Freire de Souza Santos, Dr.
Orientador

Gutemberg Gonçalves dos Santos Júnior
Convidado

Campina Grande, Paraíba
setembro de 2024

Agradecimentos

A conclusão desta jornada acadêmica não teria sido possível sem o apoio e a dedicação de diversas pessoas às quais sou imensamente grato.

Primeiramente, agradeço à minha família, que sempre esteve ao meu lado, oferecendo amor, força e incentivo. Ao meu pai Abiderman e à minha mãe Francisca, meu mais profundo reconhecimento pelo esforço e sacrifício em prol do meu crescimento pessoal e profissional.

À minha namorada Larissa, que esteve comigo nos momentos mais desafiadores, oferecendo apoio incondicional, meu mais sincero agradecimento.

Aos meus amigos de longa data, que se tornaram uma extensão da minha família e estiveram presentes desde o início dessa caminhada. Thiago, Jonnas, Léo, Higor, Paloma e tantos outros, agradeço por estarem ao meu lado, fortalecendo nossa amizade e me inspirando a continuar.

Aos amigos do curso de Engenharia Elétrica, que dividiram comigo tanto os desafios quanto as conquistas. Raynton, Alexsandra, Matheus, Mateus, Lucas, Raiff, Raisson, Jan, Djancley, Diego, Anderson, e tantos outros que me acompanharam nos momentos difíceis e contribuíram para o meu desenvolvimento acadêmico e pessoal, meu muito obrigado.

Aos meus professores, cujo conhecimento e dedicação foram fundamentais para minha formação. Agradeço especialmente ao professor Danilo, pela orientação e suporte durante o desenvolvimento deste trabalho. A professora Adriana, o professor Andrey, o professor Eduardo, o professor Reinaldo e o professor Fábio, todos vocês desempenharam um papel essencial na minha trajetória acadêmica e sou grato por todo o aprendizado.

A todos vocês, minha eterna gratidão por terem feito parte desta importante etapa da minha vida.

"Se eu tivesse uma hora para resolver um problema e minha vida dependesse da solução, eu gastaria os primeiros 55 minutos determinando a pergunta certa a se fazer, pois, uma vez que soubesse a pergunta correta, poderia resolver o problema em menos de cinco minutos."

(Albert Einstein)

Resumo

Este trabalho apresenta o desenvolvimento de um sistema para interconexão e mapeamento dinâmico entre dispositivos em um cenário de Indústria 4.0. A proposta é criar uma solução que integre dispositivos industriais, como Controladores Lógicos Programáveis (CLPs), sensores e sistemas de controle, utilizando protocolos de comunicação como OPC UA e Modbus, em conjunto com o *framework* Node-RED. A arquitetura proposta foi testada em um ambiente simulado, utilizando o *software* Factory I/O para a simulação de uma linha de produção, o CODESYS para a programação dos CLPs e o Node-RED para a orquestração da comunicação e monitoramento em tempo real. A integração com a nuvem AWS via protocolo MQTT permitiu a análise e armazenamento de dados críticos do ambiente simulado. O sistema implementado também explorou a utilização de gêmeos digitais e ontologias baseadas no padrão ISA-95 para aprimorar a eficiência e adaptabilidade do processo produtivo. Os resultados indicam que a solução desenvolvida oferece uma comunicação eficiente e flexível, capaz de se adaptar às necessidades de um ambiente industrial moderno, promovendo maior controle e monitoramento das operações em tempo real.

Palavras-chave: Indústria 4.0, interconexão, mapeamento dinâmico, OPC UA, Modbus, Node-RED, gêmeos digitais.

Abstract

This work presents the development of a system for interconnection and dynamic mapping between devices in an Industry 4.0 scenario. The proposed solution integrates industrial devices, such as Programmable Logic Controllers (PLCs), sensors, and control systems, using communication protocols like OPC UA and Modbus, in conjunction with the Node-RED framework. The proposed architecture was tested in a simulated environment using Factory I/O software to simulate a production line, CODESYS for PLC programming, and Node-RED for communication orchestration and real-time monitoring. Integration with AWS cloud via the MQTT protocol enabled the analysis and storage of critical data from the simulated environment. The implemented system also explored the use of digital twins and ontologies based on the ISA-95 standard to enhance the efficiency and adaptability of the production process. The results indicate that the developed solution provides efficient and flexible communication, capable of adapting to the needs of a modern industrial environment, promoting greater control and real-time monitoring of operations.

Keywords: Industry 4.0, interconnection, dynamic mapping, OPC UA, Modbus, Node-RED, digital twins.

Lista de ilustrações

Sumário

1 Introdução

A Indústria 4.0, também conhecida como a quarta revolução industrial, representa uma das principais revoluções tecnológicas dos últimos anos. Ela se caracteriza pela integração de técnicas avançadas de fabricação e da Internet das Coisas (IoT, do inglês *Internet of Things*), com o objetivo de criar sistemas de manufatura interconectados capazes de analisar dados e executar ações inteligentes no mundo físico. Exemplos dessa transformação incluem a operação remota de indústrias, com sistemas capazes de fornecer *feedback* tátil em comandos, como na chamada Internet Tátil (Aazam, Harras e Zeadally 2019).

Entretanto, a implementação dessas tecnologias avançadas apresenta diversos desafios, especialmente na camada inicial de comunicação. Essa camada é crucial para o sucesso da Indústria 4.0 e depende de protocolos robustos, como OPC UA (do inglês *Open Platform Communications - Unified Architecture*) e MQTT (do inglês *Message Queuing Telemetry Transport*) (Marcon et al. 2017). Embora essa diversidade de protocolos seja fundamental para atender às variadas necessidades do ambiente industrial, ela também gera heterogeneidade, dificultando a padronização e a interoperabilidade entre diferentes sistemas.

Os protocolos de comunicação utilizados são essenciais para permitir a troca de dados em tempo real entre Sistemas Ciberfísicos (CPS, do inglês *Cyber-Physical Systems*), a Internet dos Serviços (IoS, do inglês *Internet of Services*) e a própria IoT (Zezulka et al. 2018). Essa interação possibilita a criação de fábricas inteligentes e modulares, nas quais sistemas monitoram processos físicos e criam representações virtuais do mundo físico, como os Gêmeos Digitais (DT, do inglês *Digital Twin*). Esses gêmeos digitais, por sua vez, facilitam decisões descentralizadas, aumentando a eficiência e adaptabilidade da produção (Pribiš, Beňo e Drahoš 2020).

Com a crescente conexão de dispositivos à internet, o uso de *frameworks* para IoT tornou-se indispensável. Conforme observado por Derhamy et al. (2015) e Kabir (2022), esses *frameworks* aprimoram o desenvolvimento de aplicativos IoT, proporcionando rapidez, interoperabilidade, facilidade de manutenção, segurança e flexibilidade tecnológica. Simultaneamente, a necessidade de operações na borda, especialmente em fábricas distribuídas, demanda soluções de computação que garantam aplicações de baixa latência, otimizando a Qualidade de Serviço (QoS, do inglês *Quality of Service*) na produção (**Sisinni2018**).

Apesar desses avanços, ainda existem desafios na integração de dispositivos nas redes industriais, como evidenciado por Shi et al. (2016). O processamento de dados heterogêneos oriundos de diferentes dispositivos na rede de borda é um problema a ser

resolvido. Cada dispositivo segue padrões e normas de comunicação específicos, exigindo um *gateway* na borda capaz de interpretar e converter esses dados para um formato unificado e interoperável. Tal abordagem facilita o desenvolvimento de serviços de processamento na borda, elemento essencial para a infraestrutura inteligente e interconectada da Indústria 4.0.

Além do mapeamento e padronização, é importante que esses dispositivos interoperáveis em *gateways* tenham a capacidade de executar aplicações diretamente em seus ambientes. Essas aplicações devem ser programáveis de forma dinâmica e ágil, permitindo uma configuração adaptativa. *Frameworks* como Node-RED e Calvin IoT oferecem ambientes dinâmicos para a programação dessas aplicações, utilizando modelos e atores para realizar a configuração dinâmica.

Diante desse cenário, o objetivo deste projeto é desenvolver uma solução que permita a comunicação eficiente e a integração dinâmica de dispositivos industriais, incluindo Controladores Lógicos Programáveis (CLPs), sensores, atuadores e sistemas de controle, em um ambiente de manufatura inteligente. A proposta central é criar um sistema capaz de interconectar esses dispositivos e facilitar o mapeamento dinâmico dos recursos e dados na fábrica, promovendo uma integração flexível e adaptável às mudanças do ambiente produtivo.

O projeto visa implementar tecnologias e protocolos de comunicação industriais, como OPC UA e Modbus, em conjunto com *frameworks* de integração como o Node-RED. Isso permitirá a construção de uma infraestrutura que não apenas colete e processe dados em tempo real, mas também mapeie dinamicamente as interações entre os dispositivos. Esse mapeamento dinâmico identificará e organizará as interconexões no chão de fábrica, refletindo alterações nas operações e configurações em tempo real, algo essencial para cenários de produção flexíveis e customizados da Indústria 4.0.

Além disso, o sistema buscará explorar a implementação de gêmeos digitais, representações virtuais dos dispositivos físicos, e a adoção de ontologias de fabricação baseadas em padrões como o ISA-95. Tais elementos proporcionarão uma visão estruturada e padronizada dos processos produtivos, aprimorando a eficiência da interconexão e do mapeamento dos dados. A solução resultante permitirá uma operação mais ágil e eficiente, com a capacidade de adaptação a novas configurações, bem como um maior controle e monitoramento em tempo real das operações industriais.

1.1 Objetivo

O principal objetivo deste trabalho é projetar e desenvolver um sistema eficiente e adaptável para interconexão e mapeamento dinâmico em um cenário de Indústria 4.0. A proposta é criar uma solução que permita a integração de diferentes dispositivos

industriais, utilizando tecnologias e protocolos de comunicação modernos, para otimizar os processos produtivos. Para alcançar esse objetivo geral, o trabalho se divide em uma série de objetivos específicos, cada um deles abordando um aspecto essencial da integração e interoperabilidade em ambientes industriais.

1.1.1 Objetivos específicos

Para desenvolver a solução proposta, foram definidos os seguintes objetivos específicos, cada um contribuindo para o desenvolvimento e validação do sistema integrado:

- Analisar e entender os desafios da comunicação em ambientes industriais heterogêneos, considerando os principais protocolos de comunicação utilizados (OPC UA, MQTT, HTTP, DTDL, Modbus) e suas características. Essa análise é fundamental para identificar as limitações atuais e direcionar as escolhas tecnológicas que suportarão a interconexão eficiente e a troca de informações em tempo real.
- Estudar e aplicar *frameworks* de IoT, como Node-RED, para facilitar a integração de dispositivos que utilizam diferentes protocolos de comunicação. A aplicação desses *frameworks* possibilitará a programação dinâmica e a configuração adaptativa dos dispositivos, tornando o sistema flexível e capaz de se ajustar a diferentes cenários industriais.
- Desenvolver e testar uma infraestrutura de computação na borda, implementando serviços de processamento para suportar aplicações de baixa latência e avaliar o QoS dos processos industriais. Essa infraestrutura permitirá o processamento local dos dados, reduzindo a latência e otimizando o desempenho das operações na fábrica.
- Realizar experimentos e validações práticas em um ambiente simulado que represente uma fábrica inteligente. Através desses experimentos, será possível avaliar a eficiência, interoperabilidade e adaptabilidade do sistema proposto, validando sua capacidade de atender aos requisitos da Indústria 4.0.

1.2 Metodologia

A metodologia deste projeto foi estruturada para o desenvolvimento prático e modular de uma solução completa para interconexão e mapeamento dinâmico em um cenário de Indústria 4.0. A abordagem envolveu a construção de módulos interdependentes, seguindo etapas específicas que visam à implementação, integração e validação das tecnologias necessárias.

O desenvolvimento da solução foi organizado em três módulos principais, que juntos compõem o sistema final. Cada módulo foi projetado, implementado e testado de forma

independente antes da integração completa. As etapas da metodologia foram definidas da seguinte forma:

- **Módulo 1: Comunicação entre dispositivos industriais simulados:** Este módulo teve como objetivo estabelecer a comunicação entre diferentes dispositivos industriais, como CLPs e sensores simulados. As principais atividades deste módulo foram:
  - Implementar a comunicação entre os CLPs simulados utilizando o protocolo OPC UA, configurando uma infraestrutura que permita a troca de informações em tempo real;
  - Configurar sensores industriais simulados para enviar e receber dados através do protocolo MQTT, integrando-os com a plataforma em nuvem AWS IoT;
  - Desenvolver um *dashboard* interativo utilizando o Node-RED para monitoramento em tempo real dos dispositivos, permitindo a análise de desempenho e controle dos processos.
- **Módulo 2: Implementação de gêmeos digitais, Modbus e integração com a nuvem:** Neste módulo, o foco foi na criação de representações virtuais dos dispositivos físicos, interoperabilidade de protocolos, e na comunicação com a nuvem. As etapas incluíram:
  - Configurar sensores simulados para enviar dados ao Azure IoT Hub, estabelecendo um fluxo contínuo de informações entre o ambiente simulado e a nuvem;
  - Implementar gêmeos digitais utilizando DTDL na plataforma Azure Digital Twins, representando os dispositivos e processos industriais;
  - Integrar o protocolo Modbus para comunicação com alguns sensores, ampliando o aspecto de interoperabilidade do sistema e demonstrando a capacidade de trabalhar com diferentes padrões industriais;
  - Integrar os gêmeos digitais e os diferentes protocolos ao Node-RED para possibilitar o monitoramento e controle remoto, permitindo que as informações sejam exibidas e manipuladas em tempo real.
- **Módulo 3: Integração e validação da solução completa:** Com os módulos individuais implementados, este último módulo focou na integração de todos os componentes em um cenário único e na validação da solução:
  - Integrar todos os elementos desenvolvidos em um ambiente industrial simulado, incluindo CLPs, sensores, gêmeos digitais, os protocolos (OPC UA, MQTT, Modbus) e os painéis de monitoramento;

- Validar a eficácia da solução, avaliando aspectos como interoperabilidade, eficiência, adaptabilidade, e a capacidade do sistema de mapear dinamicamente as operações e reconfigurar-se em tempo real;
- Realizar testes de comunicação entre os dispositivos e as plataformas em nuvem, medindo a latência e o throughput, para garantir que o sistema atende aos requisitos de um ambiente de Indústria 4.0.

Essa abordagem metodológica permitiu a construção de uma solução modular e integrada, onde cada módulo foi desenvolvido com um foco específico, mas com o objetivo comum de criar um sistema adaptável e eficiente. Ao final, a integração dos módulos resultou em um ambiente industrial simulado capaz de interconectar e mapear dinamicamente os dispositivos, aprimorando a operação, controle e monitoramento da produção, utilizando múltiplos protocolos de comunicação para aumentar a interoperabilidade do sistema.

2 Fundamentação teórica

A Indústria 4.0 representa uma revolução no modo como os processos industriais são gerenciados, possibilitando a interconexão de sistemas ciberfísicos, Internet das Coisas (IoT), Internet dos Serviços (IoS) e gêmeos digitais para criar um ambiente produtivo inteligente e modular. A implementação desse conceito depende de tecnologias avançadas de comunicação, integração e mapeamento dinâmico, com foco em flexibilidade e adaptabilidade às mudanças no ambiente produtivo.

2.1 Protocolos de Comunicação na Indústria 4.0

A interconexão eficiente entre diferentes dispositivos industriais requer o uso de protocolos robustos que possibilitem a troca de informações em tempo real. Dentre os principais protocolos utilizados na Indústria 4.0, destacam-se o OPC UA, MQTT, Modbus e HTTP.

2.1.1 OPC UA

O padrão OPC UA é baseado em especificações desenvolvidas em estreita cooperação entre fabricantes, usuários, institutos de pesquisa e consórcios, a fim de permitir a troca segura de informações em sistemas heterogêneos. Ele conecta a troca de dados entre dispositivos usando diferentes plataformas e protocolos de comunicação, permitindo escalabilidade e interoperabilidade em sistemas de automação (Rahadian et al. 2022).

Apesar de promover a interoperabilidade, o OPC UA enfrenta desafios devido à sua complexidade, à variedade de formatos de serialização e à implementação seletiva de seus serviços, o que pode resultar em uma heterogeneidade significativa que compromete sua eficácia (Zezulka et al. 2018). Ainda assim, o OPC UA permanece um padrão importante para a interconexão em ambientes industriais, oferecendo suporte a modelos de dados complexos, comunicação criptografada e autenticação, o que o torna uma escolha adequada para sistemas que demandam alta segurança e integração flexível com gêmeos digitais.

2.1.2 MQTT

O MQTT é um protocolo de rede leve, baseado em um modelo de publicação e assinatura, projetado para comunicação eficiente em redes com largura de banda limitada e alta latência. Criado em 1999, tornou-se um padrão amplamente utilizado em sistemas de IoT devido à sua leveza e eficiência. Sua arquitetura de publicação/assinatura permite uma transmissão de dados rápida e confiável, mesmo em condições adversas de rede.

O MQTT oferece diferentes níveis de QoS, dependendo das condições da rede, das características do dispositivo e da criticidade da aplicação (Enany, Harb e Attiya 2021):

- **QoS 0**: A mensagem é entregue no máximo uma vez, sem qualquer confirmação de recebimento;
- **QoS 1**: A mensagem é entregue pelo menos uma vez. O remetente mantém a mensagem armazenada até receber uma confirmação de recebimento;
- **QoS 2**: A mensagem é entregue exatamente uma vez, sem duplicação.

Essa flexibilidade faz do MQTT uma opção versátil para diferentes cenários industriais, desde aplicações com requisitos rigorosos de tempo real até ambientes onde a eficiência de comunicação é primordial.

2.1.3 Modbus

O Modbus é um protocolo de comunicação amplamente utilizado em dispositivos industriais, especialmente em equipamentos legados, devido à sua simplicidade e robustez (Elamanov et al. 2024). Sua adoção fácil por parte das organizações industriais e fabricantes de dispositivos tornou o Modbus uma escolha popular em sistemas de automação.

No contexto da Indústria 4.0, a integração do Modbus com protocolos mais recentes, como OPC UA e MQTT, é fundamental para promover a interoperabilidade entre equipamentos antigos e novos. No entanto, a natureza isolada do Modbus apresenta desafios quando se trata de conectá-lo a aplicações IoT que utilizam diferentes protocolos. Para superar isso, um mecanismo de interworking é necessário para abstrair os dados do Modbus e mapeá-los para uma camada de serviços IoT, como o padrão oneM2M (Elamanov et al. 2024).

Essa abordagem permite que os sistemas legados continuem operando em conjunto com dispositivos modernos, contribuindo para uma migração mais gradual das plantas industriais para infraestruturas inteligentes e conectadas. A capacidade do sistema de lidar com múltiplos protocolos, incluindo Modbus, OPC UA e MQTT, demonstra sua flexibilidade e adaptabilidade, aspectos essenciais para ambientes industriais complexos.

2.1.4 HTTP

HTTP (do inglês *Hypertext Transfer Protocol*) é um protocolo de comunicação usado para transmitir dados pela Internet. É um protocolo fundamental que permite a troca de informações entre servidores web e clientes. No contexto da Indústria 4.0, o HTTP contribui ao permitir a comunicação entre Sistemas Ciberfísicos em sistemas de manufatura (Peralta-Abarca, Martínez-Bahena e Enríquez-Urbano 2021).

Os Sistemas Ciberfísicos utilizam protocolos tradicionais de Internet, como HTTP, para comunicação integrando processos de automação industrial com tecnologias de informação (Iglesias-Urkia et al. 2017). No entanto, o uso de HTTP apresenta limitações como alto consumo de energia, menor eficiência de transmissão e uso ineficiente da largura de banda, comprometendo a interação fluída entre humano e máquina (Iglesias-Urkia et al. 2017), (Ferrer et al. 2015). Para superar esses desafios, protocolos alternativos como MQTT, AMQP (do inglês *Advanced Message Queuing Protocol*) e CoAP (do inglês, *Constrained Application Protocol*) são adotados devido à sua eficiência e adequação aos sistemas automatizados (**ref10**). A integração desses novos protocolos, contudo, exige conhecimento técnico e tempo.

2.2 Ontologias e Padrão ISA-95 na Indústria 4.0

Para alcançar a interoperabilidade e a integração dinâmica entre os diversos componentes em um cenário de Indústria 4.0, o uso de ontologias e padrões industriais é fundamental. Ontologias fornecem um modelo conceitual comum que permite que diferentes sistemas entendam e compartilhem informações de forma consistente. No contexto da manufatura, uma ontologia pode ser usada para definir as relações entre máquinas, processos, produtos e fluxos de trabalho, contribuindo para um mapeamento dinâmico e padronizado dos recursos (Digital Twin Consortium 2024).

O padrão ISA-95 é um modelo internacional para a integração de sistemas de controle de produção com sistemas corporativos. Ele estabelece uma estrutura hierárquica que permite descrever e organizar os elementos da fábrica, desde o chão de fábrica (nível operacional) até o nível de negócio. Sua estrutura foi padronizada pela IEC 62264, garantindo que seja amplamente reconhecido e aplicável em diversos setores industriais (*ISA-95* 2024) (*IEC 62264* 2024). A adoção de ontologias baseadas no padrão ISA-95 ajuda a padronizar a comunicação entre diferentes sistemas, facilitando a integração entre CLPs, sensores, sistemas de controle e plataformas em nuvem.

No projeto, a implementação de uma ontologia de fábrica seguindo o padrão ISA-95 permite descrever os dispositivos, processos e fluxos de informações de maneira estruturada. Essa padronização é crucial para o mapeamento dinâmico dos recursos, proporcionando uma visão unificada do ambiente produtivo e permitindo que os gêmeos digitais reflitam com precisão as operações em tempo real.

2.3 Gêmeos Digitais e DTDL na Indústria 4.0

Os gêmeos digitais são representações virtuais de dispositivos e processos físicos que permitem monitoramento, simulação e análise em tempo real. Eles desempenham um papel

fundamental na otimização das operações e na manutenção preditiva na Indústria 4.0, ao fornecer uma visão estruturada e detalhada do ambiente produtivo. A integração dos gêmeos digitais com protocolos como OPC UA e MQTT possibilita uma comunicação bidirecional entre os sistemas físicos e virtuais, promovendo a tomada de decisões descentralizadas e a automação inteligente.

No contexto da plataforma Azure Digital Twins, os gêmeos digitais são projetados para modelar ambientes físicos inteiros, incluindo a infraestrutura de fábricas e seus processos operacionais. Eles possibilitam a coleta, processamento e análise de dados em tempo real, facilitando a detecção de anomalias e a previsão de manutenção de equipamentos (Microsoft Azure 2024).

2.3.1 DTDL

A Linguagem de Definição Digital de Gêmeos (DTDL, do inglês *Digital Twin Definition Language*) é um aspecto crucial no desenvolvimento e implementação de Gêmeos Digitais (DTs, do inglês *Digital Twins*) em vários setores. Os DTs servem como réplicas virtuais de sistemas físicos, permitindo a troca de dados em tempo real e processos de otimização (Bibow et al. 2020) (**ref13**). O DTDL desempenha um papel significativo na definição do comportamento e das interações dos Sistemas de Produção Ciber-Físicos (CPPSs, do inglês *Cyber-Physical Production Systems*) dentro da estrutura do DT, facilitando a automação de atividades essenciais de desenvolvimento e a personalização com base em domínios específicos (Singh et al. 2021). Além disso, o DTDL ajuda a estabelecer estruturas de comunicação para DTs, utilizando computação em nuvem, inteligência artificial e comunicações sem fio para aprimorar a operação e os mecanismos de *feedback* dos DTs em aplicações práticas (Luan et al. 2021). Ao incorporar o DTDL, as indústrias podem aproveitar o poder dos DTs para monitorar, otimizar, manter e tomar decisões, impulsionando a eficiência e a inovação em vários setores (González et al. 2020).

2.4 *Frameworks* de IoT para Interconexão e Mapeamento Dinâmico

A programação dinâmica e a configuração adaptativa são essenciais para a interconexão eficiente em ambientes industriais complexos. *Frameworks* de IoT, como o Node-RED, desempenham um papel importante na implementação dessas características. O Node-RED é uma ferramenta de desenvolvimento baseada em fluxo que permite a integração de diferentes protocolos de comunicação, sensores, atuadores e sistemas de controle (Node-RED 2024). No projeto desenvolvido, o Node-RED foi utilizado para orquestrar a comunicação entre dispositivos e plataformas em nuvem, possibilitando o mapeamento dinâmico das operações industriais.

Além disso, o Node-RED facilita o desenvolvimento de *dashboards* interativos

para monitoramento e controle em tempo real, fornecendo uma interface gráfica acessível para operadores e gerentes de produção. Através dessa plataforma, foi possível integrar protocolos diversos, como OPC UA, MQTT e Modbus, demonstrando a capacidade do sistema de trabalhar com múltiplos padrões industriais e aumentando sua interoperabilidade e eficiência.

3 Comunicação entre dispositivos industriais simulados

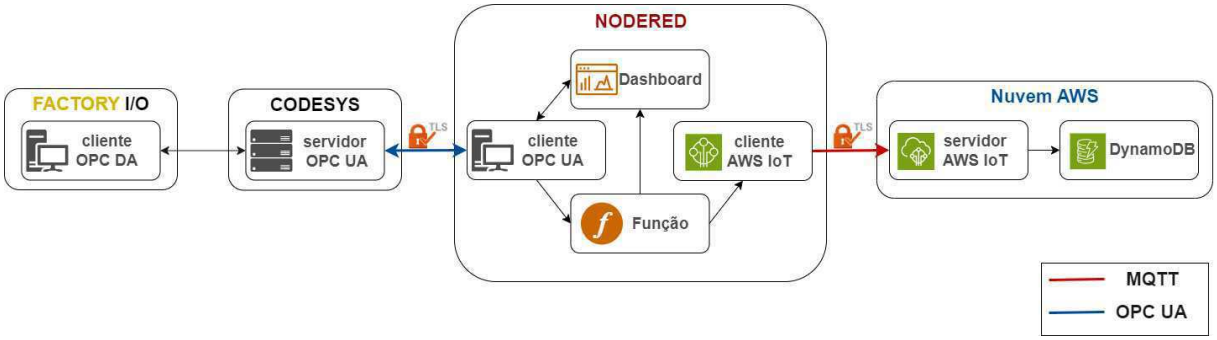
A comunicação eficiente entre sistemas heterogêneos, que muitas vezes utilizam protocolos de comunicação distintos como OPC UA, MQTT e HTTP, é um desafio central na Indústria 4.0. A necessidade de monitoramento em tempo real, análise de dados, e resposta rápida às mudanças no ambiente industrial exige soluções que garantam não apenas a conectividade, mas também a segurança e a flexibilidade operacional

Para abordar esses desafios, o primeiro microprojeto foi desenvolvido com o objetivo de criar um cenário simulado de fábrica que possibilite a comunicação segura entre diferentes dispositivos e sistemas de automação. Este cenário é um protótipo de uma arquitetura de integração que utiliza a combinação de tecnologias consolidadas na indústria, como o OPC UA, para a comunicação no chão de fábrica, e o MQTT, para a transmissão de dados para a nuvem.

