Clarice Sofia Henrique Soares

Arcabouço de Integração do Node-RED com Ambientes Simulados Industriais

Campina Grande, PB

2024

Clarice Sofia Henrique Soares

Arcabouço de Integração do Node-RED com Ambientes Simulados Industriais

Trabalho de Conclusão de Curso (TCC) submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Graduado em Engenharia Elétrica.

Universidade Federal de Campina Grande – UFCG

Centro de Engenharia Elétrica e Informática

Departamento de Engenharia Elétrica

Orientador: Danilo Freire de Souza Santos

Campina Grande, PB

2024

Clarice Sofia Henrique Soares

Arcabouço de Integração do Node-RED com Ambientes Simulados Industriais

Trabalho de Conclusão de Curso (TCC) submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Graduado em Engenharia Elétrica.

Trabalho aprovado. Campina Grande, PB, _____/_____/_____.

Danilo Freire de Souza Santos
Orientador

Gutemberg Gonçalves dos Santos Júnior
Convidado

Campina Grande, PB
2024

À minha família.

Agradecimentos

Agradeço aos meus pais, Cristiane e George, pelo apoio, compreensão e carinho verdadeiramente inigualáveis. Sem o apoio deles, minhas conquistas nunca teriam sido possíveis. Sou muito grata aos meus irmãozinhos Letícia e Pedro por todos os momentos de alegria. Agradeço imensamente a minha dupla de jornada, Ítalo que foi um pilar pra mim desde o nosso primeiro período em engenharia elétrica.

Quero expressar meu carinho aos meus amigos Julia, Mayra, Matheus, Samara, Igor e Guilherme com quem eu dividi momentos íncriveis ao longo dos anos de UFCG. Obrigada às minhas amigas Néria, Marcella, Cecília, Thayse e Gabriela que mesmo sendo de outros cursos estiveram ao meu lado durante todas as etapas da graduação.

Aproveito a oportunidade para expressar minha gratidão ao meu orientador o Prof. Dr. Danilo Freire de Souza Santos por todas as valiosas orientações e conselhos que foram fundamentais para o desenvolvimento do trabalho. Por fim, agradeço ao corpo docente e à equipe técnica da UFCG que forneceram o suporte necessário para a conclusão deste TCC.

"Though the dawn right before the sunrise is darker than anything, never forget that the stars you longed for only rise in the darkness"

(SUGA)

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

DT	*Digital Twin*
IIoT	*Industrial Internet of Things*
IoT	*Internet of Things*
PLC	*Programmable Logic Controller*
TCP	*Transmission Control Protocol*
IP	*Internet Protocol*
MQTT	*Message Queuing Telemetry Transport*
QoS	*Quality of Service*

Resumo

No âmbito da Indústria 4.0, observa-se uma crescente interconexão de dispositivos de campo, máquinas e fábricas em redes de comunicação locais e globais. Essa conectividade facilita o controle e a análise remota desses dispositivos. O *Node-RED* emerge como uma ferramenta proeminente para o desenvolvimento ágil de projetos em Internet das Coisas (IoT). Este relatório descreve a aplicação do *Node-RED* em conjunto com o simulador *Factory I/O* para estabelecer uma infraestrutura que integra ambientes simulados de processos industriais a uma interface de usuário, assegurando a sincronização de todos os componentes do sistema.

Palavras chave: *Node-RED*, simulação, Indústria 4.0, *low-code*, *dashboard*.

Abstract

Within the scope of Industry 4.0, there is an increasing interconnection of field devices, machines and factories in local and global communication networks. This connectivity facilitates remote control and analysis of these devices. Node-RED emerges as a prominent tool for the agile development of Internet of Things (IoT) projects. This report describes the application of Node-RED in conjunction with the Factory I/O simulator to establish an infrastructure that integrates simulated industrial process environments into a user interface, ensuring synchronization of all system components.

Keywords: Node-RED, simulation, Industry 4.0, low-code, dashboard.

Sumário

1 Introdução

O *Node-RED* é uma ferramenta *low-code* útil para o desenvolvimento de sistemas para a Internet. É de uso simples e possui uma variedade de aplicações. Foi originalmente desenvolvido pela IBM e atualmente possui muitos colaboradores ativos.

A programação com *Node-RED* é realizada a partir de fluxos de dados que podem se conectar entre si e possuem blocos denominados "nós" (*nodes*) que são simples de serem conectados e configurados. O usuário pode implementar blocos funcionais com Javascript e existem centenas de nós *Node-RED* que são atualizados e podem ser importados para novos projetos (MUSUNGATE; TUNCAY, 2023).

O *Node-RED* está disponível para muitas plataformas e pode ser conectado com dispositivos de *hardware*. Essa ferramenta também é capaz de criar *dashboards* que facilitam a visualização da saída do sistema. Além disso, é possível conectá-lo a serviços online a partir de protocolos de comunicação.