A escolha do OPC UA se deu pela sua robustez e capacidade de suportar comunicação segura e padronizada entre dipositivos industriais, enquanto o MQTT foi selecionado pela sua leveza e eficiência em ambientes de IoT, sendo ideal para integração com plataformas de nuvem como a AWS (do inglês, *Amazon Web Service*). Além disso, o Node-RED foi utilizado como plataforma de integração central, devido à sua flexibilidade e capacidade de orquestrar a comunicação entre diferentes protocolos e dispositivos, além de facilitar a criação de dashboards para monitormaneto em tempo real.

A Figura 1 ilustra a arquitetura proposta para o primeiro microprojeto. Nesta arquitetura, o Factory I/O simula o ambiente industrial, enquanto o CODESYS atua como servidor OPC UA, permitindo a comunicação entre o ambiente simulado e o sistema de controle. O Node-RED orquestra a integração, conectando-se ao servidor OPC UA no CODESYS e transmitindo os dados para a nuvem AWS via MQTT. Para garantir a segurança das comunicações, tanto o OPC UA quanto o MQTT foram configurados com criptografia TLS, assegurando a integridade e confidencialidade dos dados transmitidos.

Figura 1 – Arquitetura proposta para o primeiro microprojeto.



Fonte: Elaborado pelo autor.

3.1 Materiais e Métodos

Para a realização deste microprojeto, os seguintes materiais e ferramentas foram utilizados:

3.1.1 Factory I/O

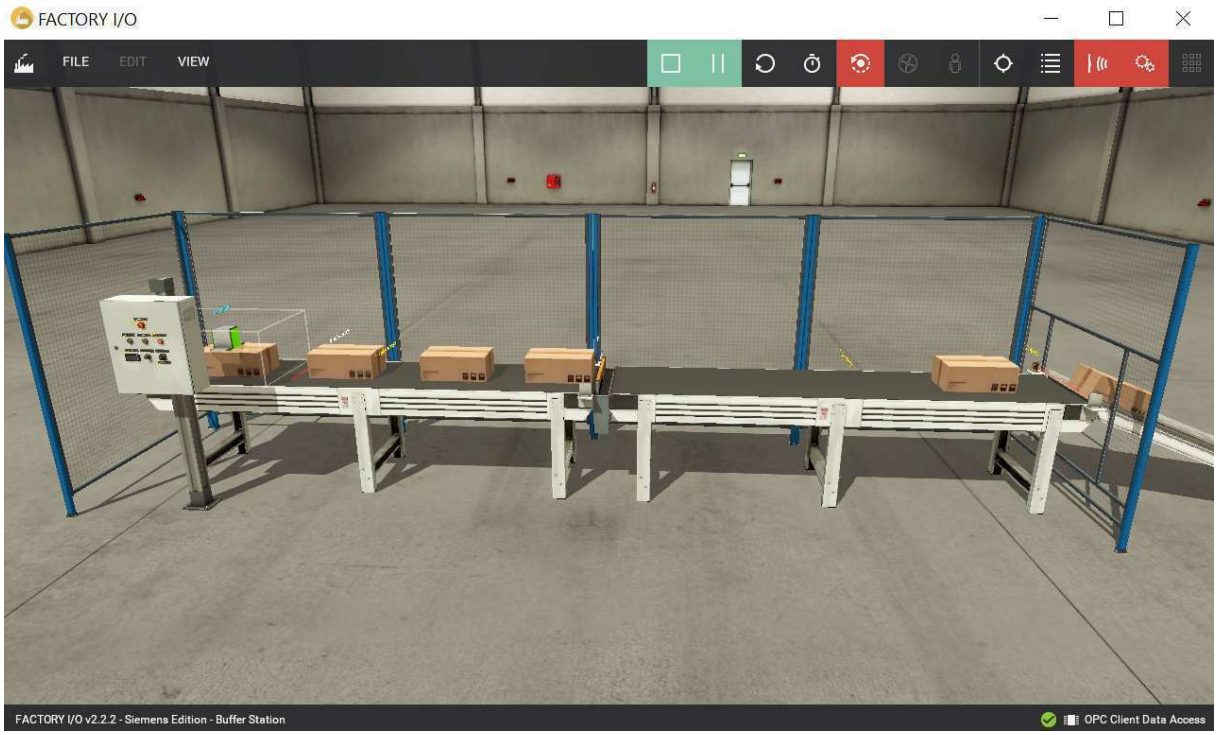
O **Factory I/O** é um software de simulação que permite criar cenários industriais virtuais, simular o funcionamento de uma fábrica e gerar dados em tempo real. No contexto deste microprojeto, foi utilizada a cena *"Buffer Station"*, uma cena pré-configurada e disponível no próprio software, que simula um processo de transporte e armazenamento temporário de peças em uma linha de produção. Essa cena foi escolhida por representar um ambiente industrial típico, permitindo testar a comunicação e integração dos sistemas.

Componentes da cena *"Buffer Station"*:

- **Esteira de Entrada (*Buffer Conveyor*)**: Responsável por mover as peças para a estação de buffer.
- **Esteira de Saída (*Exit Conveyor*)**: Transporta as peças para fora da estação de buffer, enviando-as para a próxima etapa do processo.
- **Sensores de Proximidade (*Emitter e Remover*)**: Detectam a presença de peças na entrada e na saída da estação de buffer, permitindo o controle do fluxo de materiais.
- **Velocidade do Buffer (*Buffer Vel.*)**: Controla a velocidade com que a esteira de buffer opera.
- **Contador (*Counter*)**: Registra o número de peças que passaram pela estação de buffer.

- **Botão de Emergência (*Emergency Stop*)**: Permite parar a operação em caso de emergência.
- **Botões de Controle (*Manual, Auto, Start, Stop*)**: Controlam o modo de operação da estação de buffer, permitindo iniciar, parar ou alternar entre controle manual e automático.
- **Luzes Indicadoras (*Start Light, Stop Light, Reset Light*)**: Indicadores visuais que mostram o estado operacional da estação de buffer.
- **Lâmina de Parada (*Blade Stop*)**: Um mecanismo que impede a passagem de peças na esteira, controlando o fluxo de materiais conforme necessário.

Figura 2 – Cenário *"Buffer Station"* no Factory I/O, ilustrando a operação dos componentes simulados.



Fonte: Elaborado pelo autor.

A Tabela 1 demonstra a quantidade e a variedade de dispositivos presentes na cena "Buffer Station". A estação, composta por 15 dispositivos distintos, simula um ambiente industrial real, onde a integração de diversos componentes é fundamental para o funcionamento eficiente de uma linha de produção. A análise da conexão desses dispositivos, apresentada na tabela, é crucial para a compreensão do comportamento do sistema como um todo.

Tabela 1 – Dispositivos conectados no cenário "Buffer Station"

Dispositivo	Quantidade
Esteira de Entrada (Buffer Conveyor)	1
Esteira de Saída (Exit Conveyor)	1
Sensores de Proximidade (Emitter e Remover)	2
Velocidade do Buffer (Buffer Vel.)	1
Contador (Counter)	1
Botão de Emergência (Emergency Stop)	1
Botões de Controle (Manual, Auto, Start, Stop)	4
Luzes Indicadoras (Start Light, Stop Light, Reset Light)	3
Lâmina de Parada (Blade Stop)	1
Total	**15**

Fonte: Elaborado pelo autor.

3.1.2 CODESYS

O **CODESYS** é um ambiente de desenvolvimento integrado (IDE) amplamente utilizado em automação industrial para a programação de controladores lógicos programáveis (CLPs) físicos e virtuais. No contexto deste microprojeto, o CODESYS foi escolhido por sua compatibilidade com o protocolo OPC UA, que foi utilizado como protocolo de comunicação entre o CLP virtual e outros componentes do sistema.

O desenvolvimento no CODESYS envolveu a criação de um servidor OPC UA para facilitar a comunicação entre o CLP virtual e os cliente OPC UA. O processo de implementação realizado é descrito a seguir:

3.1.2.1 Projeto no CODESYS

Um novo projeto standard criado no CODESYS 3.5.11, utilizando o dispositivo programável **CODESYS *Control Win V3***, que é um CLP virtual desenvolvido pela *3S Smart Software Solutions GmbH*.

O projeto foi configurado para incluir um programa **PLC_PRG** na linguagem de diagrama ladder (LD), conforme ilustrado na Figura 3. Este programa foi responsável pela lógica de controle básica, que inclui várias funções essenciais para o controle da operação das esteiras e contagem de peças no cenário simulado.

A programação ladder implementada no PLC_PRG é composta por diferentes blocos de funções:

- Controle e movimentação das esteiras: Os motores das esteiras são monitorados em tempo real, e feedbacks operacionais são gerados para verificar se os motores estão funcionando corretamente, possibilitando a identificação de falhas de operação.

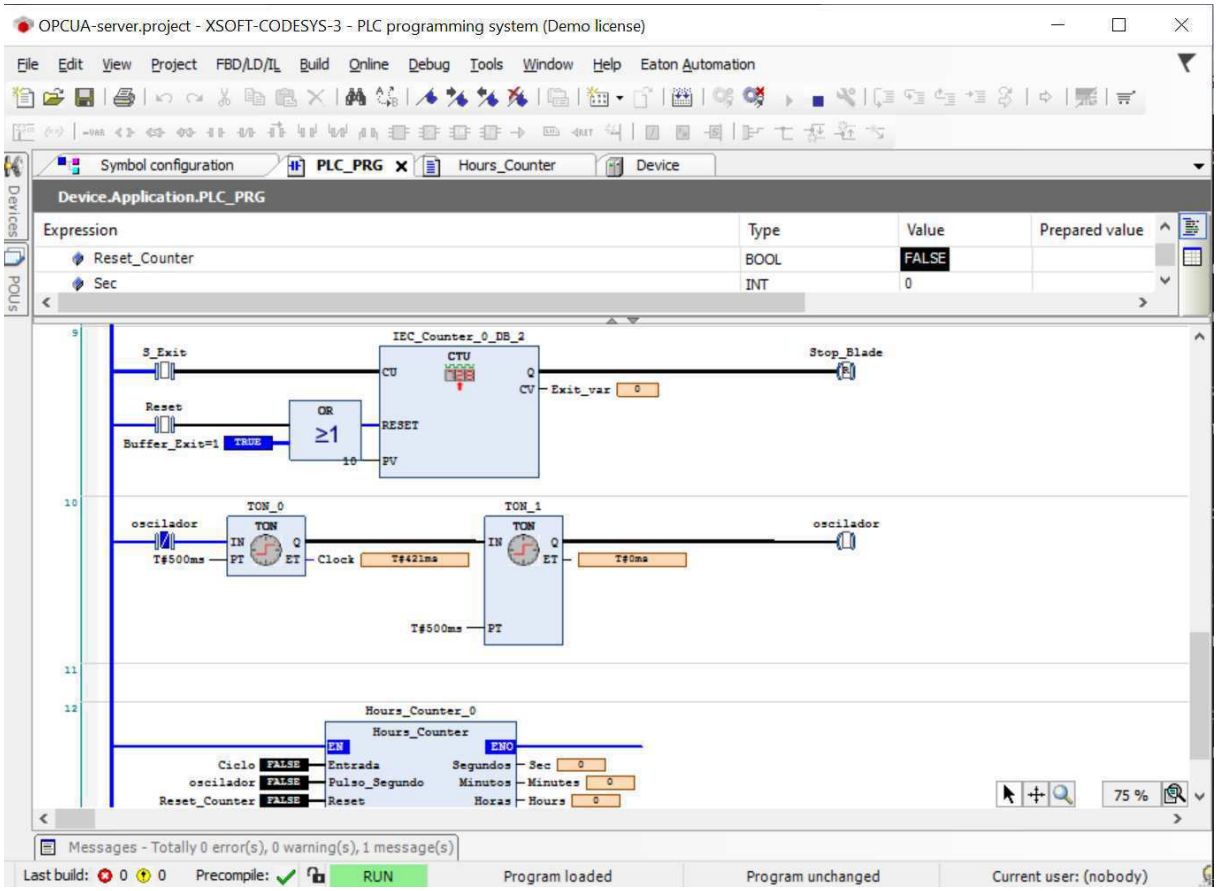
- Monitoramento e contagem de peças: Esses contadores são essenciais para manter um controle preciso sobre o fluxo de materiais, utilizando sensores de proximidade para incrementar os contadores a cada peça detectada. A lógica é configurada para permitir o reset dos contadores quando necessário, garantindo que o sistema possa ser reinicializado para novas execuções.

- Controle de segurança e parada de emergência: O controle de parada de emergência foi implementado utilizando blocos de lógica que desativam imediatamente as esteiras ao detectar um sinal de parada. A lógica de segurança garante que a operação possa ser interrompida a qualquer momento, preservando a integridade dos equipamentos e a segurança dos operadores.

- Monitoramento de parâmetros operacionais: Variáveis para monitoramento de corrente e rotação de forma que qualquer desvio desses parâmetros gera alertas automáticos que poder ser utilizados para manutenção preditica ou tomada de decisão em tempo real.

- Horímetro e registro de tempo de operação: Um horíemtro foi implementado para registrar o tempo total de operação dos sistemas, permitindo rastrear quanto tempo as esteiras e motores estiveram ativos. O horímetro acumula segundos, minutos e horas, proporcionando uma visão clara da utilização do sistema.

Figura 3 – Interface do CODESYS onde o programa foi implementado.



Fonte: Elaborado pelo autor.

3.1.2.2 Configuração do Servidor OPC UA

Para permitir a comunicação via OPC UA, foi adicionado ao projeto o objeto ***Symbol Configuration***, que prepara as variáveis do programa **PLC_PRG** para serem acessíveis externamente via protocolo OPC UA. Nesta etapa, também foram configurados os direito de acesso (*Access Rights*) para garantir que as variáveis fossem expostas ao nível adequado de segurança.

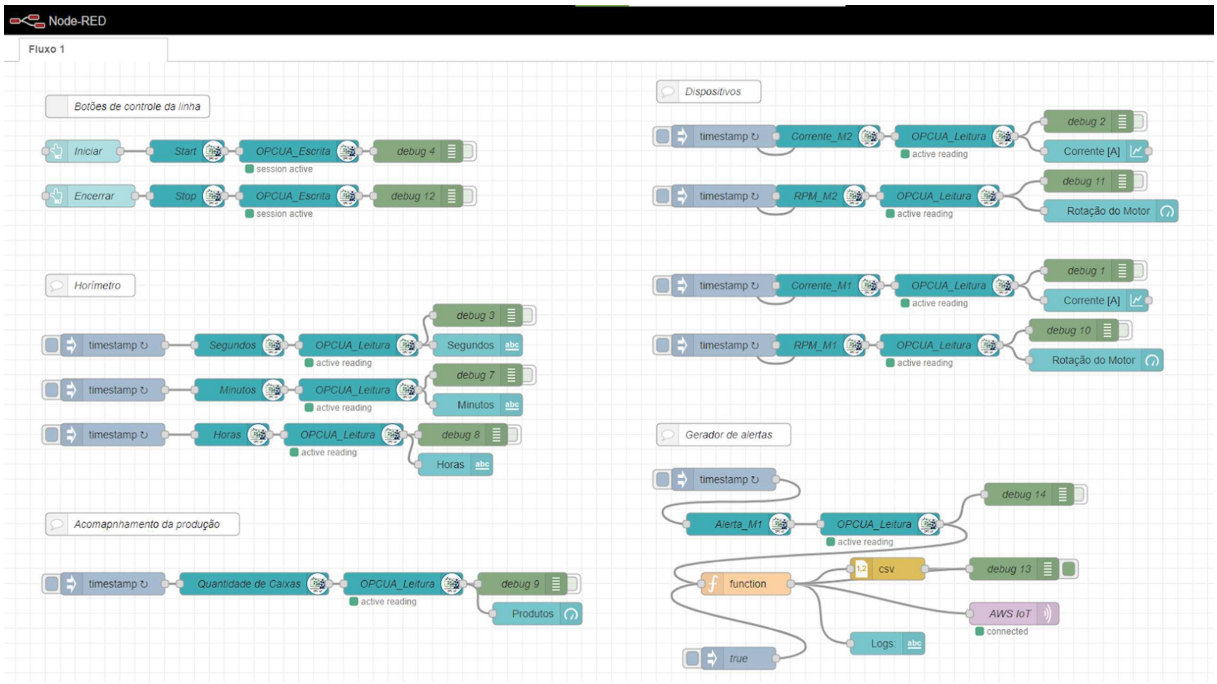
Para aumentar a segurança do sistema, um certificado digital foi configurado utilizando a extensão do CODESYS ***Security Agent***. Este Certificado é essencial para permitir a comunicação criptografada e autenticada entre o servidor OPC UA e seus clientes.

3.1.3 Node-RED

O **Node-RED** foi escolhido como a plataforma principal para orquestrar a comunicação entre o servidor OPC UA configurado no CODESYS e outros componentes do sistema, incluindo a integração com a nuvem AWS via MQTT. A Figura 4 ilustra o

fluxo completo configurado no Node-RED, incluindo a leitura de variáveis via OPC UA, a geração de alertas, e a publicação de dados na nuvem AWS.

Figura 4 – Fluxo completo configurado no Node-RED, mostrando a integração das diferentes funcionalidades como controle da linha, monitoramento de dispositivos e publicação de dados na AWS.



Fonte: Elaborado pelo autor.

O fluxo mostrado na Figura 4 é dividido em diferentes seções funcionais: controle da linha, monitoramento do horímetro, acompanhamento da produção, monitoramento dos dispositivos e geração de alertas. Cada uma dessas seções desempenha um papel crucial na operação do sistema:

- **Botões de Controle da Linha**: Configurados para iniciar e encerrar operações no ambiente simulado, enviando comandos ao CLP via OPC UA.
- **Horímetro**: Realiza a leitura periódica dos valores de tempo de operação (segundos, minutos e horas), também via OPC UA, e exibe esses valores em tempo real.
- **Acompanhamento da Produção**: Monitora a quantidade de produtos processados ao longo do tempo, permitindo o rastreamento da eficiência produtiva.
- **Monitoramento dos Dispositivos**: Leitura de dados operacionais dos dispositivos, como corrente elétrica e rotação dos motores, que são essenciais para avaliar o desempenho e a saúde do sistema.

- **Geração de Alertas**: Um nó de função é utilizado para processar os dados e gerar logs de eventos críticos, como falhas em dispositivos. Esses logs são então publicados na nuvem AWS para armazenamento e análise posterior.

O principal objetivo ao utilizar o Node-RED neste microprojeto foi facilitar a integração e a comunicação entre o CLP virtual no CODESYS e a nuvem AWS, além de fornecer uma interface gráfica para monitoramento e controle do processo industrial simulado. A seguir, a descrição do processo de configuração e os principais componentes utilizados:

3.1.3.1 Instalação de pacotes necessários

Para suportar a comunicação OPC UA, foi necessário instalar o pacote **node-red-contrib-opcua**, que fornece os nós necessários para configurar um cliente OPC UA dentro do Node-RED. Esse pacote foi essencial para conectar o Node-RED ao servidor OPC UA rodando no CODESYS.

Para a criação de *dashboards* interativos que permitam visualizar e controlar os dados em tempo real, foi utilizado o pacote **node-red-dashboard**. Este pacote adiciona uma série de nós que facilitam a construção de interfaces gráficas, como gráficos, botões, indicadores de status, e muito mais, sem a necessidade de codificação extensiva. Esse pacote permitiu criar a interface gráfica para monitorar a fábrica, visualizar dados coletados, e até mesmo enviar comandos de controle, tudo de forma centralizada e acessível.

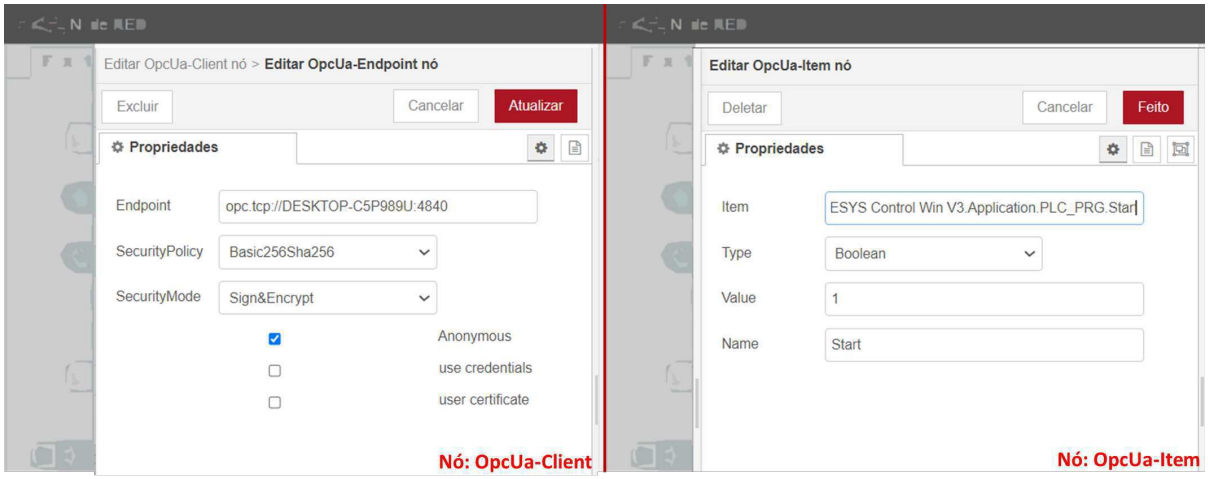
3.1.3.2 Configuração do cliente OPC UA

Com o pacote OPC UA instalado, foi configurado um nó **opcua-client** no Node-RED. Esse nó foi configurado para se conectar ao servidor OPC UA utilizando o endpoint fornecido pelo CODESYS, geralmente no formato *opc.tcp://<ip-do-servidor>:4840*.

Durante a configuração, foi necessário habilitar as políticas de segurança para comunicação assinada e criptografada (por exemplo, Basic256Sha256), conforme configurado anteriormente no CODESYS. Isso assegura que a comunicação entre o Node-RED e o servidor OPC UA seja segura e confiável.

Um segundo nó, **opcua-item**, foi configurado para acessar variáveis específicas no servidor OPC UA. Os *NodeIDs* das variáveis de interesse foram identificados, geralmente no formato *ns=4;s=/var/CODESYS Control Win V3.Application.PLC_PRG.<nome-da-variavel>*, permitindo que o Node-RED pudesse monitorar e alterar esses valores remotamente.

Figura 5 – Configuração do cliente OPC UA.



Fonte: Elaborado pelo autor.

3.1.3.3 Configuração do Nó de Função para Geração de Logs de Eventos Críticos

Utilizou-se um nó de função no Node-RED, desenvolvido para gerar e configurar logs detalhados sempre que um evento crítico é detectado, como falhas em dispositivos ou interrupções no processo produtivo.

O nó de função configurado no Node-RED é projetado para processar as mensagens recebidas e, com base em determinadas condições, gerar logs contendo informações essenciais sobre o evento ocorrido. Esses logs são então encaminhados para visualização no dashboard ou armazenados para análise posterior. A função do nó realiza as seguintes operações:

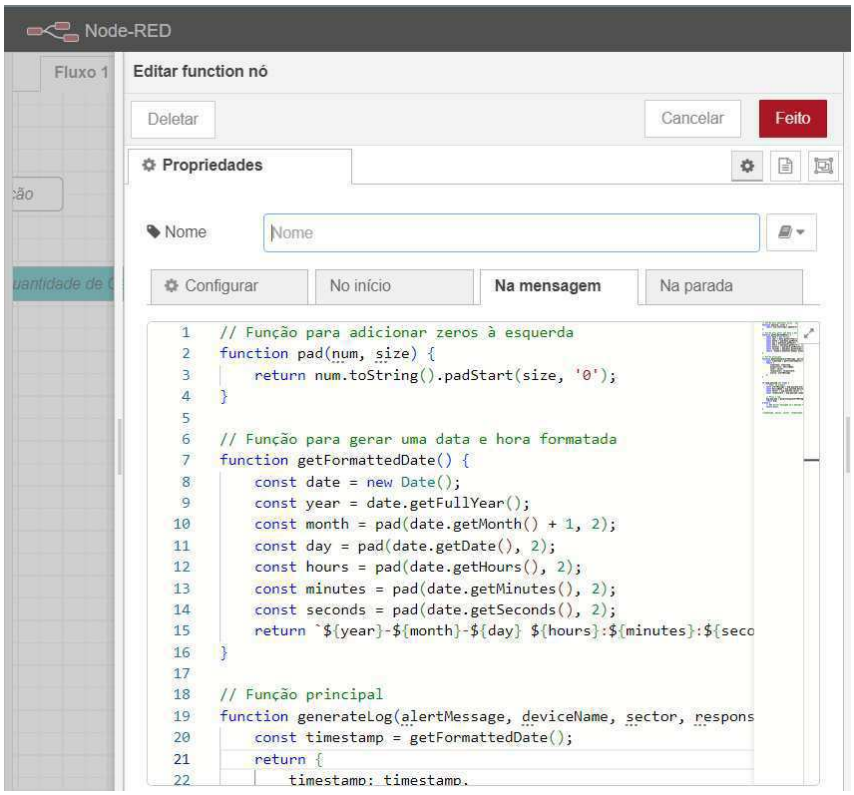
- Formatação de data e hora;
- Geração de log;
- Processamento condicional.

Os logs gerados por este nó de função são fundamentais para a supervisão e análise de eventos críticos no sistema. Eles são armazenados na nuvem e exibidos em um painel específico do dashboard do Node-RED. A Figura 6 ilustra a configuração da função.

3.1.3.4 Monitoramento e Controle via Node-RED

Foram configurados nós adicionais para monitorar e exibir os dados em tempo real através de um **Dashboard** interativo, conforme ilustrado nas Figuras 7 e 8. Esse Dashboard permite uma visualização intuitiva do status das variáveis monitoradas, como contadores de peças, status das esteiras, rotações dos motores, correntes elétricas, e o

Figura 6 – Nó de função no Node-RED configurado para gerar logs de eventos críticos com informações detalhadas.



Fonte: Elaborado pelo autor.

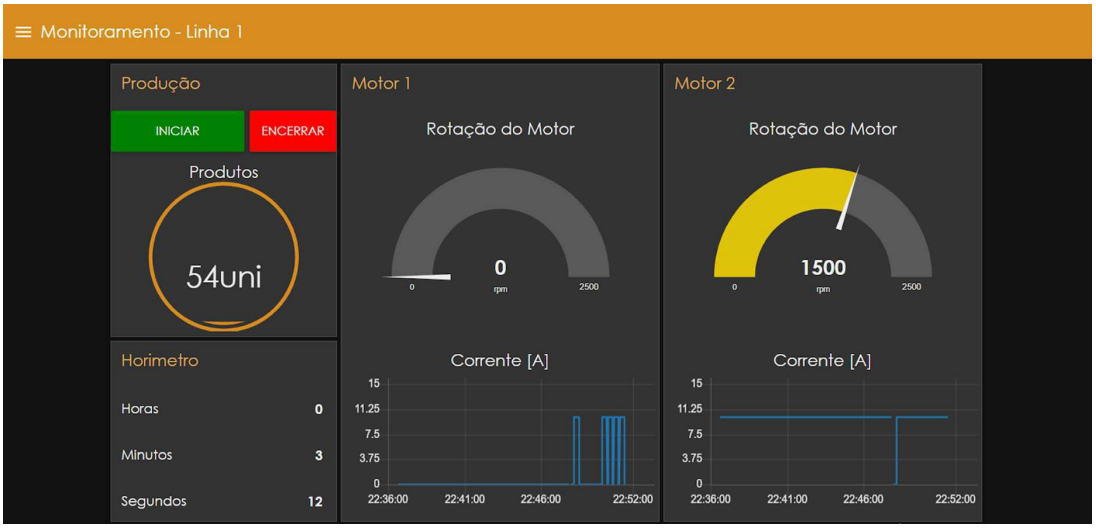
tempo de operação (horímetro). Cada um desses elementos é atualizado em tempo real, proporcionando uma interface de monitoramento eficiente e acessível.

O dashboard foi desenvolvido utilizando o pacote **node-red-dashboard**, o que possibilita que ele seja operado em diferentes plataformas, incluindo computadores e dispositivos móveis, desde que se tenha acesso ao endpoint do dashboard. Isso garante flexibilidade na gestão e supervisão do sistema, permitindo que operadores acessem e controlem o processo de qualquer lugar.

O sistema registra logs de eventos críticos, como paradas de dispositivos, com detalhes sobre o dispositivo afetado, o setor, e o responsável pelo equipamento, conforme demonstrado na Figura 8. Esses logs são apresentados em um formato de fácil leitura e podem ser utilizados para análise posterior e tomada de decisões.

Além do monitoramento, também foi realizada a configuração para permitir o controle remoto das operações do CLP. Através do dashboard, os operadores podem interagir com o sistema em tempo real, executando comandos como iniciar e parar as esteiras, ajustar parâmetros operacionais, e monitorar a eficiência do processo produtivo. Esse nível de controle remoto é fundamental para ambientes industriais que exigem supervisão constante e respostas rápidas a mudanças nas condições operacionais.

Figura 7 – Dashboard do Node-RED exibindo o monitoramento em tempo real de variáveis operacionais como contagem de produtos, rotações dos motores, correntes elétricas, e horímetro.



Fonte: Elaborado pelo autor.

Figura 8 – Dashboard do Node-RED mostrando logs de eventos críticos e detalhes do dispositivo afetado.

Fonte: Elaborado pelo autor.

3.1.3.5 Integração com a Nuvem AWS

Um dos principais objetivos do microprojeto foi a integração com a nuvem AWS para possibilitar o armazenamento e a análise dos dados coletados. Para isso, foi configurado um cliente MQTT no Node-RED, utilizando o nó **mqtt out**. Este nó foi configurado para publicar os dados gerados no ambiente industrial simulado em um tópico específico no AWS IoT, permitindo que essas informações fossem posteriormente armazenadas no DynamoDB para análise e processamento, conforme mostrado na Figura 9 (a).

O protocolo MQTT segue uma arquitetura de publicação/assinatura (*publish/subscribe*), onde os dispositivos atuam como clientes e se comunicam por meio de um broker, que no contexto deste projeto é o AWS IoT Core. No AWS IoT, o broker é responsável por receber as mensagens publicadas pelos dispositivos no ambiente industrial e distribuí-las para outros clientes que se inscreveram nos tópicos relevantes. Essa arquitetura garante a

separação entre quem envia os dados (*publisher*) e quem os recebe (*subscriber*), permitindo uma escalabilidade e flexibilidade na comunicação.

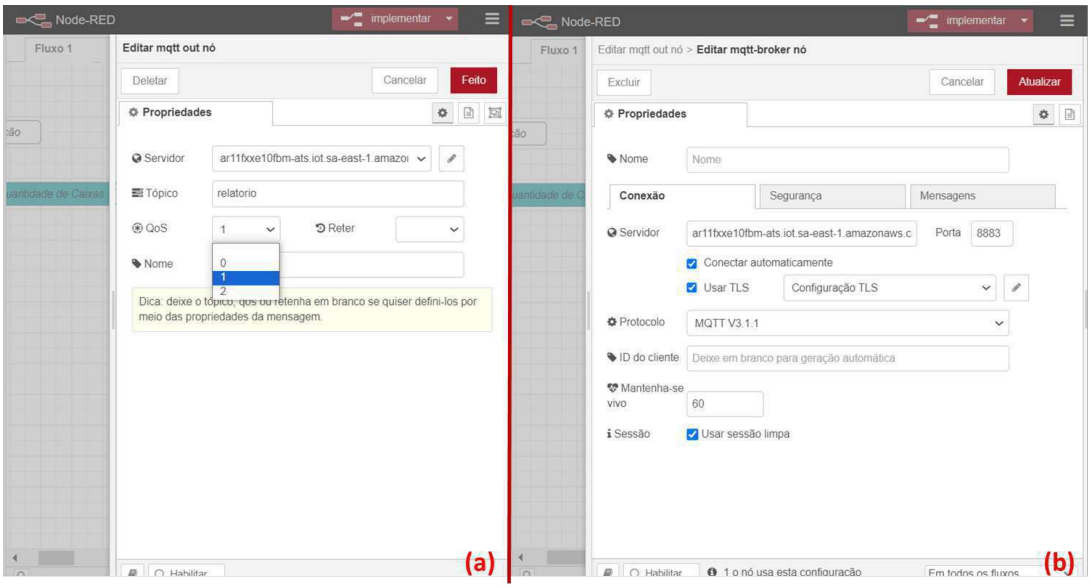
A escolha do MQTT para o envio de dados para a nuvem se deu pela necessidade de uma solução leve e eficiente, capaz de operar em ambientes com baixa largura de banda, como é comum em implementações de IoT. No AWS IoT Core, o broker MQTT é responsável por garantir a entrega de mensagens, utilizando diferentes níveis de Qualidade de Serviço (QoS), que controlam o número de tentativas de entrega das mensagens e garantem a confiabilidade da comunicação, mesmo em condições de rede instáveis.

A configuração do cliente MQTT envolveu a definição do servidor AWS IoT, a especificação do tópico de publicação, e a escolha do nível de Qualidade de Serviço (QoS). Como ilustrado na Figura 9 (a), o nó foi configurado para utilizar um QoS de nível 1, garantindo que as mensagens sejam entregues pelo menos uma vez.

A comunicação com o AWS IoT via MQTT é assegurada por certificados digitais e criptografia TLS (Transport Layer Security), garantindo que apenas dispositivos autenticados possam se conectar ao broker e que as mensagens transmitidas sejam protegidas contra interceptação e alterações. Então, a integração com o AWS IoT também envolveu a configuração de políticas de segurança, incluindo a utilização de certificados digitais para autenticar a comunicação entre o Node-RED e a AWS.

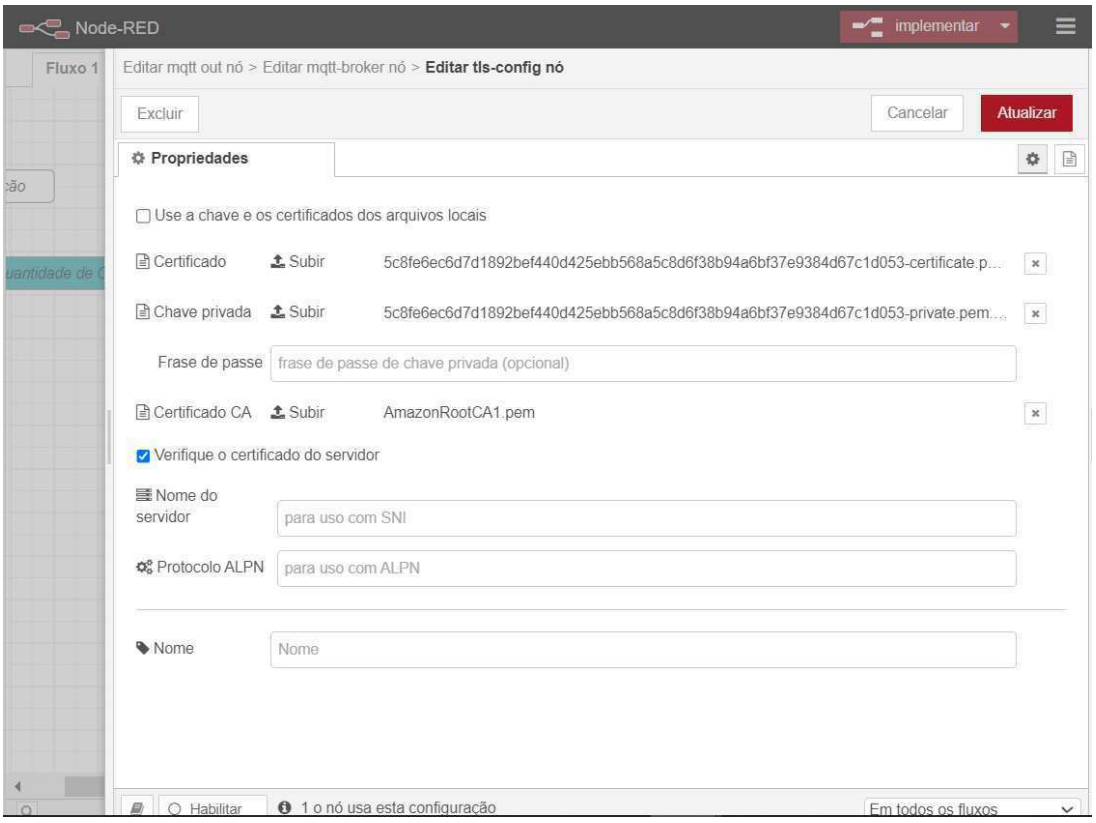
A Figura 10 mostra a configuração de segurança, onde foram carregados os certificados necessários, como o certificado de cliente, a chave privada, e o certificado da autoridade certificadora (CA). Essa configuração foi crucial para assegurar que a comunicação entre o Node-RED e a AWS fosse criptografada usando TLS (Transport Layer Security), protegendo assim a integridade e a confidencialidade dos dados transmitidos.

Figura 9 – (a) Configuração do nó **mqtt out** no Node-RED, utilizado para publicar dados no AWS IoT com QoS nível 1. (b) Configuração do broker MQTT.



Fonte: Elaborado pelo autor.

Figura 10 – Configuração de segurança TLS para o cliente MQTT no Node-RED, incluindo o upload de certificados para comunicação segura com o AWS IoT.



Fonte: Elaborado pelo autor.

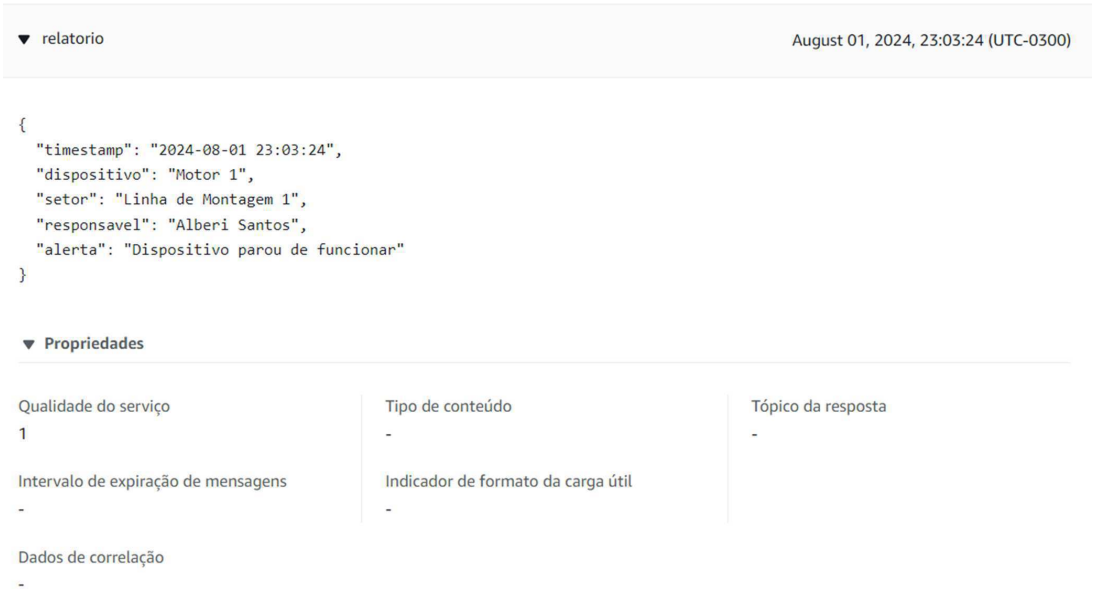
3.1.4 Integração com a Nuvem AWS

Para garantir que os dados coletados no ambiente industrial simulado fossem armazenados de forma segura e estivessem disponíveis para análise posterior, foi realizada uma integração com a plataforma de computação em nuvem AWS. A AWS foi escolhida por ser aplamente utilizada, sua robustez, escalabilidade e por oferecer uma ampla gama de serviços que suportam as necessidades de monitoramento e análise de dados em tempo real. A seguir, são descritos os principais passos realizados na AWS para viabilizar essa integração.

3.1.4.1 Configuração do AWS IoT

O primeiro passo foi configurar o serviço AWS IoT, que é um serviço gerenciado que facilita a conexão segura de dispositivos à nuvem e a troca de dados entre eles. No Node-RED, foi configurado um cliente MQTT para publicar dados em um tópico específico no AWS IoT. No AWS IoT, foi necessário criar uma *"Thing"*, que representa um dispositivo virtual no AWS IoT, e configurar políticas de segurança baseadas em certificados para garantir que apenas dispositivos autenticados pudessem publicar dados no tópico designado. O MQTT, sendo um protocolo leve e eficiente para comunicação M2M (*Machine-to-Machine*), foi ideal para a transmissão de dados em ambientes industriais.

Figura 11 – Mensagem MQTT publicada no tópico **relatorio** através do cliente MQTT configurado no Node-RED. A mensagem contém informações sobre um evento crítico ocorrido no ambiente industrial simulado.

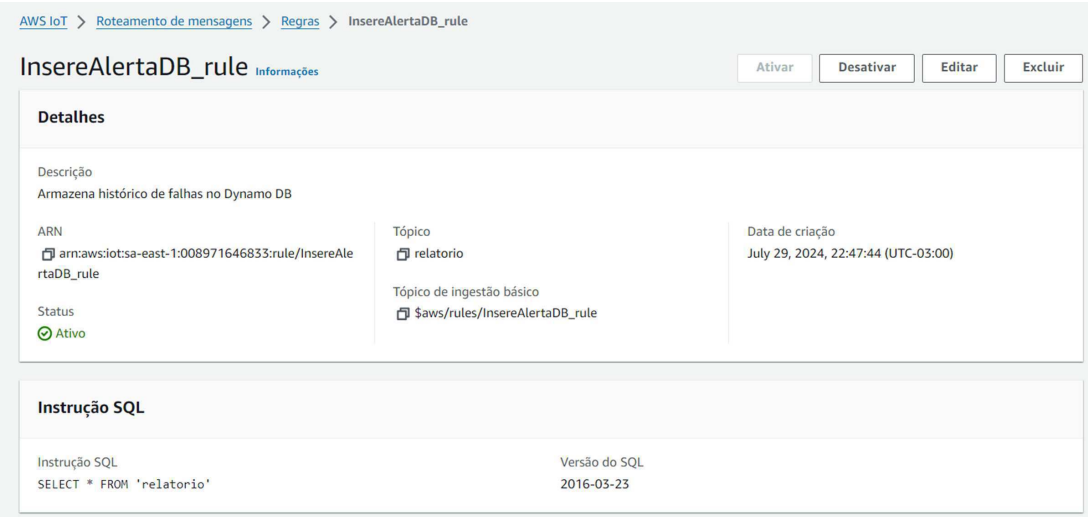


Fonte: Elaborado pelo autor.

3.1.4.2 Configuração de Regras de Roteamento

Com o tópico MQTT configurado, o próximo passo foi criar regras de roteamento no AWS IoT para direcionar as mensagens recebidas para o serviço de banco de dados DynamoDB. A criação dessas regras foi feita através do serviço AWS IoT Rules, que permite filtrar e transformar os dados do MQTT e encaminhá-los para outros serviços da AWS. A regra **InsereAlertaDB_rule** foi criada para selecionar todas as mensagens publicadas no tópico 'relatorio' e armazená-las em uma tabela DynamoDB. A AWS IoT Rules oferece flexibilidade para aplicar lógica SQL nas mensagens, permitindo que apenas dados relevantes sejam armazenados ou processados. Essa configuração é essencial para garantir que os dados sejam armazenados de forma estruturada e estejam disponíveis para consultas futuras.

Figura 12 – Configuração da regra **InsereAlertaDB_rule** no AWS IoT, que roteia mensagens do tópico 'relatorio' para uma tabela DynamoDB.

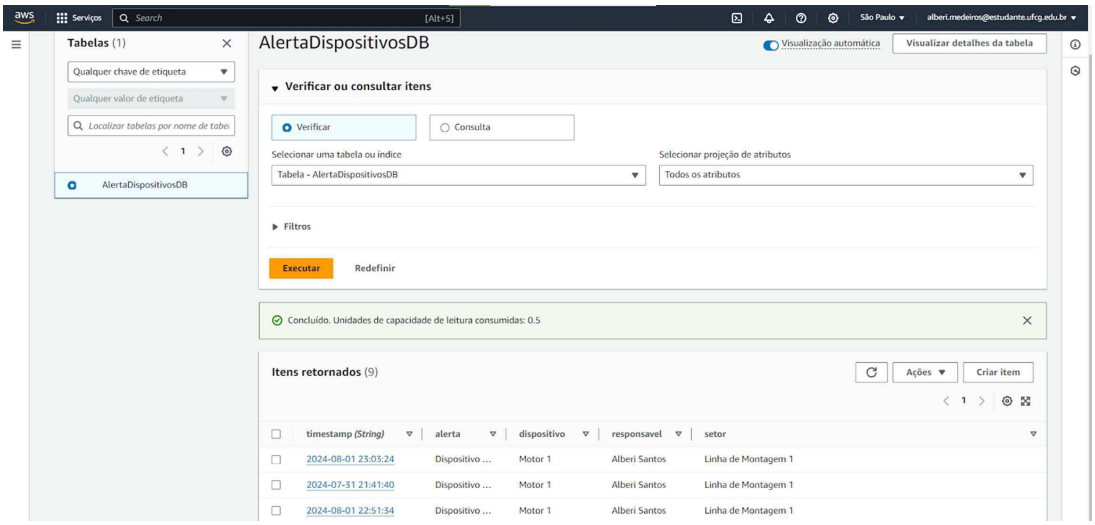


Fonte: Elaborado pelo autor.

3.1.4.3 Criação da Tabela DynamoDB

Para armazenar os dados coletados, foi criada uma tabela no DynamoDB chamada **AlertaDispositivosDB**. O Amazon DynamoDB é um serviço de banco de dados NoSQL totalmente gerenciado, que oferece baixa latência e alta escalabilidade, tornando-o ideal para aplicações que exigem alta disponibilidade e rapidez na consulta de dados. A tabela foi configurada para receber e organizar as mensagens enviadas pelo AWS IoT, permitindo uma rápida consulta e análise dos dados. Cada entrada na tabela contém informações como o timestamp do evento, o dispositivo afetado, o setor onde o dispositivo está localizado, o responsável pelo dispositivo, e a mensagem de alerta. Essa estrutura de armazenamento facilita a realização de análises detalhadas e o monitoramento de padrões de falhas ou anomalias no ambiente simulado.

Figura 13 – Consulta à tabela **AlertaDispositivosDB** no DynamoDB, mostrando logs de eventos críticos armazenados.



Fonte: Elaborado pelo autor.

3.1.4.4 Análise e Visualização dos Dados

Após o armazenamento dos dados no DynamoDB, foi possível realizar consultas diretamente na tabela para analisar os eventos críticos registrados. A AWS oferece ferramentas como o Amazon DynamoDB Console, que facilita a consulta e visualização dos dados armazenados. Essas ferramentas permitem que os usuários filtrem os dados por diferentes atributos, como timestamp ou tipo de alerta, identificando rapidamente padrões e anomalias que podem exigir intervenção. Além disso, os dados armazenados no DynamoDB podem ser integrados com outros serviços de análise da AWS, como o Amazon QuickSight ou o AWS Lambda, para criar painéis personalizados ou acionar funções automáticas em resposta a determinados eventos.

3.2 Análise e discussão

A análise a seguir examina o comportamento das comunicações entre diferentes componentes do sistema, focando em aspectos como latência, throughput e distribuição de pacotes. Duas capturas de tráfego de rede foram realizadas: uma entre o servidor OPC UA e o Node-RED, e outra entre o servidor AWS IoT e o Node-RED. Essas capturas permitiram avaliar como o tráfego de rede foi impactado por eventos específicos, como o desligamento do servidor e o pré-processamento dinâmico de mensagens no Node-RED. A partir dos resultados, foi possível identificar padrões de desempenho e compreender melhor os efeitos das configurações de rede e filtragem de mensagens no sistema.

3.2.1 Análise da Captura de Tráfego entre Servidor OPC UA e Node-RED

Para analisar a comunicação entre o servidor OPC UA e o Node-RED, foi realizada uma captura de tráfego utilizando o software Wireshark. A captura ocorreu na interface de loopback do sistema, onde foi filtrado o tráfego TCP na porta 4840, que é tipicamente utilizada pelo protocolo OPC UA. A Tabela 2 resume as características gerais da captura.

Tabela 2 – Características da Captura de Tráfego

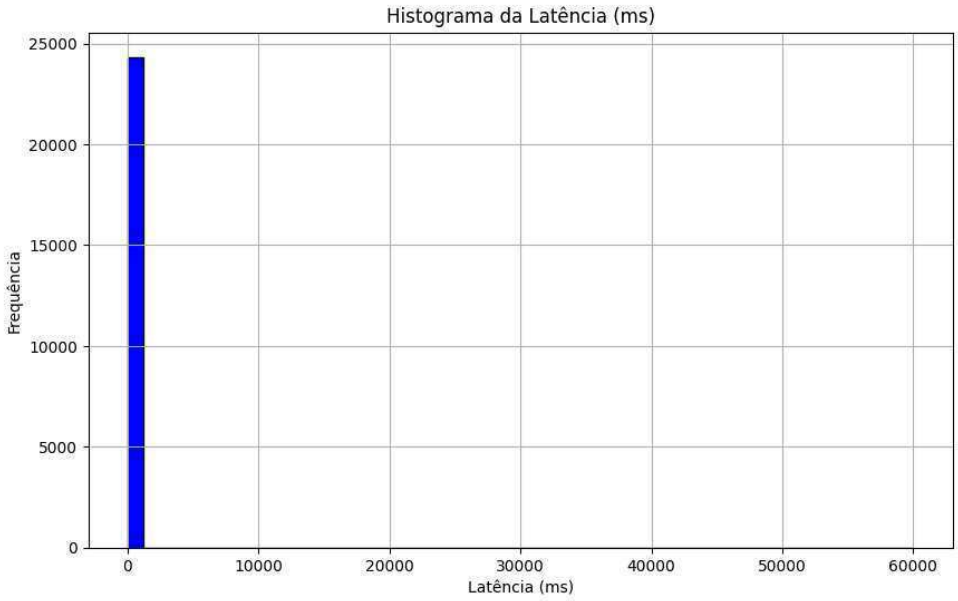
Parâmetro	**Valor**
Nome do Arquivo	tcp.port4840_ensaio0.pcapng
Tamanho do Arquivo	8353 kB
Primeiro Pacote	2024-09-02 20:40:00
Último Pacote	2024-09-02 21:10:11
Duração Total	00:30:10
Total de Pacotes Capturados	51,997
Tamanho Médio de Pacotes	129 bytes
Taxa Média de Bits	29 kbps

Durante a captura, foi simulado um estresse na rede com o desligamento do servidor OPC UA entre os períodos de 21:00 e 21:05, como mostrado nas Figuras 14 e 17.

3.2.2 Latência e Throughput

A latência média geral observada durante o período de captura foi de 0.529 segundos, com um throughput médio de 4.37 KB/s. A Figura 14 apresenta o histograma da latência registrada durante a captura, destacando uma concentração significativa de pacotes com baixa latência.

Figura 14 – Histograma da Latência (ms) durante a captura de tráfego entre servidor OPC UA e Node-RED.



Fonte: Elaborado pelo autor.

Para uma análise temporal mais detalhada da latência, foi gerado um gráfico de dispersão (Figura 16), que mostra a variação da latência ao longo do tempo. Este gráfico é útil para identificar variações e outliers durante o período de captura.

Figura 15 – Gráfico de Dispersão da Latência ao Longo do Tempo.

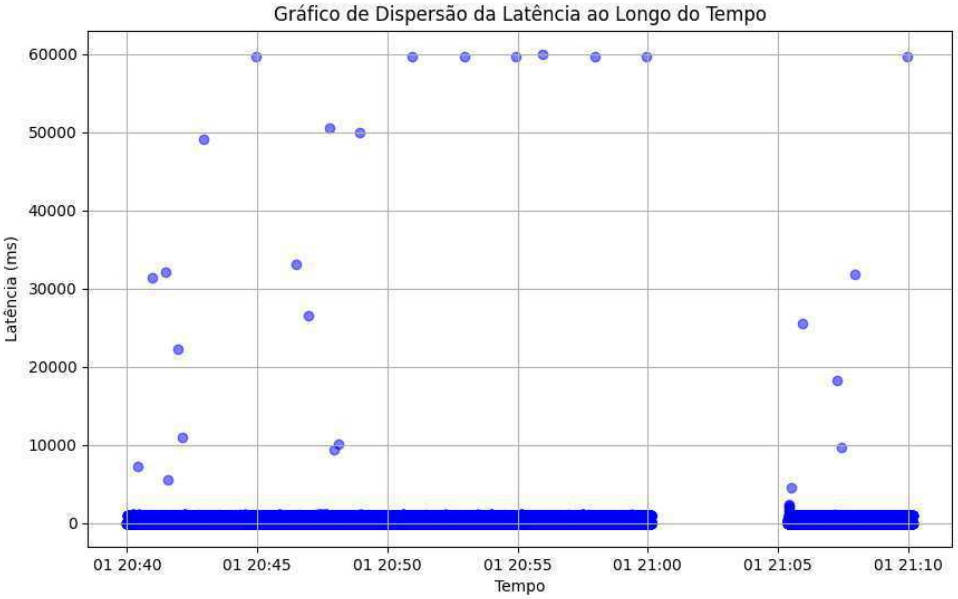
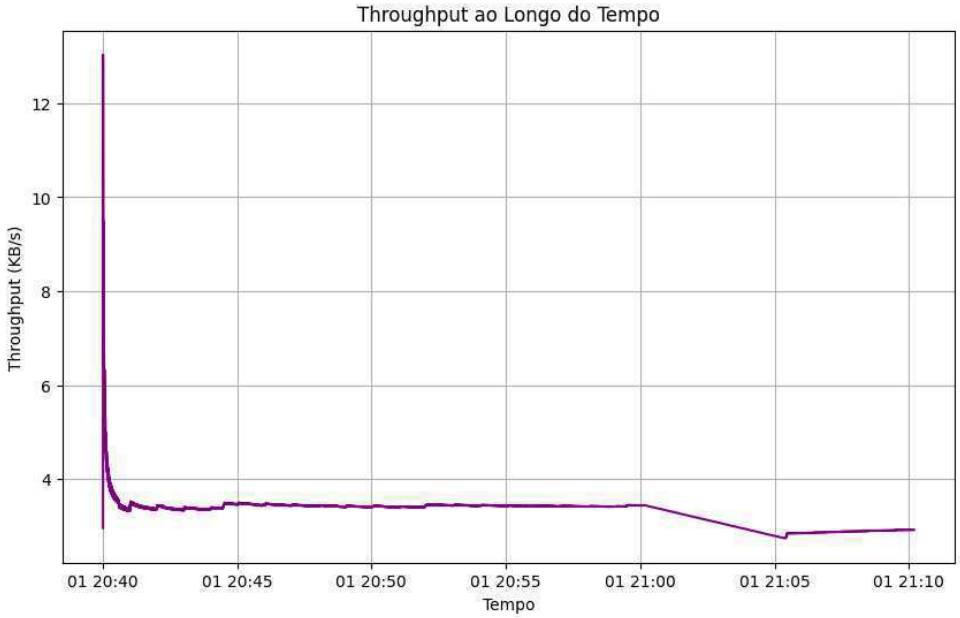


Figura 16 – Fonte: Elaborado pelo autor

A Figura 17 ilustra o throughput ao longo do tempo, evidenciando uma queda abrupta no momento do desligamento do servidor e uma recuperação gradual após o

restabelecimento da comunicação.

Figura 17 – Throughput ao longo do tempo (KB/s) durante a captura de tráfego.



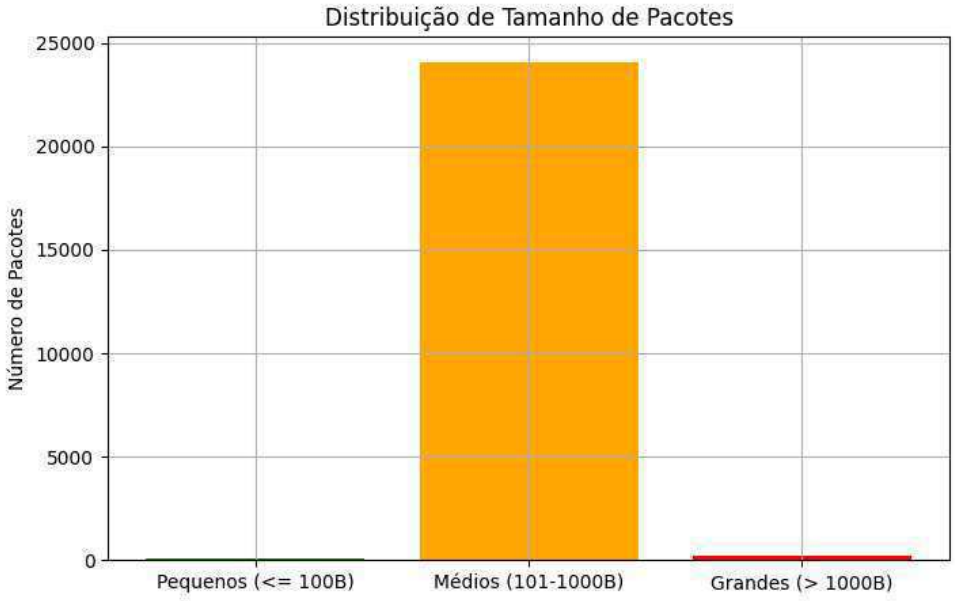
Fonte: Elaborado pelo autor.

O throughput começa muito alto no início da captura e rapidamente diminui para um valor estável e menor. Essa queda rápida no throughput pode indicar um burst inicial de dados seguido por uma transmissão mais estável e controlada. Isso é típico em cenários onde a conexão inicial envolve uma transferência de dados mais pesada (como configuração ou handshake inicial), seguida por um fluxo contínuo de dados menor.

3.2.3 Distribuição do Tamanho de Pacotes

A distribuição dos tamanhos dos pacotes capturados é ilustrada na Figura 39, mostrando que a maioria dos pacotes (24,087) tinha tamanho médio (entre 101 e 1000 bytes), com poucos pacotes pequenos (70 pacotes com menos de 100 bytes) e pacotes grandes (210 pacotes com mais de 1000 bytes).

Figura 18 – Distribuição dos Tamanhos de Pacotes durante a captura de tráfego.



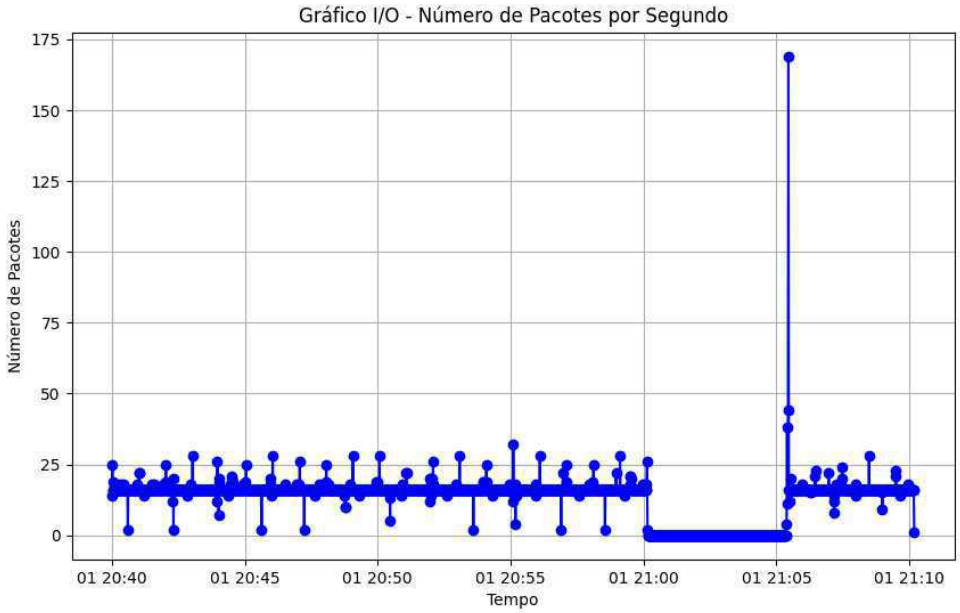
Fonte: Elaborado pelo autor.

O tráfego de rede é predominantemente composto por pacotes de tamanho médio, o que é comum em muitos protocolos de comunicação, como OPC UA. Isso indica uma quantidade considerável de dados sendo transmitida em mensagens de tamanho moderado, possivelmente contendo leituras de dados ou comandos.

3.2.4 Análise de Tráfego de Dados

Os gráficos I/O (*Input/Output* - do inglês, Entrada/Saída) fornecem uma visão detalhada da atividade da rede ao longo do tempo, permitindo uma análise aprofundada do comportamento do tráfego durante a captura. A Figura 40 mostra o número de pacotes transmitidos por segundo, enquanto a Figura 41 exibe a taxa de bytes por segundo.

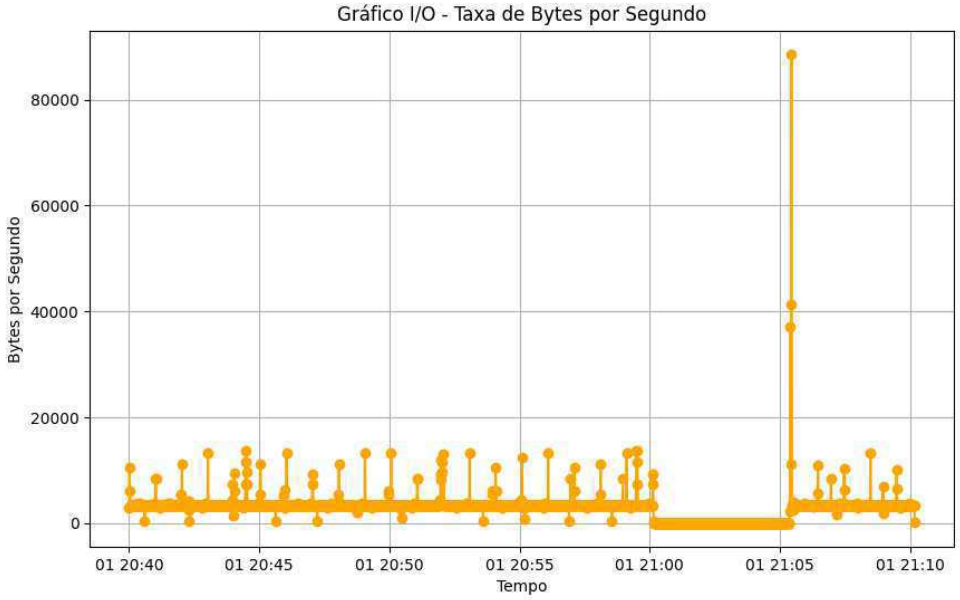
Figura 19 – Gráfico I/O - Número de Pacotes por Segundo durante a captura de tráfego.



Fonte: Elaborado pelo autor.

A análise dos gráficos I/O revela uma operação estável da rede até aproximadamente 21:00, momento em que o servidor OPC UA foi intencionalmente desligado. Durante o desligamento, há uma queda significativa no número de pacotes transmitidos e na taxa de bytes, conforme observado nas Figuras 40 e 41. Logo após o reinício do servidor, um pico nos gráficos indica uma recuperação da comunicação, sugerindo um acúmulo de pacotes que foram processados de uma vez.

Figura 20 – Gráfico I/O - Taxa de Bytes por Segundo durante a captura de tráfego.



Fonte: Elaborado pelo autor.

3.2.5 Análise da Captura de Tráfego entre Servidor AWS IoT e Node-RED

A segunda captura foi realizada para analisar o tráfego entre o servidor AWS IoT e o Node-RED, utilizando a porta 8883, que é comumente usada para comunicações MQTT sobre TLS. A captura foi feita utilizando o Wireshark, com as características gerais resumidas na Tabela 3.

Tabela 3 – Características da Captura de Tráfego (AWS IoT - Node-RED)

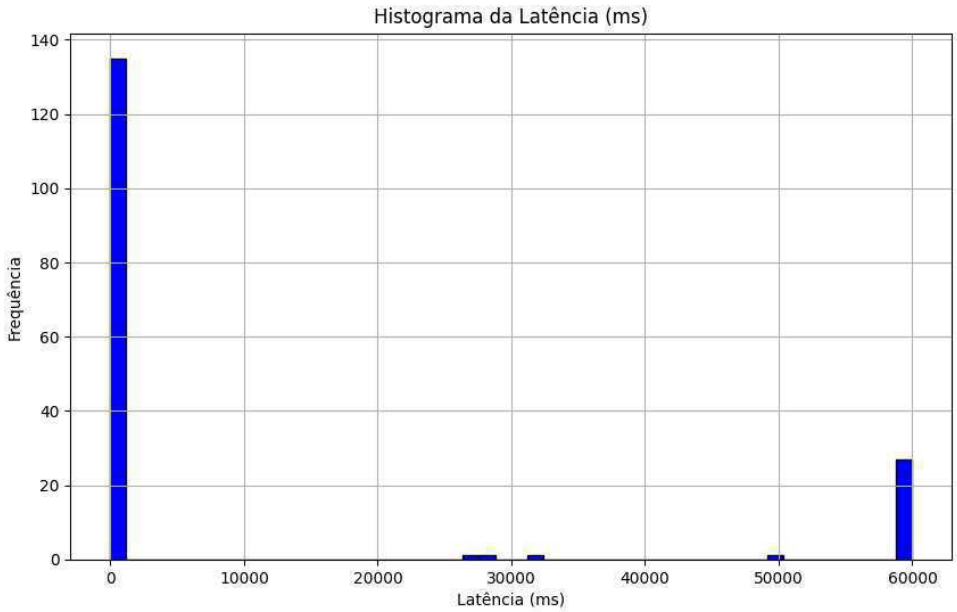
Parâmetro	**Valor**
Nome do Arquivo	tcp.port8883_ensaio0.pcapng
Tamanho do Arquivo	8460 kB
Primeiro Pacote	2024-09-02 20:40:09
Último Pacote	2024-09-02 21:10:12
Duração Total	00:30:03
Total de Pacotes Capturados	16,213
Tamanho Médio de Pacotes	488 bytes
Taxa Média de Bits	35 kbps

Durante essa captura, o tráfego entre o servidor AWS IoT e o Node-RED foi monitorado, e os resultados mostram um padrão de comunicação consistente, com alguns períodos de maior latência e throughput flutuante, como mostrado nas Figuras 21 e 23. É importante destacar que a quantidade de mensagens enviadas para o AWS foi reduzida devido a uma configuração dinâmica realizada no Node-RED. Conforme descrito em capítulos anteriores, essa configuração consiste em uma função que realiza o pré-processamento das mensagens recebidas do servidor OPC UA, filtrando apenas as mensagens de alerta que precisam ser armazenadas no AWS. Esse filtro seletivo impactou diretamente o número de pacotes capturados.

3.2.6 Latência e Throughput

A latência média geral observada durante a captura foi significativamente maior em comparação à captura 1, com um valor de 13.304 segundos, e o throughput médio foi de 0.01 KB/s. A Figura 21 apresenta o histograma da latência, onde podemos observar que, embora a maioria dos pacotes tenha baixa latência, um número considerável apresenta valores elevados, possivelmente devido à sobrecarga de rede ou retransmissões.

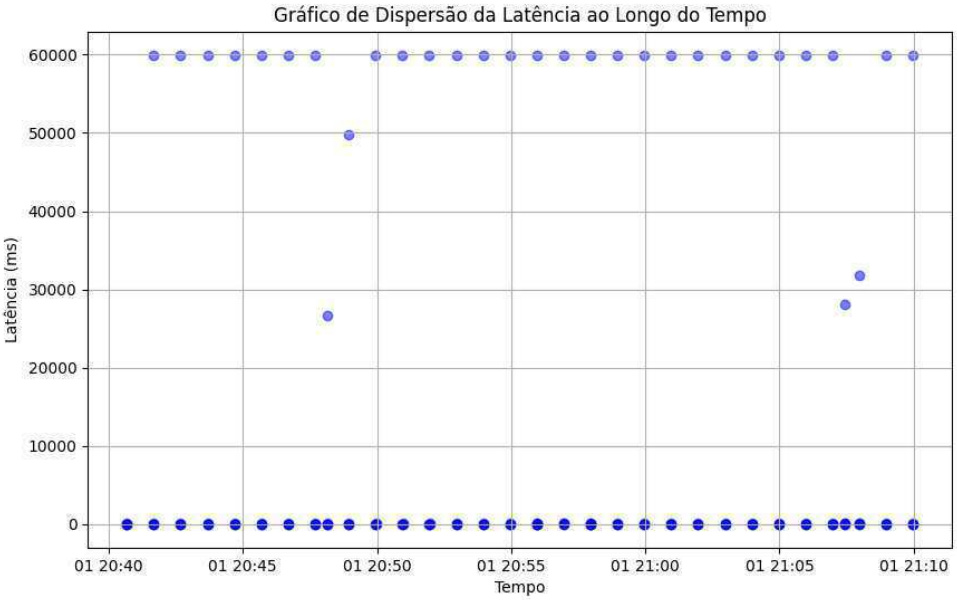
Figura 21 – Histograma da Latência (ms) durante a captura de tráfego entre servidor AWS IoT e Node-RED.



Fonte: Elaborado pelo autor.

A Figura 22 mostra o gráfico de dispersão da latência ao longo do tempo, evidenciando outliers de latência em determinados intervalos, com pacotes que ultrapassam 60.000 ms de latência, o que pode ser explicado por retransmissões ou atrasos significativos.

Figura 22 – Gráfico de Dispersão da Latência ao Longo do Tempo.

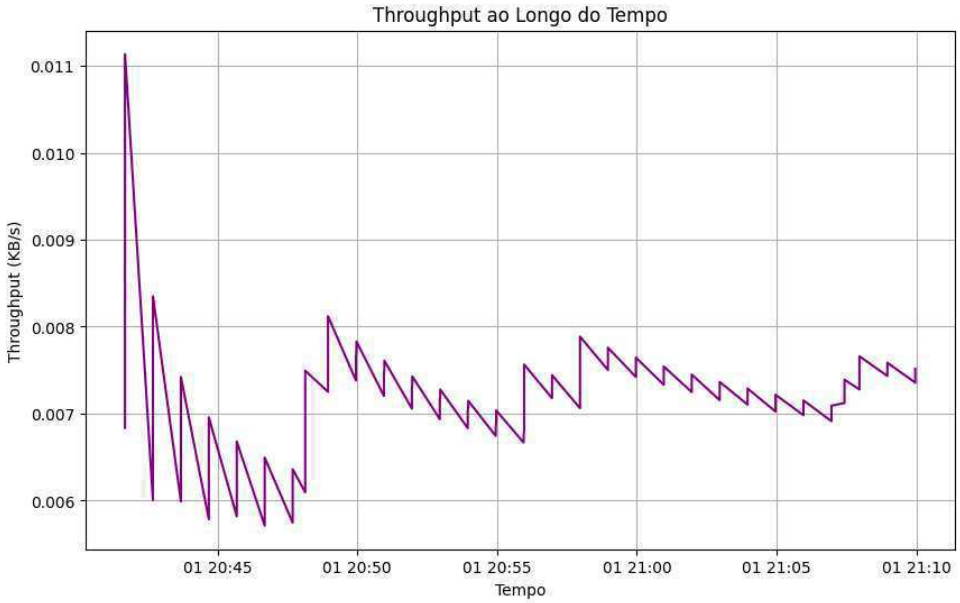


Fonte: Elaborado pelo autor.

Na Figura 23, é possível observar que o throughput se mantém relativamente baixo

e estável ao longo da captura, com algumas flutuações, especialmente no início, onde há um pico, possivelmente relacionado ao início das conexões MQTT.

Figura 23 – Throughput ao longo do tempo (KB/s) durante a captura de tráfego.

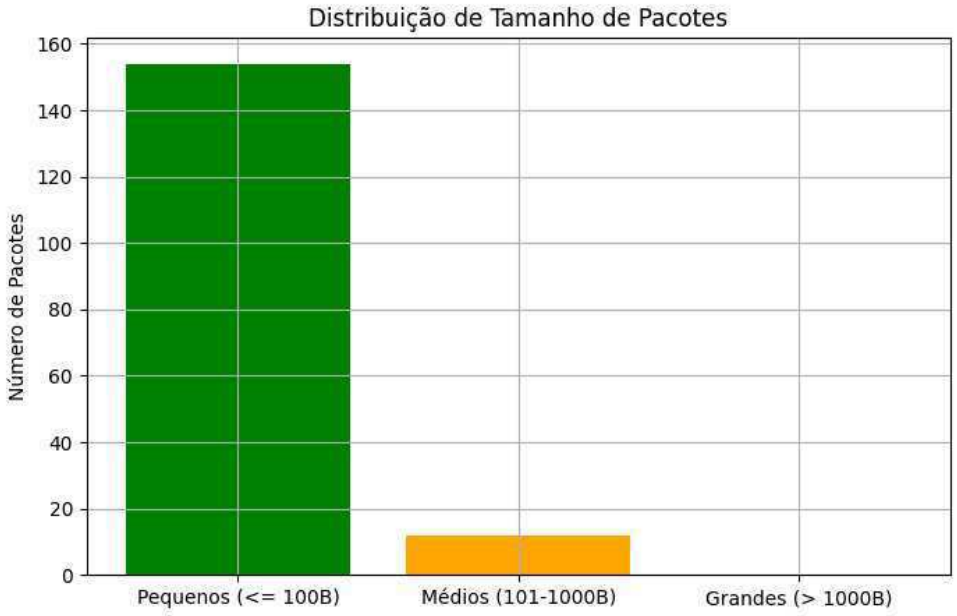


Fonte: Elaborado pelo autor.

3.2.7 Distribuição do Tamanho de Pacotes

A distribuição do tamanho dos pacotes é mostrada na Figura 24. A maioria dos pacotes capturados era de tamanho pequeno (<= 100 bytes), o que é esperado em comunicações baseadas em MQTT, onde o overhead de cada mensagem é pequeno. No entanto, também foi registrado um pequeno número de pacotes médios (entre 101 e 1000 bytes), com nenhum pacote classificado como grande (> 1000 bytes).

Figura 24 – Distribuição dos Tamanhos de Pacotes durante a captura de tráfego.

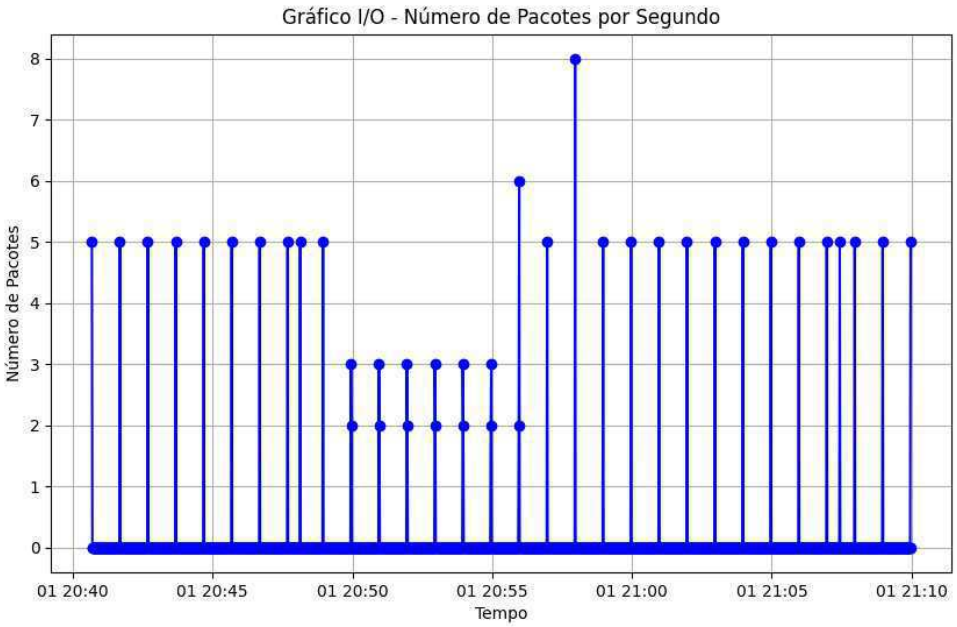


Fonte: Elaborado pelo autor.

3.2.8 Análise de Tráfego de Dados

Os gráficos I/O fornecem uma visão detalhada do comportamento do tráfego ao longo do tempo. A Figura 25 apresenta o número de pacotes transmitidos por segundo, enquanto a Figura 26 mostra a taxa de bytes por segundo. Ambos os gráficos mostram um padrão relativamente estável, com algumas variações pontuais.

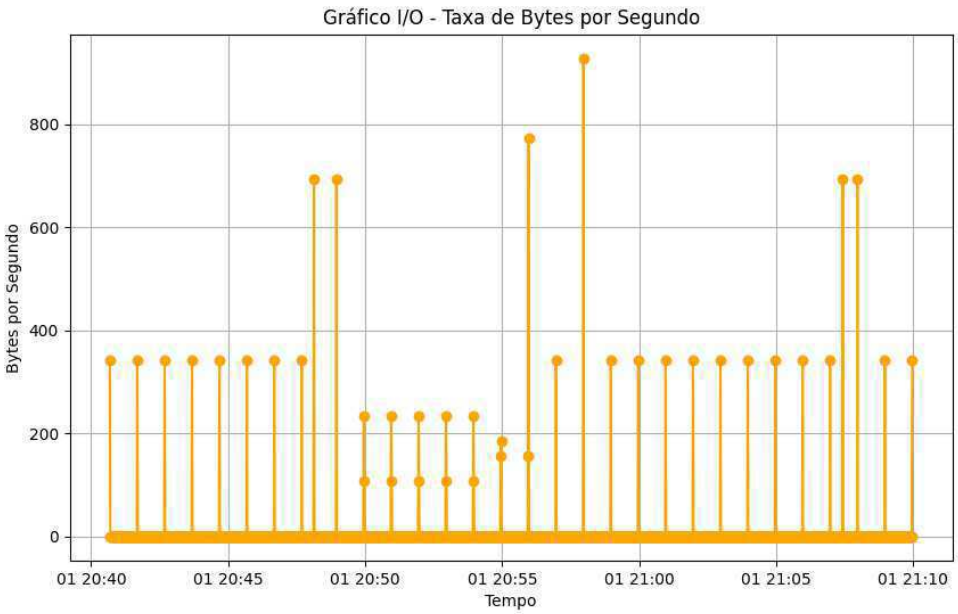
Figura 25 – Gráfico I/O - Número de Pacotes por Segundo durante a captura de tráfego.



Fonte: Elaborado pelo autor.

A Figura 26 mostra a taxa de bytes por segundo ao longo do tempo. Embora o número de pacotes por segundo tenha permanecido estável, a taxa de bytes transmitidos variou mais significativamente, com alguns picos, especialmente entre 20:50 e 21:00, que podem estar relacionados à variação de dados transmitidos em pacotes MQTT.

Figura 26 – Gráfico I/O - Taxa de Bytes por Segundo durante a captura de tráfego.



Fonte: Elaborado pelo autor.

4 Implementação de gêmeos digitais, Modbus e integração com a nuvem

Este capítulo aborda a implementação de um caso de uso para comunicação, interconexão e mapeamento dinâmico de dispositivos em um cenário de Indústria 4.0. A solução proposta combina diferentes tecnologias, incluindo o protocolo Modbus, o Node-RED e o *Azure Digital Twins*, para integrar e monitorar dados provenientes de um Controlador Lógico Programável (CLP).

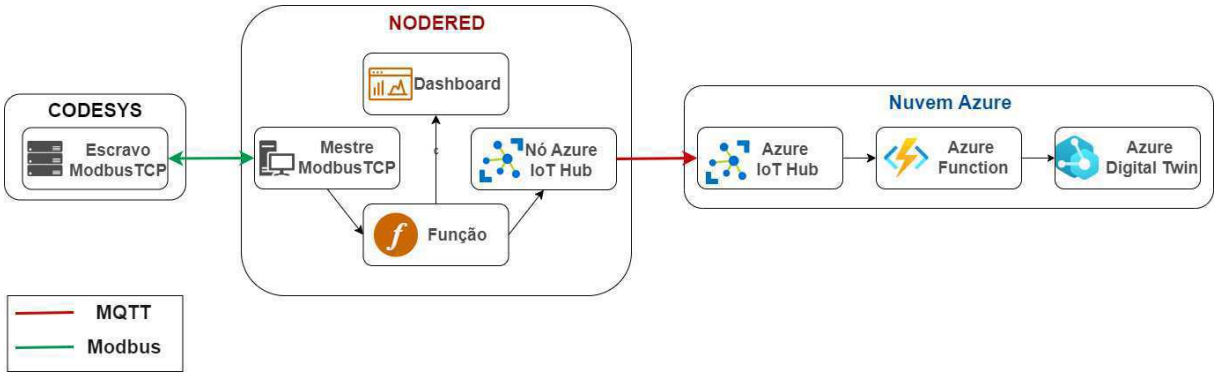
No cenário proposto, o CLP é programado no ambiente CODESYS para controlar processos industriais. Ele é configurado como um dispositivo escravo Modbus TCP, fornecendo dados em tempo real das entradas e saídas. A comunicação ocorre via Ethernet, utilizando a porta padrão do Modbus TCP (porta 502), permitindo a interconexão com sistemas externos. O Node-RED desempenha o papel de cliente Modbus (mestre), adquirindo os dados do CLP por meio do protocolo Modbus e realizando a leitura dos registros configurados.

O Node-RED atua como o sistema central da arquitetura, processando os dados coletados em tempo real. Por meio de funções personalizadas, como a calculadora de OEE, os dados são analisados e exibidos em um painel de controle (*dashboard*) para monitoramento contínuo. Essa integração permite uma análise detalhada do status operacional do CLP e dos processos industriais, facilitando o acompanhamento das operações.

Além do monitoramento local, o Node-RED envia os dados processados para a nuvem, integrando-se ao Azure IoT Hub por meio do protocolo MQTT. O nó do Azure IoT Hub no Node-RED é responsável por transmitir essas informações para a plataforma de nuvem da Microsoft. A partir daí, os dados são direcionados para uma função personalizada no Azure Functions, que decodifica a mensagem e atualiza os gêmeos digitais modelados no Azure Digital Twins. Essa atualização em tempo real reflete o status e o desempenho da linha de produção no ambiente digital, fornecendo uma visão abrangente e atualizada do chão de fábrica.

Dessa forma, a solução cria uma interconexão dinâmica entre os elementos físicos e digitais da fábrica. Essa interconexão é característica essencial em ambientes de Indústria 4.0, pois possibilita a coleta, processamento e análise de informações em tempo real, permitindo otimizar processos e apoiar a tomada de decisões. A Figura 27 ilustra o fluxo completo de comunicação e os componentes envolvidos no cenário proposto.

Figura 27 – Arquitetura do cenário de interconexão e comunicação Modbus com Node-RED e integração ao Azure.



Fonte: Elaborado pelo atutor.

4.1 Materiais e Métodos

4.1.1 Configuração do CLP no CODESYS

O ambiente CODESYS foi configurado para simular um CLP com o programa de controle 'PLC_PRG', que inclui temporizadores (TON) responsáveis por gerar sinais oscilantes com tempos predeterminados. A lógica de controle é implementada em linguagem ladder e programada para atuar em um ambiente industrial simulado.

A configuração do CLP no CODESYS foi realizada da seguinte forma:

- Adicionou-se um dispositivo *ModbusTCP Slave* à árvore de dispositivos no CODESYS, permitindo que o CLP se comporte como um escravo Modbus.
- No programa de controle, variáveis foram definidas e associadas a registradores Modbus. O mapeamento é feito através da interface de configuração do dispositivo Modbus TCP no CODESYS.
- Foram configurados registradores de entrada (*Input Registers*) e de retenção (*Holding Registers*) para representar os estados das variáveis no CLP. Essas variáveis podem ser acessadas pelo mestre Modbus via comandos específicos.

4.1.2 Configuração do Node-RED como Cliente Modbus

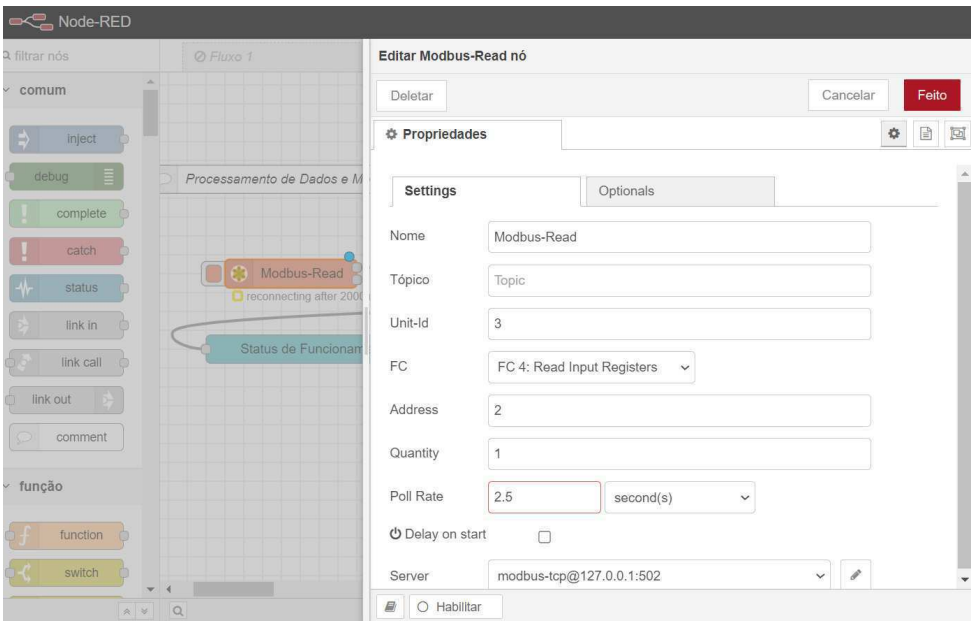
No lado do mestre, o Node-RED foi configurado para se conectar ao dispositivo escravo Modbus programado no CODESYS. A configuração do nó Modbus no Node-RED envolve diversos parâmetros essenciais para a comunicação adequada:

- Definição do endereço IP do CLP (neste caso, `localhost`) e da porta de comunicação (502) para estabelecer a conexão Modbus TCP.

- Configuração dos nós Modbus para leitura dos registradores de entrada e retenção do CLP, especificando o *Unit-ID* e os endereços dos registradores conforme mapeados no CODESYS.
- Ajuste da taxa de amostragem (*poll rate*) para adquirir os dados em intervalos regulares, garantindo a captura em tempo real das variáveis de processo.

A Figura 28 ilustra a configuração do nó `Modbus-Read` no Node-RED, onde se define os parâmetros mencionados, como o tipo de função Modbus a ser utilizada, o endereço do registrador e o intervalo de aquisição dos dados.

Figura 28 – Configuração do nó Modbus-Read no Node-RED.



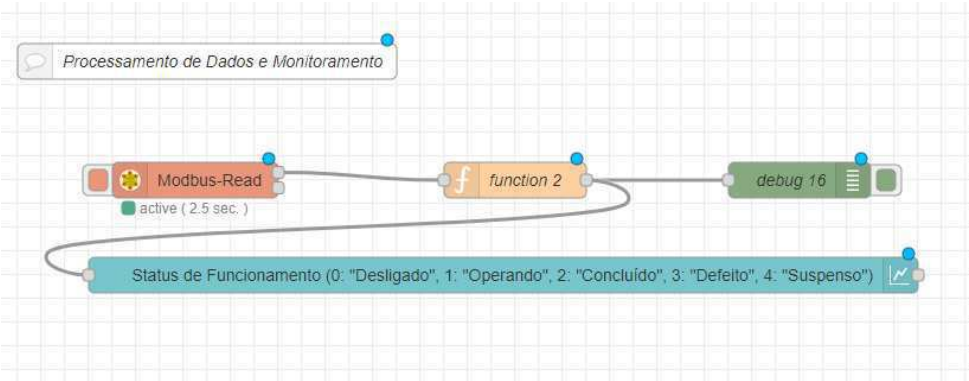
Fonte: Elaborado pelo atutor.

4.1.3 Processamento de Dados e Monitoramento com Node-RED

Para coletar, processar e monitorar os dados provenientes do CLP, foi implementada uma função personalizada no Node-RED. Essa função interpreta os dados recebidos do CLP e mapeia os códigos de status em descrições compreensíveis, facilitando o monitoramento e a análise dos processos industriais.

A Figura 29 apresenta o fluxo implementado no Node-RED. O nó `Modbus-Read` realiza a leitura dos dados do CLP em intervalos configurados (neste caso, a cada 2,5 segundos). Esses dados são então processados pela `function 2`, que mapeia os códigos de status recebidos para descrições como "Desligado", "Operando", "Concluído", "Defeito"e "Suspenso". Por fim, as informações processadas são exibidas no painel de monitoramento e também enviadas para um nó de depuração para análise adicional.

Figura 29 – Fluxo de processamento de dados e monitoramento no Node-RED.



Elaborada pelo autor.

4.1.3.1 Função de Processamento no Node-RED

A função implementada no Node-RED verifica se a carga útil (*payload*) recebida contém um status válido e ajusta a variável `codigoStatusModbus`. A lógica da função é mostrada abaixo:

```
// Função Node-RED
// Verifica se o payload contém um status válido e ajusta ‘codigoStatusModbus‘
let codigoStatusModbus;

// Função para obter o status da montagem no Node-RED
function obterStatusMontagem(codigoStatusModbus) {
    // Mapeamento de status baseado nos códigos de status
    let mapeamentoStatus = {
        0: "Estação pronta para trabalhar",
        1: "Em progresso",
        2: "Trabalho concluído e peça boa fabricada",
        3: "Trabalho concluído e sucata fabricada",
        4: "Estação em estado de falha"
    };

    // Retorna a descrição do status com base no código de status
    return mapeamentoStatus[codigoStatusModbus] || "Status Desconhecido";
}

// Chama a função para obter a descrição do status
let descricaoStatus = obterStatusMontagem(codigoStatusModbus);
```

```
// Define a saída para o Node-RED
msg.payload = codigoStatusModbus;
return msg;
```

Esta função realiza o mapeamento dos códigos de status provenientes do CLP, representando-os em uma forma mais compreensível para monitoramento. A variável `msg.payload` é então atualizada com o código de status e enviada para um nó *dashboard* no Node-RED.

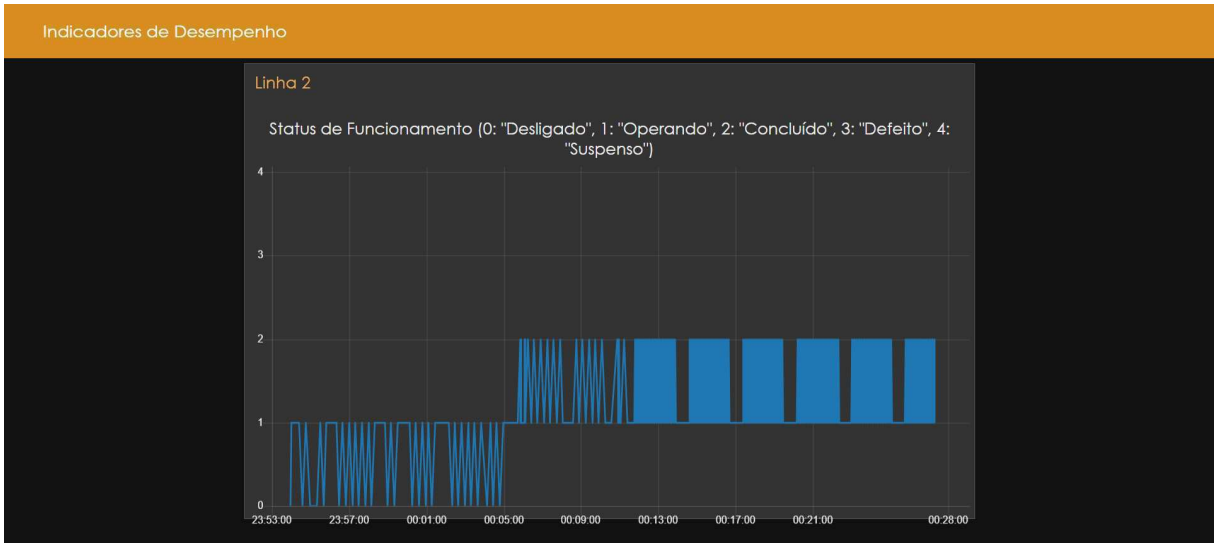
4.1.3.2 Visualização dos Dados

Os dados processados pela função são enviados para um gráfico de linha no *dashboard* do Node-RED. A configuração do gráfico inclui:

- O eixo X representa o tempo, possibilitando a visualização do histórico de status do CLP ao longo de um intervalo de uma hora.
- O eixo Y representa os diferentes estados do CLP, com uma legenda que define os possíveis valores: "Desligado"(0), "Operando"(1), "Concluído"(2), "Defeito"(3) e "Suspenso"(4).
- A atualização dos dados no gráfico ocorre em tempo real, conforme o Node-RED recebe e processa novos dados do CLP.

A imagem abaixo (Figura 30) ilustra o gráfico gerado no dashboard, permitindo uma análise visual imediata do status de funcionamento do CLP ao longo do tempo.

Figura 30 – Dashboard do Node-RED exibindo o status de funcionamento do CLP.



Fonte: Elaborado pelo atutor

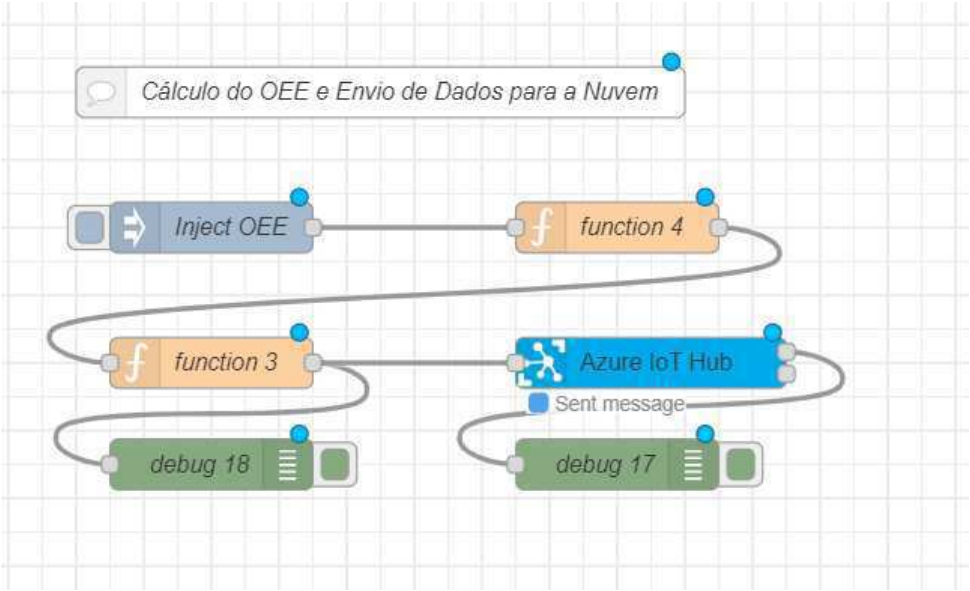
4.1.4 Cálculo do OEE e Envio de Dados para a Nuvem

No cenário de interconexão em um ambiente de Indústria 4.0, o monitoramento da eficiência das operações de produção é fundamental. Para isso, foi implementada uma função no Node-RED que calcula o *Overall Equipment Effectiveness* (OEE), um indicador-chave que reflete a disponibilidade, o desempenho e a qualidade do processo produtivo.

Após o cálculo, os dados do OEE são preparados e enviados para a nuvem por meio do serviço Azure IoT Hub. Essa integração permite que as informações de desempenho sejam analisadas e monitoradas remotamente, proporcionando insights em tempo real sobre as operações da fábrica.

A Figura 31 apresenta o fluxo implementado no Node-RED. A partir de um nó `Inject OEE`, a função é acionada para calcular o OEE utilizando os dados coletados do processo produtivo. Em seguida, a função `function 3` realiza o cálculo e envia os dados processados para o nó `Azure IoT Hub`, que os transmite para a nuvem.

Figura 31 – Fluxo de cálculo do OEE e envio de dados para a nuvem no Node-RED.



Fonte: Elaborado pelo atutor.

4.1.4.1 Função para Calcular o OEE

A função desenvolvida no Node-RED realiza o cálculo do OEE com base em cinco parâmetros: tempo de produção planejado, tempo de operação, ciclo ideal, total de produtos produzidos e produtos bons. A lógica da função é mostrada abaixo:

```
// Função para calcular OEE no Node-RED
function calcularOEE(tempoProducaoPlanejado, tempoOperacao, cicloIdeal, totalProdut
```

```
    // Calcular Disponibilidade
    let disponibilidade = (tempoProducaoPlanejado > 0) ? (tempoOperacao / tempoProd

    // Calcular Desempenho
    let desempenho = (tempoOperacao > 0) ? ((cicloIdeal * totalProdutosProduzidos)

    // Calcular Qualidade
    let qualidade = (totalProdutosProduzidos > 0) ? (produtosBons / totalProdutosPr

    // Calcular OEE
    let oee = disponibilidade * desempenho * qualidade;

    return {
        OEE: Math.round(oee * 100 * 100) / 100,  // Retorna como porcentagem com du
        Disponibilidade: Math.round(disponibilidade * 100 * 100) / 100,
        Desempenho: Math.round(desempenho * 100 * 100) / 100,
        Qualidade: Math.round(qualidade * 100 * 100) / 100
    };
}
```

Após o cálculo do OEE, a função retorna um objeto contendo o OEE total, disponibilidade, desempenho e qualidade, todos em porcentagem com duas casas decimais.

4.1.4.2 Envio dos Dados ao Azure IoT Hub

Os dados calculados são, então, preparados para envio ao Azure IoT Hub. O Node-RED utiliza o protocolo MQTT para conectar-se ao Azure e transmitir os dados. A mensagem gerada pela função é estruturada da seguinte forma:

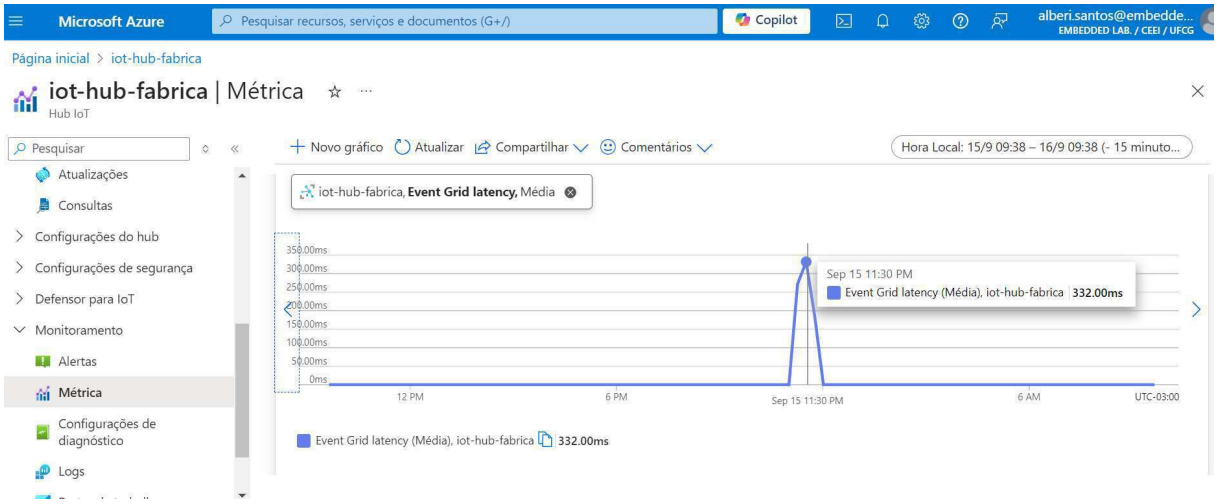
```
// Cria a mensagem para enviar ao Azure IoT Hub
msg.payload = {
    "deviceId": "FLM1_Esteira1_OEE",
    "key": "0HPWfNoe/g7FWGU1O7r/Rn4EICfS8YRgFAIoTK7Q+Ao=",
    "protocol": "mqtt",
    "data": {
        "StepId": "factory",
        "OEE": dadosOEE.OEE,
        "Disponibilidade": dadosOEE.Disponibilidade,
        "Desempenho": dadosOEE.Desempenho,
        "Qualidade": dadosOEE.Qualidade
```

```
    }
};
```

O nó *Azure IoT Hub* no Node-RED é configurado para enviar a mensagem para a nuvem, utilizando o endereço do serviço (`iot-hub-fabrica.azure-devices.net`) e o protocolo MQTT. A Figura **??** mostra o fluxo configurado no Node-RED para calcular e enviar o OEE.

O monitoramento do processo de transmissão e atualização de dados é realizado através do portal do Azure. A Figura 32 mostra uma métrica do Azure IoT Hub, exibindo a latência média do *Event Grid* para o cenário configurado. Essas métricas são essenciais para avaliar o desempenho da solução e garantir a eficiência na comunicação entre os dispositivos industriais e os serviços de nuvem.

Figura 32 – Métricas do Azure IoT Hub, mostrando a latência do *Event Grid*.



Fonte: Elaborado pelo atutor.

4.1.5 Integração com a Nuvem

Após o cálculo do OEE e o envio dos dados para o Azure IoT Hub, é necessário processar e utilizar essas informações para alimentar sistemas avançados, como gêmeos digitais. O *Azure IoT Hub* encaminha os dados para uma função personalizada no *Azure Functions*, que realiza o processamento e atualização do *Azure Digital Twins*.

4.1.5.1 Processamento com Azure Functions

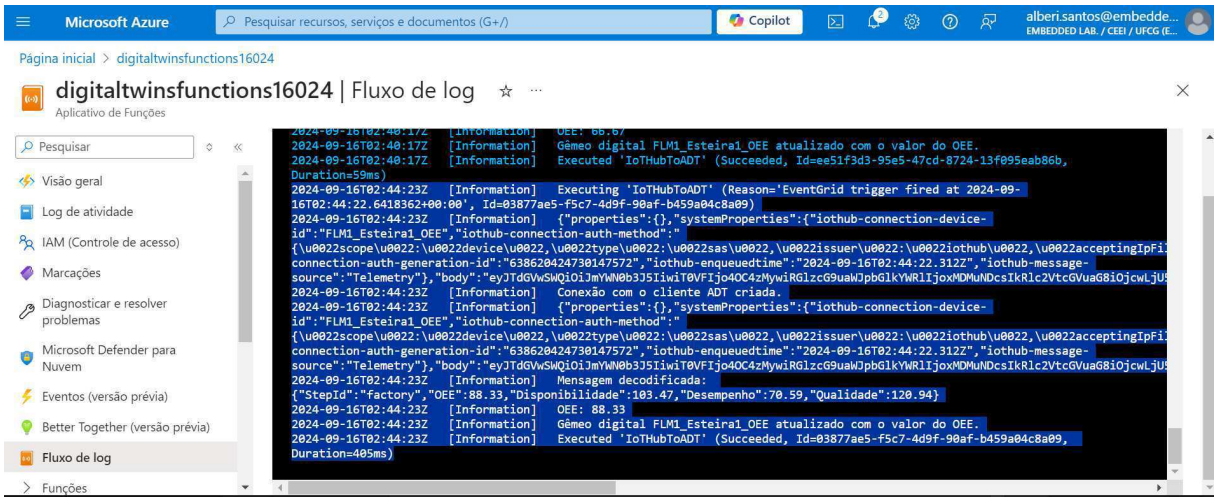
O *Azure IoT Hub* recebe os dados enviados pelo Node-RED e os encaminha para uma função personalizada (*Azure Functions*). A função recebe e decodifica a mensagem, realiza uma análise e obtém o valor do OEE da mensagem. Em seguida, utiliza este valor para atualizar a propriedade do gêmeo digital correspondente no *Azure Digital Twins*.

O código responsável por essa operação está detalhado no Apêndice A.1. A implementação da função segue os seguintes passos:

- Decodifica a mensagem em *base64* e a converte para um objeto JSON.
- Obtém o valor do OEE da mensagem.
- Cria um documento JSON *patch* para atualizar a propriedade do gêmeo digital no Azure Digital Twins.
- Atualiza o gêmeo digital `FLM1_Esteira1_OEE` com o valor do OEE recebido.

Os logs gerados pelo Azure Functions fornecem informações detalhadas sobre o processamento de mensagens, atualização de gêmeos digitais e possíveis erros ocorridos. A Figura 33 apresenta uma visão do fluxo de logs no portal do Azure, destacando a execução bem-sucedida da atualização do gêmeo digital com o valor do OEE.

Figura 33 – Fluxo de logs no Azure Functions durante o processamento das mensagens.



Fonte: Elaborado pelo autor.

4.1.5.2 Atualização do Gêmeo Digital no Azure Digital Twins

Após o processamento da mensagem, a função personalizada atualiza o gêmeo digital no *Azure Digital Twins*. O gêmeo digital `FLM1_Esteira1_OEE` é representado na plataforma e possui propriedades que refletem o estado atual da linha de produção. A Figura 34 ilustra a interface do *Azure Digital Twins* com o gêmeo digital e suas propriedades atualizadas.

Figura 34 – Gêmeo digital no Azure Digital Twins com a propriedade OEE atualizada.

Fonte: Elaborado pelo atutor.

4.1.6 Implementação de Gêmeos Digitais

A implementação de gêmeos digitais no cenário de Indústria 4.0 foi realizada utilizando a plataforma *Azure Digital Twins*. Para modelar os gêmeos digitais, foi utilizado um conjunto de ontologias pré-existentes do setor de fabricação, baseadas na norma ISA-95. A ISA-95 é um padrão internacional para a integração entre sistemas de controle industrial e sistemas empresariais, fornecendo um modelo de referência para a interconexão e descrição dos diversos componentes de uma planta fabril.

4.1.6.1 Ontologias para Fabricação Baseadas na ISA-95

As ontologias utilizadas foram obtidas do repositório do Digital Twin Consortium[1], que fornece modelos para representar a estrutura de uma fábrica, suas áreas, equipamentos, processos e indicadores de desempenho. A Figura 35 mostra a estrutura do gêmeo digital implementado, que inclui os seguintes elementos:

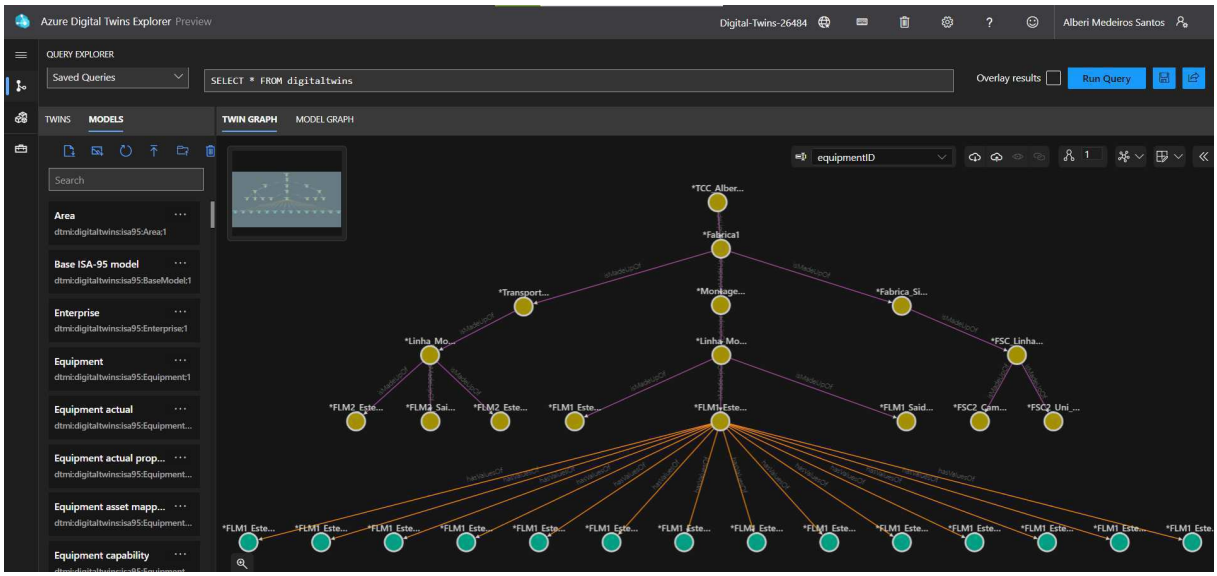
- ***Enterprise*:** Representa a fábrica como um todo, contendo as diferentes áreas de produção.
- ***Area*:** Descreve as áreas específicas dentro da fábrica, seguindo o modelo ISA-95 para definir seções como linha de montagem e transporte de materiais.
- ***Equipment*:** Modela os equipamentos dentro da fábrica, incluindo esteiras e máquinas, com suas propriedades específicas, como o OEE.
- ***Equipment Actual*:** Representa o estado atual dos equipamentos, incluindo métricas de desempenho, disponibilidade e qualidade.

[1] <https://github.com/digitaltwinconsortium/ManufacturingOntologies/tree/main>

- ***Process Capability*:** Define as capacidades e limites de operação de cada equipamento, descrevendo, por exemplo, os ciclos ideais de operação e os produtos que podem ser fabricados.

O uso destas ontologias padronizadas proporciona uma representação abrangente e escalável da planta fabril no ambiente de gêmeos digitais, permitindo integrar dados de diferentes fontes e sistemas de forma coerente.

Figura 35 – Estrutura de gêmeos digitais no Azure Digital Twins baseada na ontologia ISA-95.



Fonte: Elaborado pelo autor.

4.1.6.2 Modelagem dos Gêmeos Digitais no Azure Digital Twins

A implementação no Azure Digital Twins utiliza o conceito de *Digital Twins Definition Language* (DTDL) para definir os modelos dos gêmeos digitais. Cada tipo de gêmeo digital (como equipamentos e áreas) é descrito em um arquivo JSON que especifica suas propriedades, relacionamentos e telemetria. A plataforma permite a criação de gráficos interconectados que representam a planta fabril e suas operações em tempo real.

A seguir, um exemplo de um modelo DTDL utilizado para criar o gêmeo digital `FLM1_Esteira1_OEE`, que define as propriedades e relacionamentos do equipamento:

```
{
  "@context": "dtmi:dtdl:context;2",
  "@id": "dtmi:digitaltwins:isa95:EquipmentProperty;1",
  "@type": "Interface",
  "displayName": "Equipment property",
  "description": "Properties of equipment shall be presented as equipment
```

```
                properties. An equipment shall have zero or more
                equipment properties. These specify the current property
                values of the equipment for the associated equipment
                class property.",
  "comment": "According to ANSI/ISA-95.00.02-2018 Enterprise-Control
             System Integration - Part 2: Objects and Attributes for
             - Approved 24 May 2018",
  "extends": [
    "dtmi:digitaltwins:isa95:ResourceProperty;1",
    "dtmi:digitaltwins:isa95:TestableObjectProperty;1"
  ],
  "contents": [
    {
      "@type": "Relationship",
      "name": "contains",
      "displayName": "Contains",
      "description": "The nested equipment property(s) makes up part
                     of this equipment property as the whole.",
      "target": "dtmi:digitaltwins:isa95:EquipmentProperty;1",
      "comment": "Relationship type: Composition hierarchy"
    },
    {
      "@type": "Relationship",
      "name": "mapsTo",
      "displayName": "Maps to",
      "description": "If the parent equipment supports an equipment class,
                     the equipment class property(s) is applied in this
                     equipment property(s).",
      "target": "dtmi:digitaltwins:isa95:EquipmentClassProperty;1",
      "comment": "Relationship type: Dependency"
    }
  ]
}
```

No projeto, o gêmeo digital `FLM1_Esteira1_OEE` foi modelado para refletir o status e o desempenho da esteira na linha de montagem. Ele é atualizado pela função personalizada no Azure Functions, que recebe os dados do OEE via IoT Hub e aplica a atualização no gêmeo correspondente.

4.1.6.3 Como Funciona o Azure Digital Twins

O Azure Digital Twins permite modelar e monitorar o ambiente físico, digitalizando os processos e relacionamentos entre os componentes da fábrica. A partir dos dados coletados e das ontologias ISA-95 aplicadas, a plataforma cria um grafo de gêmeos digitais, que pode ser explorado para visualizar as operações da fábrica, identificar gargalos e otimizar processos.

Conforme detalhado na documentação da Microsoft Azure (2024), o Azure Digital Twins oferece as seguintes funcionalidades:

- **Modelagem**: Define modelos de gêmeos digitais usando DTDL, representando propriedades, eventos e relacionamentos.
- **Conectividade**: Integra-se com outras soluções, como IoT Hub e Azure Functions, para coletar e processar dados em tempo real.
- **Análise**: Permite a execução de consultas complexas para analisar o estado dos gêmeos e gerar insights sobre a operação da fábrica.

4.2 Análise e Discussão

A análise a seguir examina o comportamento da comunicação entre diferentes componentes do sistema em um cenário de Indústria 4.0, com foco em aspectos como latência, throughput e distribuição de pacotes. Para realizar essa análise, foi feita uma captura de tráfego de rede da comunicação Modbus entre o software CODESYS e o Node-RED, utilizando a ferramenta Wireshark. Essa captura de tráfego permitiu avaliar o impacto da transmissão de dados em tempo real e identificar padrões de desempenho, além de fornecer insights sobre o comportamento do sistema e as configurações de rede utilizadas.

A partir dos dados capturados, foram examinados parâmetros-chave, como latência média, throughput, distribuição do tamanho dos pacotes e a atividade de rede ao longo do tempo. Esses resultados são essenciais para entender o desempenho da comunicação em um ambiente industrial conectado e para verificar se as operações atendem aos requisitos de eficiência e confiabilidade do projeto.

4.2.1 Análise da Captura de Tráfego entre CODESYS e Node-RED

Para analisar a comunicação entre o software CODESYS e o Node-RED, foi realizada uma captura de tráfego utilizando o software Wireshark. A captura ocorreu na interface de loopback do sistema, onde foi filtrado o tráfego TCP na porta 502, que é

comumente utilizada pelo protocolo Modbus TCP. A Tabela 4 resume as características gerais da captura de tráfego realizada durante o experimento.

Tabela 4 – Características da Captura de Tráfego Modbus

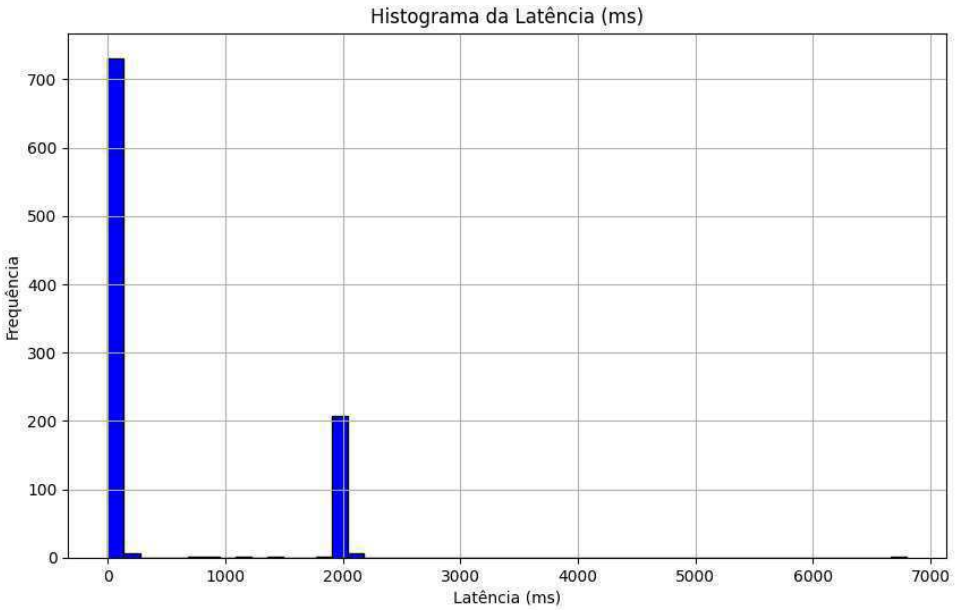
Parâmetro	**Valor**
Nome do Arquivo	tcp.port502_ensaio0.pcapng
Tamanho do Arquivo	78 kB
Primeiro Pacote	2024-09-17 16:45:40
Último Pacote	2024-09-17 16:55:33
Duração Total	00:09:52
Total de Pacotes Capturados	959
Tamanho Médio de Pacotes	50 bytes
Taxa Média de Bits	643 bps

A captura foi realizada em um cenário de operação contínua, coletando dados do CODESYS através do protocolo Modbus TCP. Essa análise permitiu identificar padrões de comunicação e avaliar a eficiência da troca de informações entre o Node-RED e o CODESYS. As Figuras 36 e 38 apresentam mais detalhes sobre a latência e o throughput ao longo do período de captura.

4.2.2 Latência e Throughput

A latência média geral observada durante o período de captura foi de 0.467 segundos, com um throughput médio de 0.13 KB/s. A Figura 36 apresenta o histograma da latência registrada durante a captura, destacando uma concentração significativa de pacotes com baixa latência.

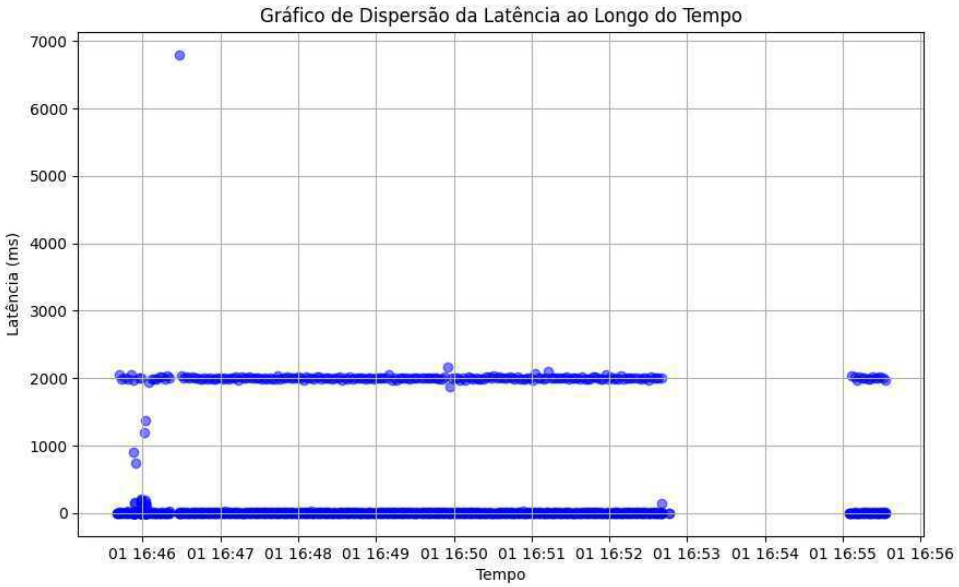
Figura 36 – Histograma da Latência (ms) durante a captura de tráfego entre CODESYS e Node-RED.



Fonte: Elaborado pelo autor.

Para uma análise temporal mais detalhada da latência, foi gerado um gráfico de dispersão (Figura 37), que mostra a variação da latência ao longo do tempo. Este gráfico é útil para identificar variações e outliers durante o período de captura.

Figura 37 – Gráfico de Dispersão da Latência ao Longo do Tempo.

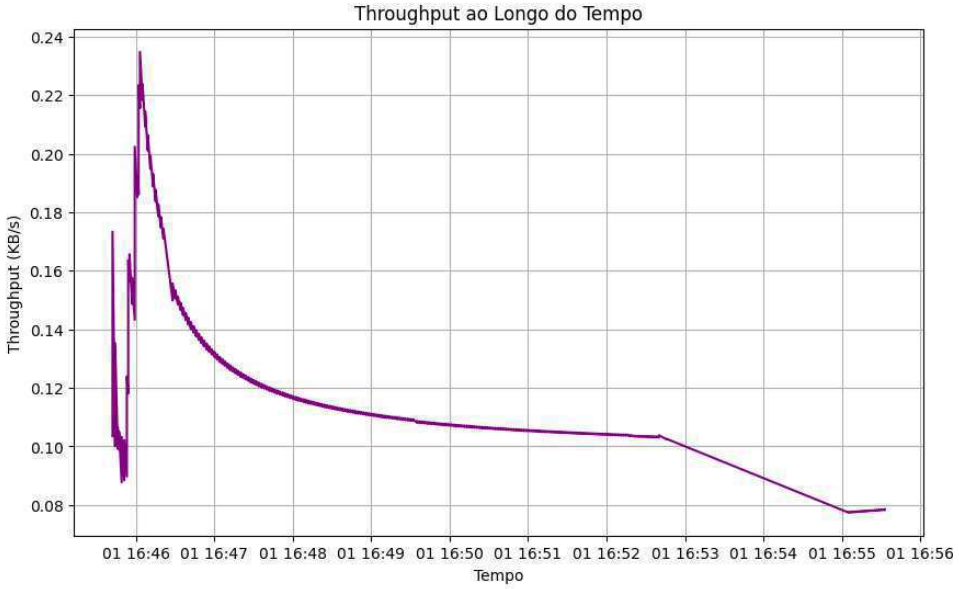


Fonte: Elaborado pelo autor.

A Figura 38 ilustra o throughput ao longo do tempo, evidenciando um padrão de queda gradual após um pico inicial, que é típico em comunicações onde ocorre um burst

de dados logo no início da conexão, seguido por uma transmissão mais estável e contínua.

Figura 38 – Throughput ao longo do tempo (KB/s) durante a captura de tráfego.



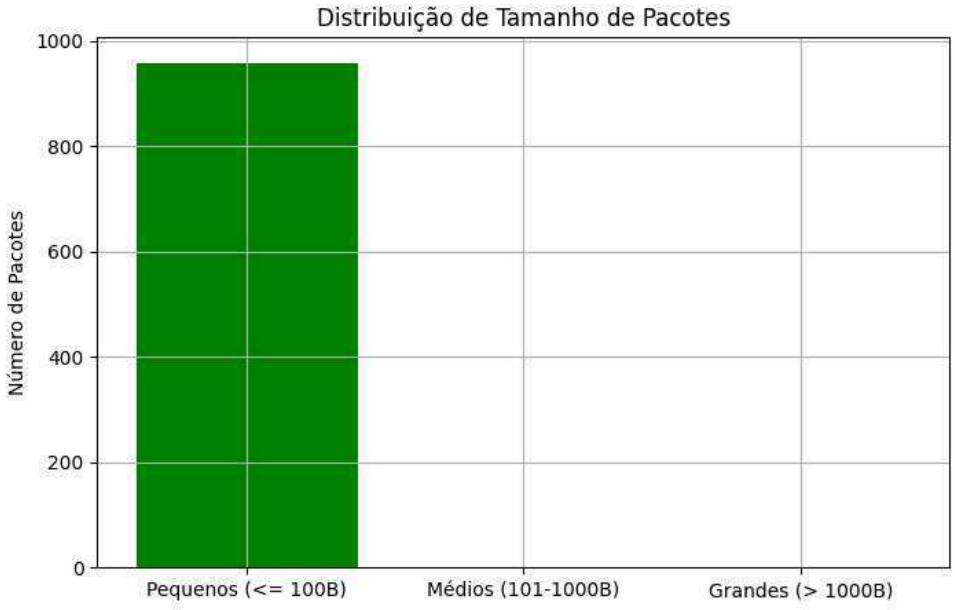
Fonte: Elaborado pelo autor.

O throughput começa alto no início da captura e rapidamente diminui para um valor estável e menor. Essa queda rápida no throughput pode indicar uma transferência inicial mais pesada, como a configuração ou um handshake, seguida por um fluxo contínuo de dados menor, típico em trocas de informações periódicas como as encontradas no protocolo Modbus.

4.2.3 Distribuição do Tamanho de Pacotes

A distribuição dos tamanhos dos pacotes capturados é ilustrada na Figura 39, mostrando que todos os pacotes capturados (959) tinham tamanho pequeno (menos de 100 bytes), não havendo pacotes médios (entre 101 e 1000 bytes) ou grandes (mais de 1000 bytes).

Figura 39 – Distribuição dos Tamanhos de Pacotes durante a captura de tráfego.



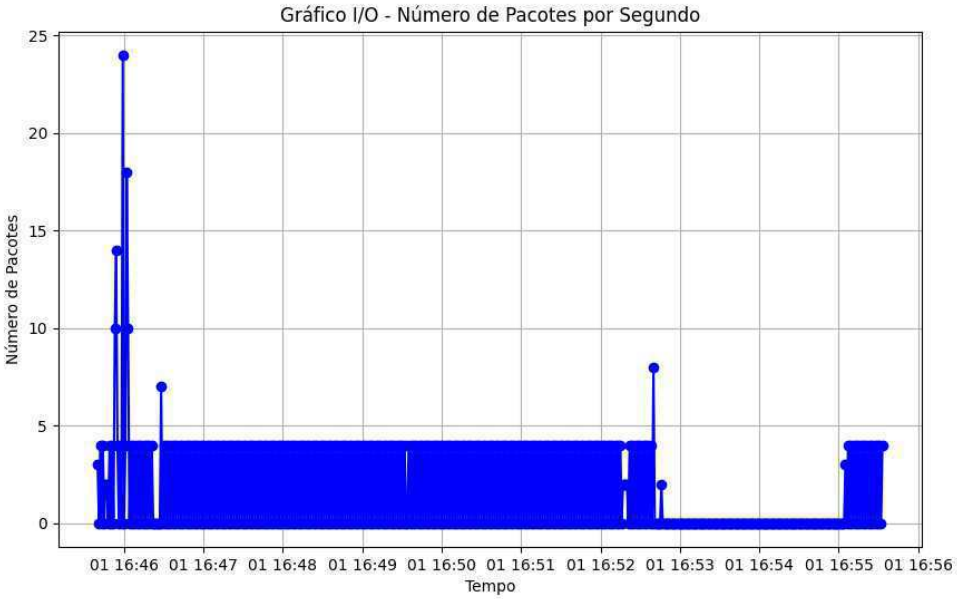
Fonte: Elaborado pelo autor.

O tráfego de rede é composto exclusivamente por pacotes de tamanho pequeno, o que é característico de comunicações envolvendo o protocolo Modbus. Essa característica sugere que os dados trocados são pequenos, consistindo provavelmente em leituras simples ou comandos básicos enviados ao CLP.

4.2.4 Análise de Tráfego de Dados

Os gráficos I/O (*Input/Output* - do inglês, Entrada/Saída) fornecem uma visão detalhada da atividade da rede ao longo do tempo, permitindo uma análise aprofundada do comportamento do tráfego durante a captura. A Figura 40 mostra o número de pacotes transmitidos por segundo, enquanto a Figura 41 exibe a taxa de bytes por segundo.

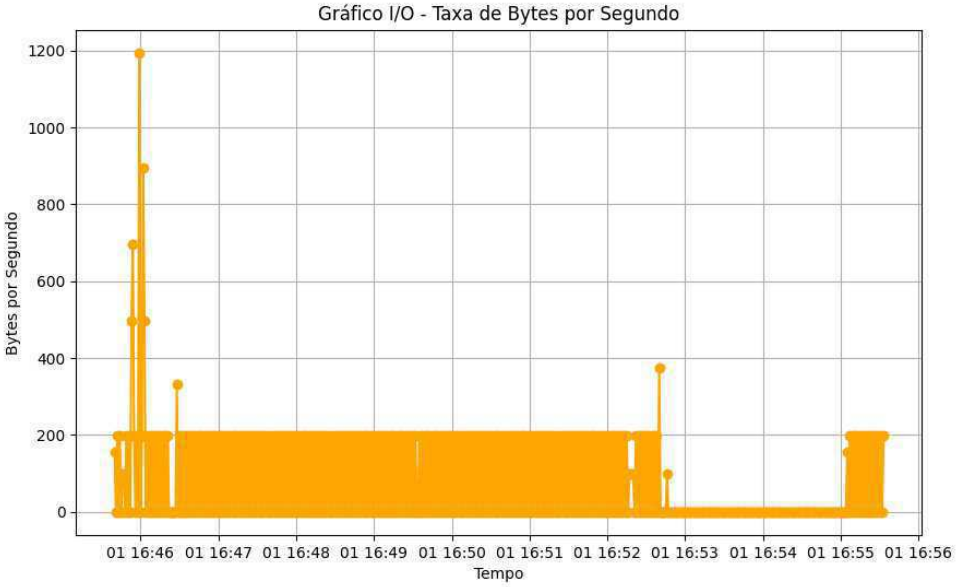
Figura 40 – Gráfico I/O - Número de Pacotes por Segundo durante a captura de tráfego.



Fonte: Elaborado pelo autor.

A análise dos gráficos I/O revela uma operação relativamente estável na rede, com picos esporádicos de atividade. Esses picos podem estar associados a eventos específicos de comunicação com o CLP, como leituras ou escritas de registros. A Figura 41 mostra a taxa de bytes por segundo, indicando uma quantidade modesta de dados sendo transmitida em intervalos regulares.

Figura 41 – Gráfico I/O - Taxa de Bytes por Segundo durante a captura de tráfego.



Fonte: Elaborado pelo autor.

4.2.5 Análise da Captura de Tráfego entre Node-RED e Azure

Para analisar a comunicação entre o Node-RED e a plataforma de nuvem Azure, foi realizada uma captura de tráfego utilizando o software Wireshark. A captura ocorreu na interface Wi-Fi do sistema, onde foi filtrado o tráfego TCP na porta 8883, que é tipicamente utilizada pelo protocolo MQTT seguro. A Tabela 5 resume as características gerais da captura.

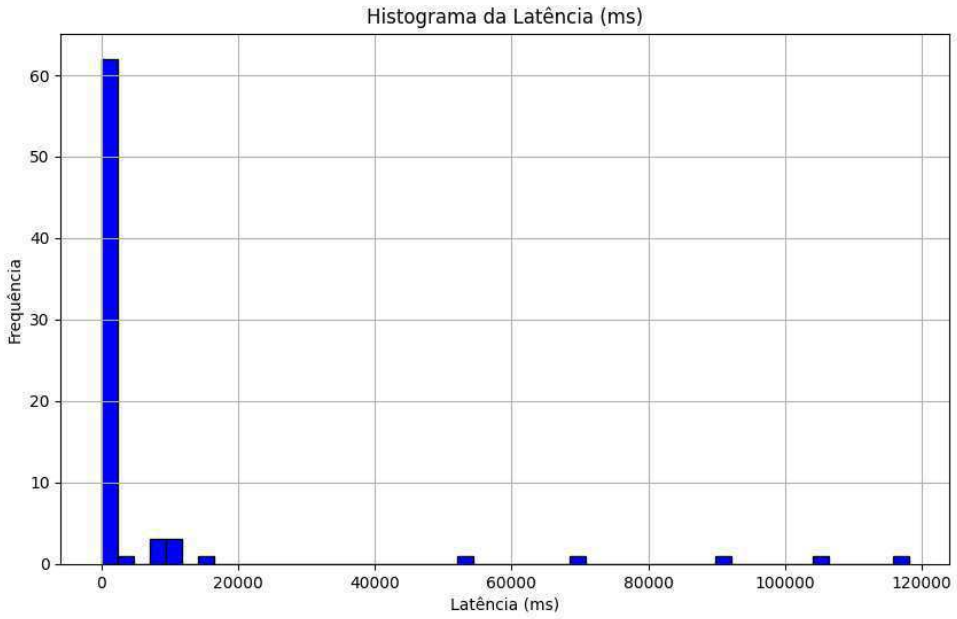
Tabela 5 – Características da Captura de Tráfego

Parâmetro	**Valor**
Nome do Arquivo	tcp.port8883_ensaio0.pcapng
Tamanho do Arquivo	14 kB
Primeiro Pacote	2024-09-17 16:46:24
Último Pacote	2024-09-17 16:55:05
Duração Total	00:08:41
Total de Pacotes Capturados	75
Tamanho Médio de Pacotes	152 bytes
Taxa Média de Bits	175 bps

4.2.6 Latência e Throughput

A latência média geral observada durante o período de captura foi de 6.955 segundos, com um throughput médio de 0.02 KB/s. A Figura 42 apresenta o histograma da latência registrada durante a captura, destacando uma variabilidade significativa nos tempos de resposta.

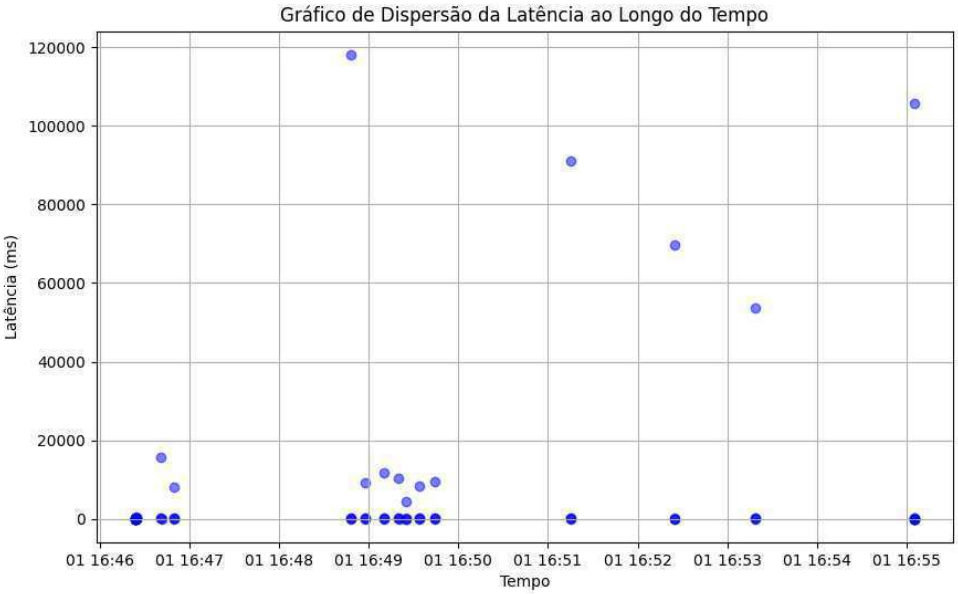
Figura 42 – Histograma da Latência (ms) durante a captura de tráfego entre Node-RED e Azure.



Fonte: Elaborado pelo autor.

A Figura 43 mostra o gráfico de dispersão da latência ao longo do tempo, evidenciando picos ocasionais que podem indicar variações na estabilidade da conexão.

Figura 43 – Gráfico de Dispersão da Latência ao Longo do Tempo.

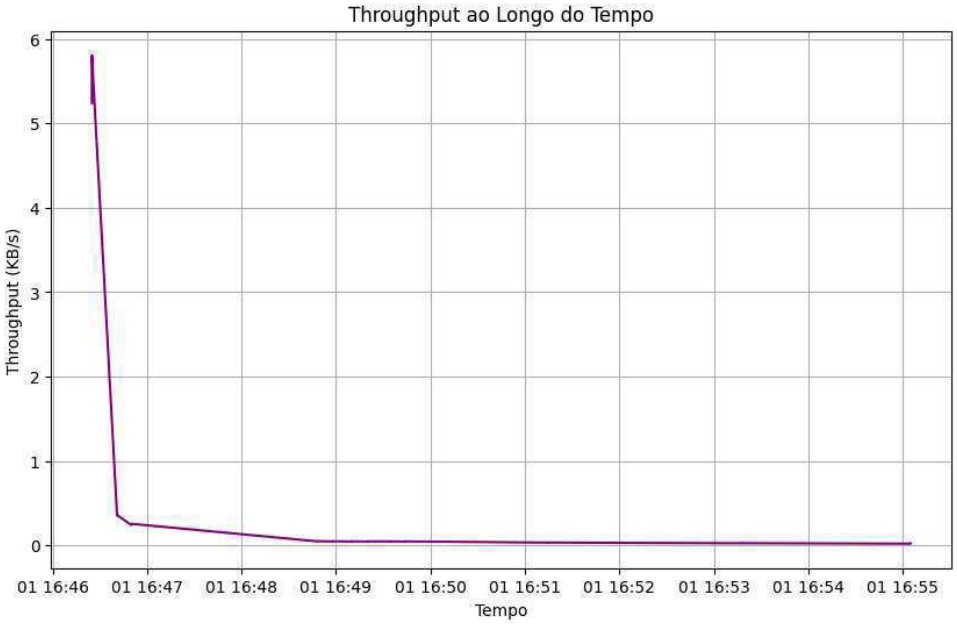


Fonte: Elaborado pelo autor.

A Figura 44 ilustra o throughput ao longo do tempo, que começa elevado no início da captura e rapidamente estabiliza em um valor baixo, típico em cenários onde há uma

transmissão inicial mais pesada seguida por um fluxo contínuo de dados menores.

Figura 44 – Throughput ao longo do tempo (KB/s) durante a captura de tráfego.

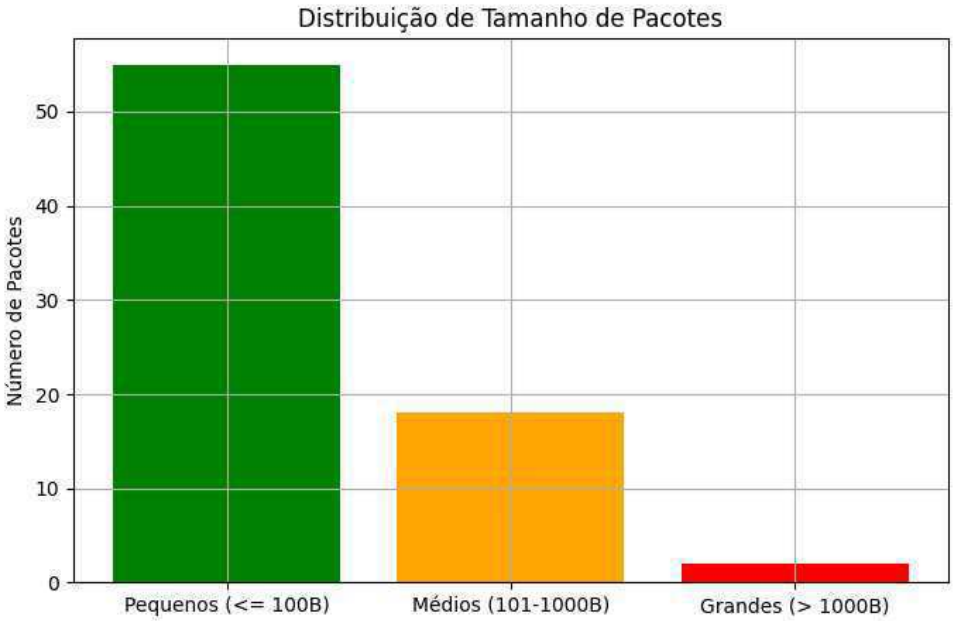


Fonte: Elaborado pelo autor.

4.2.7 Distribuição do Tamanho de Pacotes

A distribuição dos tamanhos dos pacotes capturados é ilustrada na Figura 45, mostrando que a maioria dos pacotes (55) eram pequenos (<= 100 bytes), com 18 pacotes de tamanho médio (101 - 1000 bytes) e apenas 2 pacotes grandes (> 1000 bytes).

Figura 45 – Distribuição dos Tamanhos de Pacotes durante a captura de tráfego.

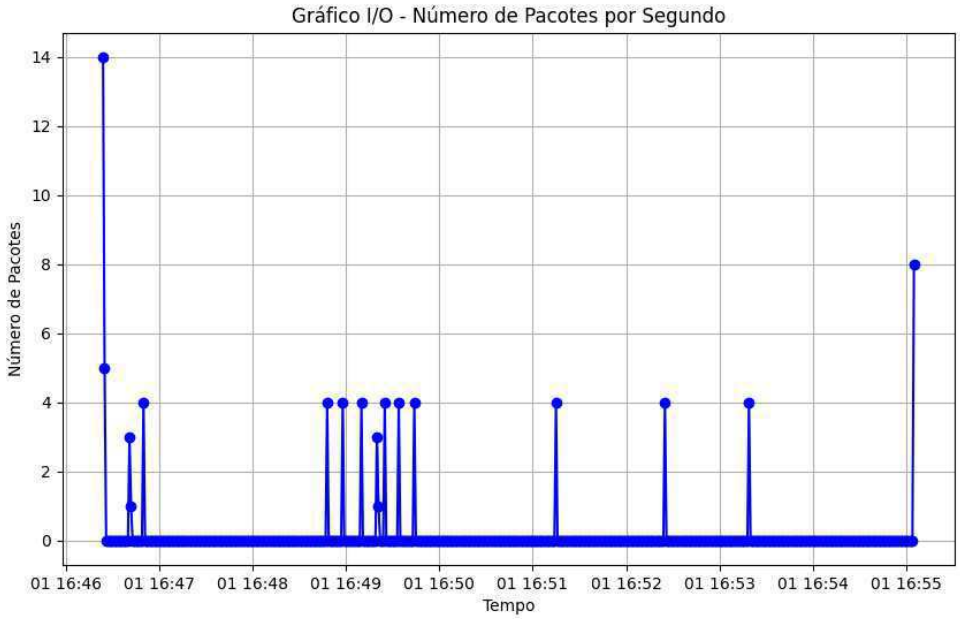


Fonte: Elaborado pelo autor.

4.2.8 Análise de Tráfego de Dados

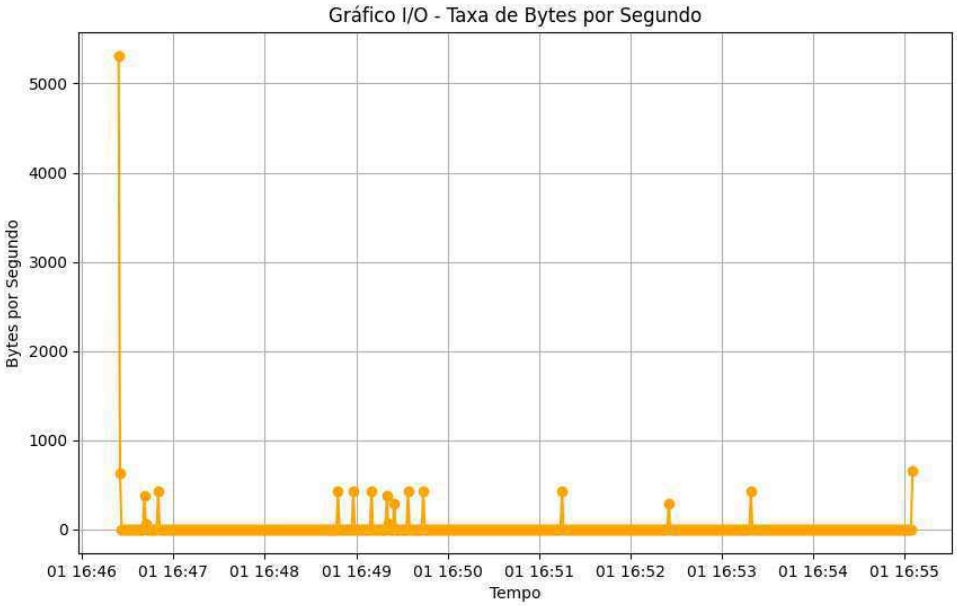
Os gráficos I/O fornecem uma visão detalhada da atividade da rede ao longo do tempo. A Figura 46 mostra o número de pacotes transmitidos por segundo, enquanto a Figura 47 exibe a taxa de bytes por segundo.

Figura 46 – Gráfico I/O - Número de Pacotes por Segundo durante a captura de tráfego.



Fonte: Elaborado pelo autor.

Figura 47 – Gráfico I/O - Taxa de Bytes por Segundo durante a captura de tráfego.



Fonte: Elaborado pelo autor.

5 Integração e validação da solução completa

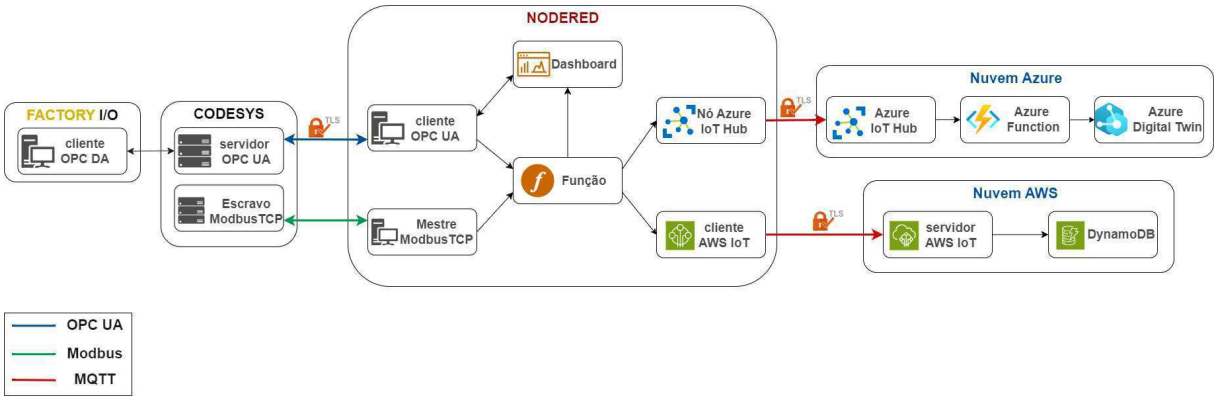
Neste capítulo, é apresentado o projeto integrado que visa demonstrar o funcionamento de um fluxo de comunicação fim-a-fim em um cenário de Indústria 4.0. Este fluxo integra diferentes tecnologias e protocolos para interconexão e mapeamento dinâmico entre dispositivos, abordando os principais desafios e soluções implementadas ao longo deste trabalho. O objetivo principal é ilustrar como os diversos componentes, tais como Factory I/O, CODESYS, Node-RED, e plataformas de nuvem (AWS e Azure), interagem para criar uma infraestrutura de produção inteligente e eficiente.

O cenário desenvolvido no projeto simula uma fábrica que opera de maneira autônoma, permitindo monitoramento, controle e processamento de dados em tempo real. A comunicação entre os dispositivos industriais utiliza protocolos como OPC UA e Modbus, enquanto o Node-RED atua como orquestrador central, integrando e processando dados provenientes dos dispositivos e publicando-os nas plataformas de nuvem (AWS e Azure) via MQTT. O projeto implementa a interconexão entre sistemas ciberfísicos e a nuvem, incorporando gêmeos digitais para representar os elementos físicos no ambiente digital, possibilitando, assim, uma análise detalhada e uma otimização dos processos industriais.

5.1 Arquitetura Geral do Sistema

A Figura 48 apresenta a arquitetura geral do cenário integrado desenvolvido neste projeto. Nela, é possível observar a interconexão entre os diferentes componentes do sistema, incluindo a comunicação entre os dispositivos industriais simulados, o orquestrador central (Node-RED) e os serviços de nuvem (AWS e Azure).

Figura 48 – Arquitetura do cenário de interconexão e comunicação Modbus com Node-RED e integração ao Azure.



Fonte: Elaborado pelo autor.

A arquitetura foi projetada para permitir a comunicação eficiente e segura entre sistemas heterogêneos utilizando protocolos como OPC UA, Modbus e MQTT. O Factory I/O simula o ambiente industrial, enquanto o CODESYS atua como um servidor OPC UA e um dispositivo escravo ModbusTCP. O Node-RED é responsável pela orquestração central, integrando-se com a nuvem para realizar o processamento e armazenamento de dados, bem como a atualização dos gêmeos digitais.

A Tabela 6 demonstra os dipositivos utilizados no caso de uso para demonstrar integração dos diversos tipos para atender às demandas de uma aplicação industrial. A capacidade do Node-RED de conectar esses dispositivos de forma eficiente é fundamental para o sucesso do sistema. A análise da quantidade e do tipo de dispositivos conectados é crucial para avaliar a eficácia dessa integração e identificar possíveis pontos de falha ou gargalos de comunicação.

Tabela 6 – Dispositivos Conectados ao Node-RED

Dispositivo	**Quantidade**	**Descrição**
Botões de controle	2	Permitem a interação manual com o sistema, como iniciar, parar ou resetar processos.
Leitura de RPM (motores 1 e 2)	2	Capturam a velocidade de rotação dos motores 1 e 2, fornecendo dados para monitoramento e controle.
Leitura de corrente (motores 1 e 2)	2	Medem a corrente elétrica consumida pelos motores 1 e 2, permitindo avaliar o consumo de energia e detectar possíveis sobrecargas.
Relógios temporizadores	3	Controlam o tempo de execução de processos, permitindo a automatização de tarefas e a criação de sequências temporizadas.
Contador de produtos	1	Registra o número de produtos processados, fornecendo dados para análise de produtividade e geração de relatórios.
Indicadores de funcionamento	2	Fornecem feedback visual sobre o estado do sistema, como luzes indicadoras de funcionamento normal ou alarmes.
Publicadores IoT	2	Encaminham dados do sistema para a nuvem ou outros dispositivos IoT, permitindo o monitoramento remoto e a integração com outros sistemas.
Total	**14**	

Fonte: Elaborado pelo autor.

5.2 Materiais e Métodos

Para este capítulo, o principal foco foi integrar os cenários desenvolvidos nos capítulos 4 e 5, criando um fluxo fim-a-fim de comunicação em um ambiente de Indústria 4.0. Este projeto integrador reúne a simulação industrial no Factory I/O, a configuração de controladores lógicos programáveis (CLPs) no CODESYS, a orquestração dos dados com o Node-RED e a comunicação com serviços de nuvem, formando uma infraestrutura completa e interconectada.

No Capítulo 4, a comunicação entre o Node-RED e a Linha de Produção simulada no Factory I/O com CLP programado no CODESYS, foi estabelecida através do protocolo OPC UA, com o CODESYS atuando como servidor OPC UA e o Factory I/O como cliente.

Já no Capítulo 5, a integração do CODESYS com o Node-RED ocorreu via protocolo ModbusTCP, onde o Node-RED foi configurado como mestre para acessar as variáveis do CLP e processá-las. Por fim, os dados foram enviados à nuvem usando o protocolo MQTT, conectando-se aos serviços da AWS e do Azure.

Nesta integração final, foi mantida a configuração dos componentes já descritos, adicionando uma camada de orquestração no Node-RED para gerenciar os dados provenientes de diferentes fontes e direcioná-los para a nuvem. A Figura 48 ilustra a arquitetura completa do sistema proposto. O ambiente Node-RED foi ajustado para receber dados do servidor OPC UA do CODESYS, processá-los conforme descrito no Capítulo 5 e, em seguida, publicar as informações no Azure IoT Hub e no AWS IoT, conforme detalhado nos capítulos anteriores.

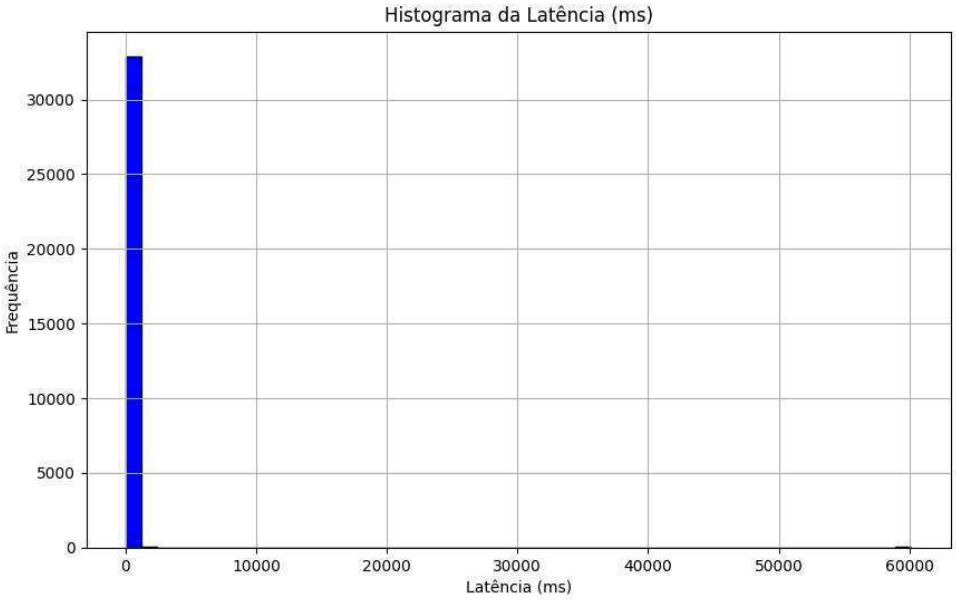
5.3 Análise e Discussão

Neste capítulo, foram analisados os três principais protocolos de comunicação utilizados no cenário de Indústria 4.0 desenvolvido: OPC UA, Modbus e MQTT. As capturas foram realizadas em diferentes momentos do fluxo de comunicação, e as análises focaram em métricas como latência, throughput, distribuição do tamanho de pacotes e taxa de transmissão. Além disso, foi monitorado o consumo de recursos do sistema durante a execução, conforme ilustrado na Figura 59.

5.3.1 Análise da Comunicação OPC UA

Para a análise da comunicação entre o servidor OPC UA e o Node-RED, foi realizada uma captura de tráfego utilizando o filtro ‘tcp port 4840‘. A latência média geral registrada durante a comunicação OPC UA foi de 0.324 segundos, com um throughput médio de 5.32 KB/s. O histograma da latência (Figura 49) indica que a maioria das mensagens apresenta baixa latência, com alguns outliers que ultrapassam valores mais elevados.

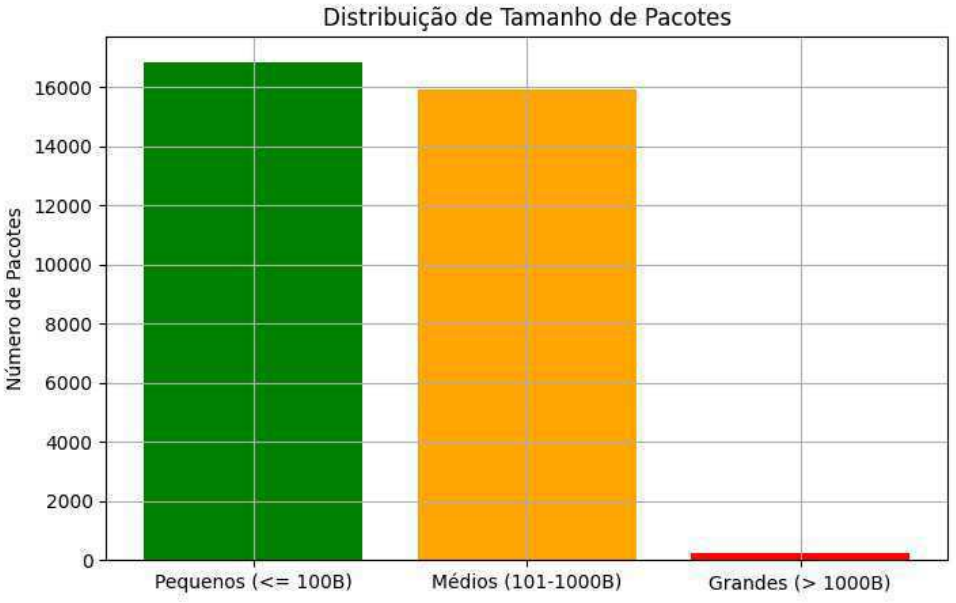
Figura 49 – Histograma da Latência (ms) durante a captura de tráfego OPC UA.



Fonte: Elaborado pelo autor.

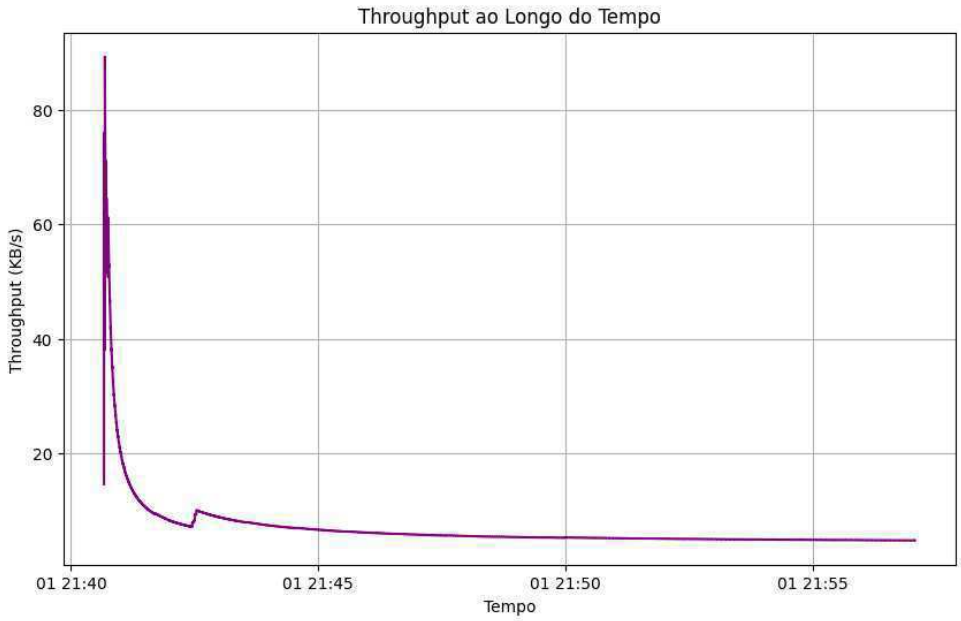
A distribuição do tamanho dos pacotes capturados (Figura 50) mostrou que a comunicação é predominantemente composta por pacotes pequenos (<= 100 bytes) e médios (101-1000 bytes), com uma pequena quantidade de pacotes grandes (> 1000 bytes). O gráfico de throughput ao longo do tempo (Figura 51) apresentou um pico inicial de transmissão que se estabilizou rapidamente.

Figura 50 – Distribuição do Tamanho de Pacotes durante a captura de tráfego OPC UA.



Fonte: Elaborado pelo autor.

Figura 51 – Throughput ao longo do tempo (KB/s) durante a captura de tráfego OPC UA.



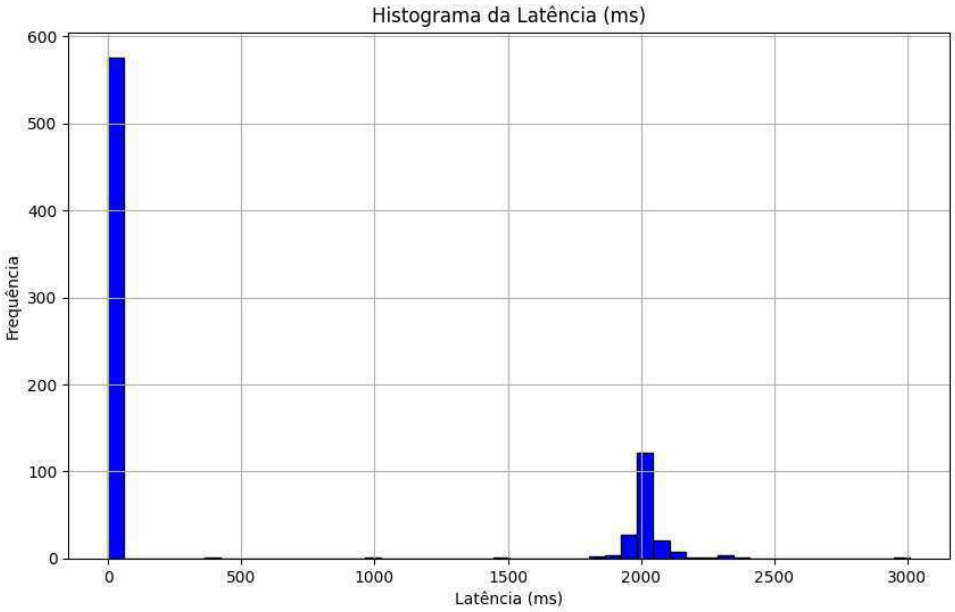
Fonte: Elaborado pelo autor.

De modo geral, a comunicação OPC UA apresentou bom desempenho, com baixa latência média e um padrão de transmissão estável.

5.3.2 Análise da Comunicação Modbus

Na comunicação Modbus, utilizando o filtro ‘tcp port 502‘, a latência média foi de 0.500 segundos e o throughput médio foi de 0.14 KB/s. O histograma da latência (Figura 52) revelou dois picos distintos, indicando variações na resposta do servidor ou possíveis congestionamentos de rede.

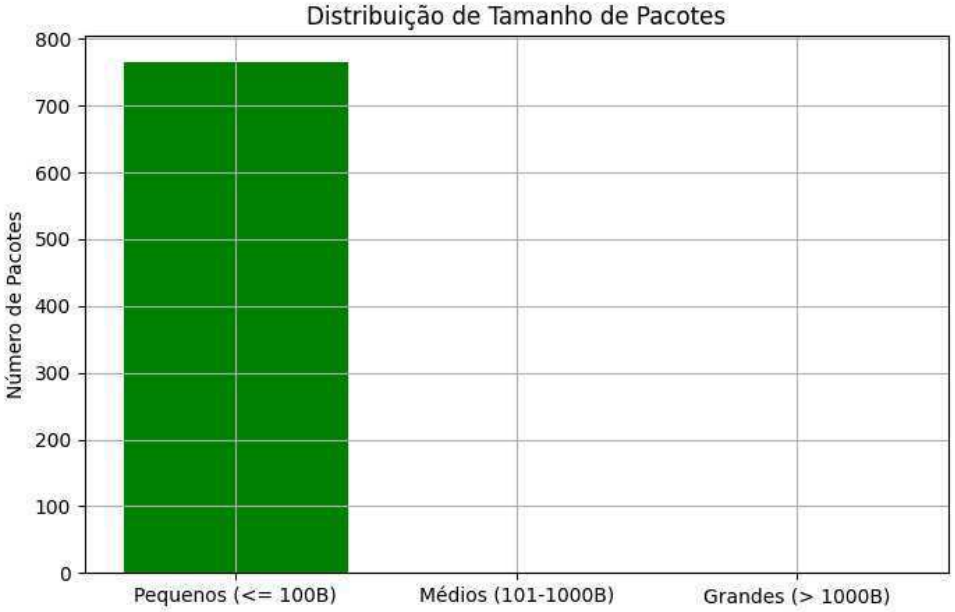
Figura 52 – Histograma da Latência (ms) durante a captura de tráfego Modbus.



Fonte: Elabordo pelo autor.

Todos os pacotes capturados foram pequenos (<= 100 bytes), conforme mostrado na Figura 53, o que é típico do protocolo Modbus. O gráfico de throughput (Figura 55) evidenciou um pico inicial seguido por uma estabilização, refletindo o padrão esperado das operações Modbus.

Figura 53 – Distribuição do Tamanho de Pacotes durante a captura de tráfego Modbus.



Fonte: Elaborado pelo autor.

Figura 54 – Throughput ao longo do tempo (KB/s) durante a captura de tráfego Modbus.

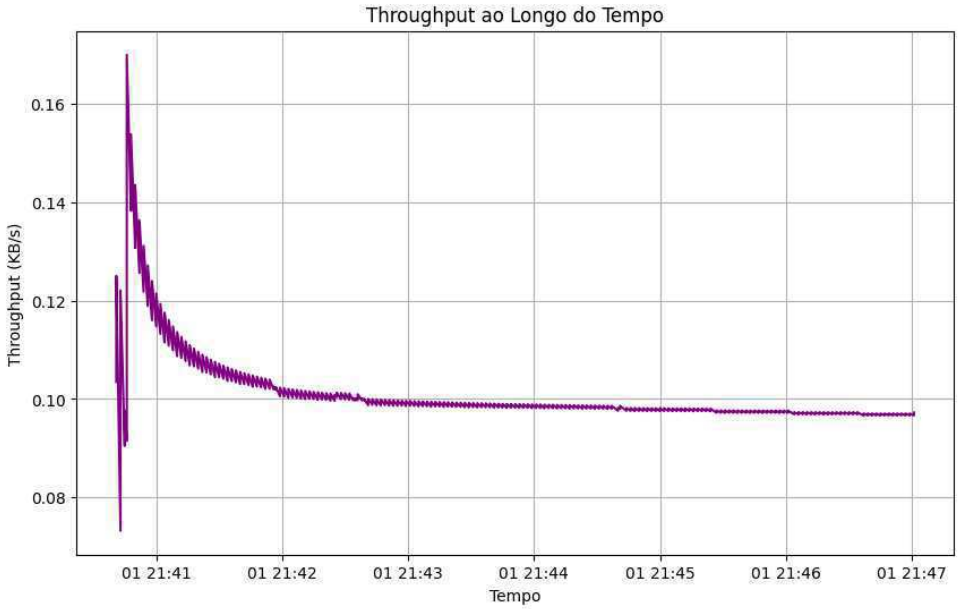
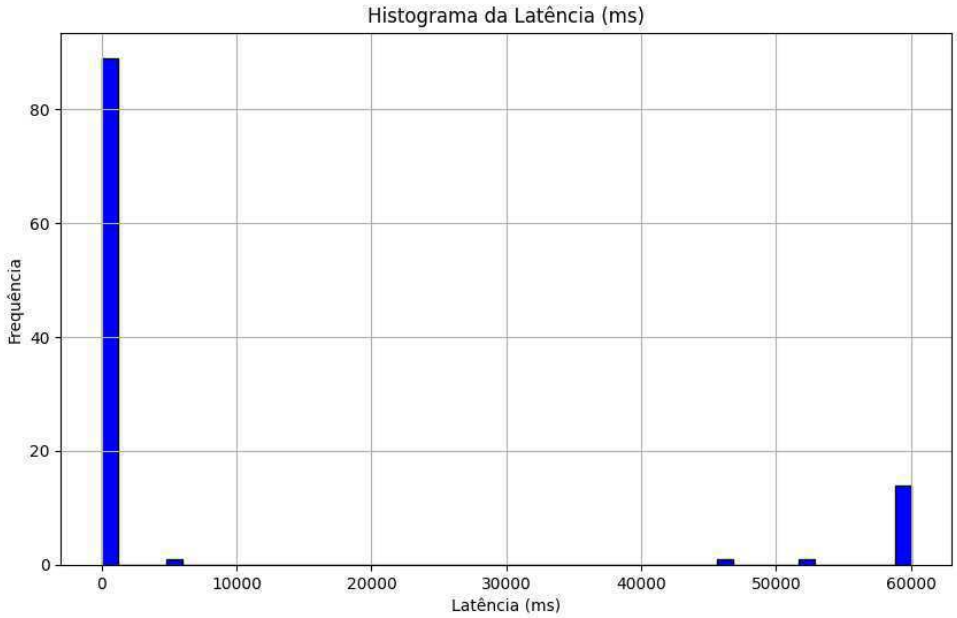


Figura 55 – Fonte: Elaborado pelo autor.

5.3.3 Análise da Comunicação MQTT

A análise da comunicação MQTT, realizada com o filtro ‘tcp port 8883‘, mostrou uma latência média de 8.929 segundos e um throughput médio de 0.03 KB/s. O histograma da latência (Figura 56) apresentou picos significativos, especialmente acima de 60.000 ms, indicando possíveis atrasos na transmissão ou processamento ao se comunicar com a nuvem.

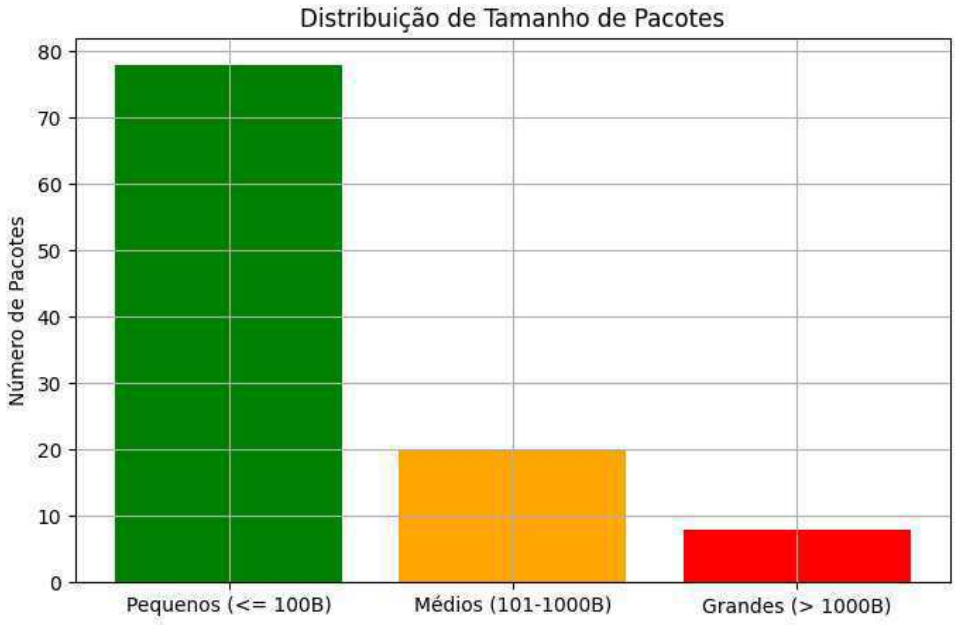
Figura 56 – Histograma da Latência (ms) durante a captura de tráfego MQTT.



Fonte: Elaborado pelo autor.

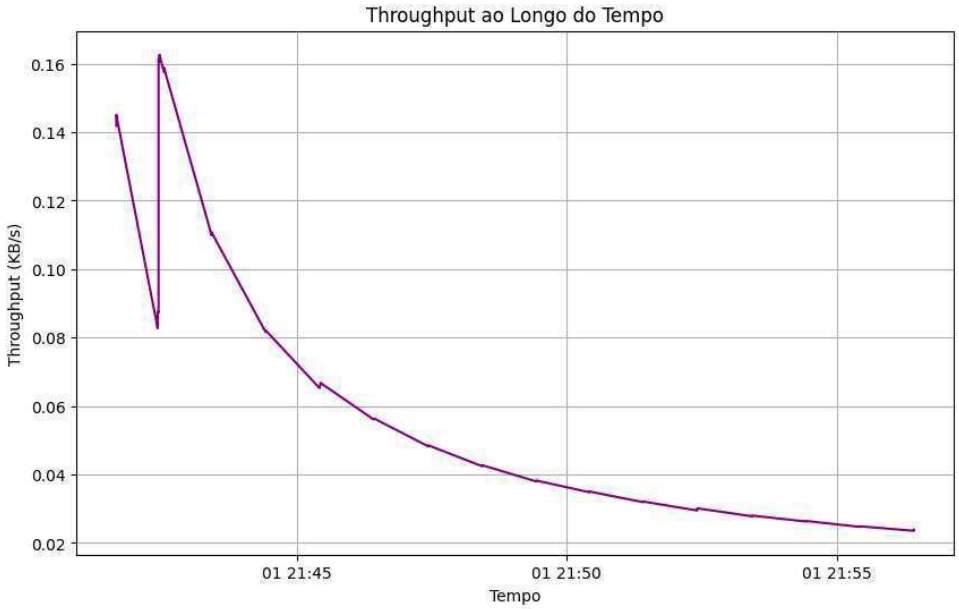
A distribuição do tamanho dos pacotes (Figura 57) revelou que a maioria era pequena (<= 100 bytes), com alguns pacotes médios e grandes. A análise do throughput ao longo do tempo (Figura 58) mostrou um padrão decrescente, com um pico inicial seguido por uma redução constante.

Figura 57 – Distribuição do Tamanho de Pacotes durante a captura de tráfego MQTT.



Fonte: Elaborado pelo autor.

Figura 58 – Throughput ao longo do tempo (KB/s) durante a captura de tráfego MQTT.

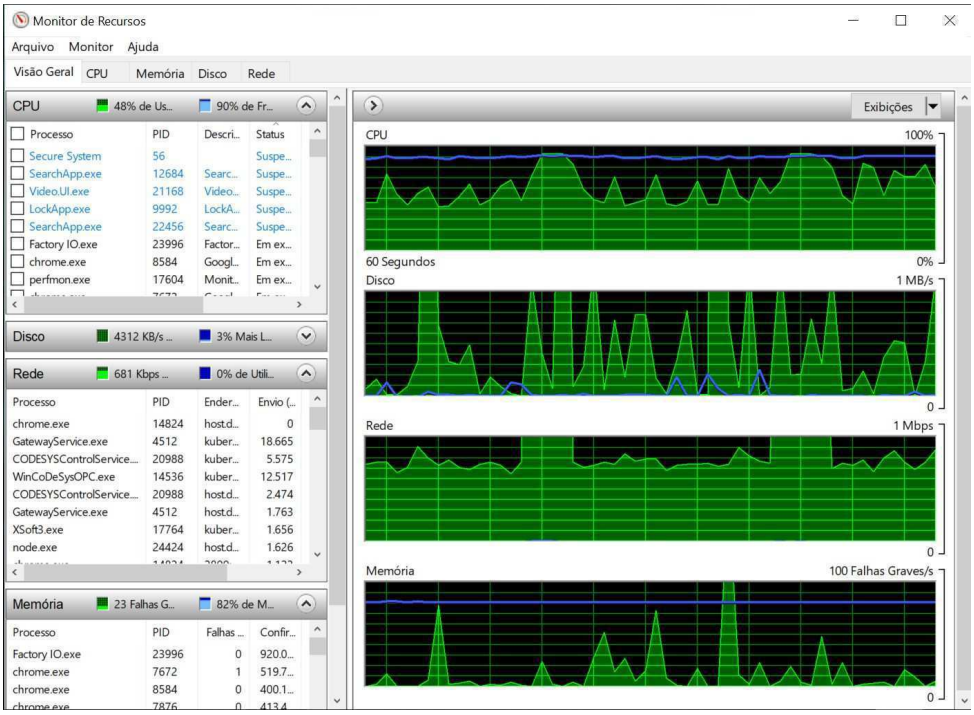


Fonte: Elaborado pelo autor

5.3.4 Consumo de Recursos do Sistema

O monitoramento do consumo de recursos do sistema durante os testes revelou que, embora o uso da CPU e da memória tenha variado (conforme Figura 59), ele permaneceu dentro de níveis aceitáveis para as operações realizadas. A comunicação OPC UA e Modbus teve um impacto relativamente baixo no uso da rede, enquanto o protocolo MQTT, devido à comunicação com a nuvem, apresentou picos de tráfego.

Figura 59 – Monitoramento de recursos do sistema durante os testes de comunicação.



Fonte: Elaborado pelo autor.

5.3.5 Discussão Geral

As análises realizadas evidenciam comportamentos distintos para cada protocolo de comunicação. O OPC UA apresentou um desempenho estável com baixa latência e throughput moderado, sendo adequado para cenários de monitoramento e controle em tempo real. O Modbus, apesar da sua simplicidade e latência ligeiramente superior, mostrou-se eficiente para trocas regulares de dados em redes industriais.

Por outro lado, o MQTT apresentou latência elevada e um throughput muito baixo, além de picos significativos, sugerindo que, embora seja eficiente para transmissões esporádicas e compactas, enfrenta desafios em cenários que demandam respostas em tempo real. Esses resultados são coerentes com as características dos protocolos, onde o MQTT é geralmente utilizado para comunicação de longa distância e transmissão de dados compactos.

Em resumo, a escolha do protocolo de comunicação depende do contexto da aplicação e das restrições do sistema. O cenário desenvolvido integra essas três tecnologias de forma a aproveitar os pontos fortes de cada uma, criando uma infraestrutura flexível e adaptável ao ambiente de Indústria 4.0.

6 Conclusão

A solução desenvolvida neste projeto resultou em um sistema completo para interconexão e mapeamento dinâmico de dispositivos em um cenário de Indústria 4.0. Essa solução é composta por uma arquitetura modular e integrada que utiliza protocolos e *frameworks* de IoT para interligar Controladores Lógicos Programáveis (CLPs), sensores industriais, gêmeos digitais, e plataformas em nuvem. Os principais elementos da solução são detalhados a seguir:

- **Interconexão entre dispositivos industriais:** Foi desenvolvida uma infraestrutura de comunicação capaz de interconectar diferentes dispositivos industriais, como CLPs e sensores, utilizando o protocolo OPC UA para comunicação no chão de fábrica e MQTT para integração com sistemas externos, como a nuvem AWS. Essa interconexão garante uma troca de informações eficiente e segura, refletindo as operações em tempo real.

- **Mapeamento dinâmico de recursos:** A solução permite o mapeamento dinâmico dos recursos presentes no ambiente industrial, facilitando a identificação e organização das interconexões entre os dispositivos. Com a integração de *frameworks* como Node-RED, foi possível implementar uma configuração adaptativa dos dispositivos, possibilitando mudanças nas operações e atualizações automáticas da infraestrutura de comunicação.

- **Implementação de gêmeos digitais:** Uma parte fundamental da solução foi a implementação de gêmeos digitais utilizando DTDL e a plataforma Azure IoT. Esses gêmeos digitais criam representações virtuais dos dispositivos físicos, permitindo o monitoramento, análise e simulação das operações industriais. Isso aprimora a capacidade de tomar decisões informadas em tempo real, além de possibilitar a manutenção preditiva dos sistemas.

- **Integração com plataformas em nuvem:** A solução inclui a integração dos dados coletados no ambiente industrial com plataformas em nuvem, como AWS e Azure. Essa integração proporciona o armazenamento seguro e análise avançada dos dados, além de facilitar a visualização das informações em painéis personalizados para controle e monitoramento.

- **Flexibilidade e adaptabilidade:** A arquitetura modular da solução foi projetada para ser flexível e adaptável a diferentes cenários industriais. Através do uso de *frameworks* de IoT, como Node-RED, é possível configurar e reconfigurar a interconexão

dos dispositivos, facilitando a adaptação do sistema a novas demandas produtivas e mudanças nas operações.

A integração final de todos os microprojetos demonstrou a eficácia da solução ao simular um ambiente industrial complexo, onde a interconexão dinâmica entre diferentes dispositivos foi estabelecida com sucesso. O sistema resultante mostrou-se capaz de coletar, processar e mapear dados em tempo real, integrando-se com gêmeos digitais e plataformas em nuvem. Essa solução, portanto, atende aos requisitos da Indústria 4.0, fornecendo um ambiente industrial inteligente, interconectado e preparado para operações flexíveis e adaptáveis.

6.1 Trabalhos Futuros

Apesar dos resultados alcançados, este trabalho abre espaço para futuras pesquisas e melhorias, tais como:

- Implementação de técnicas de inteligência artificial e aprendizado de máquina para aprimorar a análise preditiva e a tomada de decisões com base nos dados coletados;
- Exploração de outras tecnologias e protocolos de comunicação, como 5G e TSN (Time-Sensitive Networking), para avaliar seu impacto em ambientes industriais;
- Estudo da integração com sistemas de gestão empresarial (ERP) para criar uma visão mais abrangente da fábrica, conectando o chão de fábrica com os processos de negócios.

Bibliografia

Aazam, Mohammad, Khaled A. Harras e Sherali Zeadally (2019). "Fog Computing for 5G Tactile Industrial Internet of Things: QoE-Aware Resource Allocation Model". Em: *IEEE Transactions on Industrial Informatics* 15.5, pp. 3085–3092.

Bibow, Pascal et al. (2020). "Model-Driven Development of a Digital Twin for Injection Molding". Em: *Advanced Information Systems Engineering*. Ed. por Schahram Dustdar et al. Cham: Springer International Publishing, pp. 85–100. ISBN: 978-3-030-49435-3.

Digital Twin Consortium (2024). *Manufacturing Ontologies*. <https://github.com/digitaltwinconsortium/ManufacturingOntologies/tree/main?tab=readme-ov-file>. Acessado em: 18 de setembro de 2024.

Elamanov, Sherzod et al. (2024). "Interworking between Modbus and internet of things platform for industrial services". Em: *Digital Communications and Networks* 10.2, pp. 461–471. ISSN: 2352-8648. DOI: <https://doi.org/10.1016/j.dcan.2022.09.013>. URL: <https://www.sciencedirect.com/science/article/pii/S2352864822001882>.

Enany, Marwa O Al, Hany M. Harb e Gamal Attiya (2021). "A New Back-off Algorithm with Priority Scheduling for MQTT Protocol and IoT Protocols". Em: *International Journal of Advanced Computer Science and Applications* 12.11. DOI: <10.14569/IJACSA.2021.0121140>. URL: <http://dx.doi.org/10.14569/IJACSA.2021.0121140>.

Ferrer, Borja Ramis et al. (2015). "Potentials of Web Standards for Automation Control in Manufacturing Systems". Em: *2015 IEEE European Modelling Symposium (EMS)*, pp. 359–366. DOI: <10.1109/EMS.2015.59>.

González, Mikel et al. (2020). "A Digital Twin for Operational Evaluation of Vertical Transportation Systems". Em: *IEEE Access* 8, pp. 114389–114400. DOI: <10.1109/ACCESS.2020.3001686>.

IEC 62264 (2024). <https://en.wikipedia.org/wiki/IEC_62264>. Acessado em: 18 de setembro de 2024.

Iglesias-Urkia, Markel et al. (2017). "Towards a lightweight protocol for Industry 4.0: An implementation based benchmark". Em: *2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM)*, pp. 1–6. DOI: <10.1109/ECMSM.2017.7945894>.

ISA-95 (2024). <https://en.wikipedia.org/wiki/ANSI/ISA-95>. Acessado em: 18 de setembro de 2024.

Luan, Tom H. et al. (2021). "The Paradigm of Digital Twin Communications". Em: *CoRR* abs/2105.07182. arXiv: <2105.07182>. URL: <https://arxiv.org/abs/2105.07182>.

Marcon, P. et al. (2017). "Communication technology for industry 4.0". Em: *2017 Progress In Electromagnetics Research Symposium - Spring (PIERS)*, pp. 1694–1697. DOI: <10.1109/PIERS.2017.8262021>.

Microsoft Azure (2024). *How Azure Digital Twins works*. Acessado em: 17 de setembro de 2024. URL: <https://learn.microsoft.com/en-us/training/modules/introduction-to-azure-digital-twins/3-how-azure-digital-twins-works>.

Node-RED (2024). *About Node-RED*. Acessado em: 17 de setembro de 2024. URL: <https://nodered.org/about/>.

Peralta-Abarca, Jesús del Carmen, Beatriz Martínez-Bahena e Juana Enríquez-Urbano (mar. de 2021). "Industria 4.0". Em: *Inventio* 16.39, pp. 1–7. DOI: <10.30973/inventio/2020.16.39/4>. URL: <https://inventio.uaem.mx/index.php/inventio/article/view/45>.

Pribiš, Rudolf, Lukáš Beňo e Peter Drahoš (2020). "An Industrial Communication Platform for Industry 4.0 - case study". Em: *2020 Cybernetics Informatics (KI)*, pp. 1–9. DOI: <10.1109/KI48306.2020.9039873>.

Rahadian, Helmy et al. (2022). "Open Source OPC UA Data Traffic Characteristic and Anomaly Detection using Image-Encoding based Convolutional Neural Network". Em: *2022 7th International Conference on Electric Vehicular Technology (ICEVT)*, pp. 52–59. DOI: <10.1109/ICEVT55516.2022.9925002>.

Singh, Maulshree et al. (2021). "Digital Twin: Origin to Future". Em: *Applied System Innovation* 4.2. ISSN: 2571-5577. DOI: <10.3390/asi4020036>. URL: <https://www.mdpi.com/2571-5577/4/2/36>.

Zezulka, F. et al. (2018). "Communication Systems for Industry 4.0 and the IIoT". Em: *IFAC-PapersOnLine* 51.6. 15th IFAC Conference on Programmable Devices and Embedded Systems PDeS 2018, pp. 150–155. ISSN: 2405-8963. DOI: <https://doi.org/10.1016/j.ifacol.2018.07.145>. URL: <https://www.sciencedirect.com/science/article/pii/S2405896318308899>.

Apêndices

APÊNDICE A –

A.1 Função IoTHubToADT em C#

Listing A.1 – Função para processar eventos do IoT Hub e atualizar gêmeos digitais no Azure Digital Twins.

```
using System;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Extensions.EventGrid;
using Microsoft.Extensions.Logging;
using Azure.Messaging.EventGrid;
using Azure;
using Azure.Core.Pipeline;
using Azure.Identity;
using Newtonsoft.Json.Linq;
using Newtonsoft.Json;
using System.Threading.Tasks;
using System.Net.Http;
using Azure.DigitalTwins.Core;

namespace DigitalTwinsTraining
{
    public class Function1
    {
        private static readonly HttpClient httpClient = new
            HttpClient();
        private static string adtServiceUrl = Environment.
            GetEnvironmentVariable("ADT_SERVICE_URL");

        [FunctionName("IoTHubToADT")]
        public async Task Run([EventGridTrigger] EventGridEvent
            eventGridEvent, ILogger log)
        {
            log.LogInformation(eventGridEvent.Data.ToString());

            var credentials = new DefaultAzureCredential();
            DigitalTwinsClient client = new DigitalTwinsClient(
```

```
            new Uri(adtServiceUrl), credentials, new
                DigitalTwinsClientOptions
            { Transport = new HttpClientTransport(httpClient
                ) });
        log.LogInformation($"Conexao com o cliente ADT
            criada.");

        if (eventGridEvent != null && eventGridEvent.Data !=
            null)
        {
            try
            {
                log.LogInformation(eventGridEvent.Data.
                    ToString());

                // Decodificar a mensagem de base64
                JObject deviceMessage = JObject.Parse(
                    eventGridEvent.Data.ToString());
                string body = deviceMessage["body"].ToString
                    ();
                string decodedBody = System.Text.Encoding.
                    UTF8.GetString(Convert.FromBase64String(
                    body));
                log.LogInformation($"Mensagem decodificada:
                    {decodedBody}");

                // Parse a mensagem JSON
                JObject oeeMessage = JObject.Parse(
                    decodedBody);
                double oee = (double)oeeMessage["OEE"];
                log.LogInformation($"OEE: {oee}");

                // Definir o ID do gemeo digital a ser
                    atualizado
                string twinId = "FLM1_Esteira1_OEE";

                // Criar um documento JSON Patch para
                    atualizar a propriedade "value"
                var updateTwinData = new JsonPatchDocument()
```

```
                    ;
                updateTwinData.AppendReplace("/value", oee.
                    ToString());
                updateTwinData.AppendReplace("/
                    valueUnitOfMeasure", "%");

                // Atualizar o gemeo digital
                await client.UpdateDigitalTwinAsync(twinId,
                    updateTwinData);
                log.LogInformation($"Gemeo digital {twinId}
                    atualizado com o valor do OEE.");
            }
            catch (Exception ex)
            {
                log.LogError($"Erro ao processar o evento: {
                    ex.Message}");
                throw;
            }
        }
    }
}
```

s