Usando os nós, fluxos podem ser construídos para representar um processo ou atividade do mundo real. É possível moderar todos os elementos e suas relações, bem como os dados que são trocados entre eles.

Portanto, com o *Node-RED* é possível capturar, processar e analisar dados permitindo que o usuário possa atuar com base na informação obtida. No contexto industrial a flexibilidade gerada pelo desenvolvimento *low-code* é uma das principais soluções para realizar projetos de Internet das Coisas (do inglês *Internet of Things* - IoT) rapidamente.

O Gêmeo Digital (do inglês *Digital Twin* - DT) é um dos conceitos essenciais no contexto da Indústria 4.0, e sua importância cresce continuamente na academia e na indústria (PIRES et al., 2021). O DT é um reflexo virtual de um ativo físico que, por meio de coleta de dados e simuladores, permite controle, otimização, monitoramento e melhora a tomada de decisão.

Esse trabalho tem como objetivo utilizar o *Node-RED* e o simulador *Factory I/O* [1] na criação de um estrutura baseada nos conceitos de gêmeos digitais, capaz de conectar representações simuladas de situações reais do ambiente industrial com uma interface de usuário, mantendo todas as partes sincronizadas. A abordagem propõe o uso dos protocolos de comunicação Modbus TCP/IP (Modbus, 2024) e MQTT (MQTT, 2024) para realizar a conexão entre as partes da estrutura.

[1] Disponível em: <https://www.factoryio.com/>

1.1 Objetivo geral

Este trabalho tem como objetivo investigar a aplicação do *Node-RED* no contexto da Internet Industrial das Coisas (IIoT), analisando seus benefícios quanto à flexibilidade e simplicidade no desenvolvimento. Pretende-se desenvolver um painel de controle (*dashboard*) por meio do *Node-RED*, que possibilitará aos usuários o monitoramento remoto e o controle do comportamento de um tanque de líquido em uma fábrica, representados em um *software* de simulação. Com isso, espera-se demonstrar as amplas funcionalidades do *Node-RED* na indústria e estabelecer uma base para futuras implementações em outros projetos.

1.2 Objetivos específicos

Figuram os seguintes objetivos específicos:

- Estudar e compreender protocolos de comunicação como Modbus TCP/IP e MQTT;
- Estabelecer a integração do *Node-RED* com um ambiente simulado;
- Tratar e visualizar no *Node-RED* os dados recebidos pela simulação;
- Enviar informações para a simulação a partir do *Node-RED*;
- Utilizar o *Node-RED* para construir uma interface de usuário;
- Apresentar uma abordagem para a integração do *Node-RED* com ambientes simulados.

1.3 Organização do Documento

Este relatório está estruturado em 6 capítulos que têm como objetivo detalhar todos os passos realizados para o desenvolvimento deste trabalho e apresentar os resultados obtidos. O **Capítulo 1** apresenta uma visão geral do trabalho, bem como os objetivos e o método aplicado para alcançar os objetivos. No **Capítulo 2**, os detalhes das tecnologias usadas são explicados de maneira que o leitor possa entender as particularidades do projeto.

No **Capítulo 3** é apresentada a arquitetura do sistema. No **Capítulo 4**, os detalhes da implementação são explicados. No **Capítulo 5** é feita uma discussão dos resultados obtidos e como eles foram validados. Por fim, o **Capítulo 6** resume brevemente o que foi alcançado no trabalho.

1.4 Metodologia

Neste capítulo, será detalhado o método utilizado para alcançar os objetivos propostos neste trabalho. Inicialmente, foi realizada uma revisão bibliográfica de artigos acadêmicos sobre o tema "Gêmeos Digitais e *Node-RED*". Após a leitura, identificaram-se quais artigos seriam utilizados como referência. A revisão da literatura permitiu uma melhor compreensão do problema em estudo.

Em seguida, foram realizados cursos *online* de *Node-RED* para adquirir os conhecimentos necessários para fazer uso da ferramenta. Para a escolha do *software* de simulação, em um primeiro momento foi proposto o uso do FlexSim baseado no artigo (PIRES et al., 2021), entretanto, surgiram dificuldades de utilização. Devido às experiências obtidas na graduação, optou-se pelo uso do *software Factory I/O* para a simulação.

O desenvolvimento do projeto foi realizado gradualmente. Incialmente, foi feita uma versão para validação da possibilidade de conexão entre as ferramentas utilizadas. Posteriormente, foram adicionadas novas funcionalidades com base em *feedbacks* recebidos e novas descobertas. Ao final, ocorreu uma validação dos resultados obtidos.

2 Revisão Bibliográfica e Tecnológica

Neste capítulo são apresentadas as principais ferramentas e conceitos necessários para o desenvolvimento do projeto.

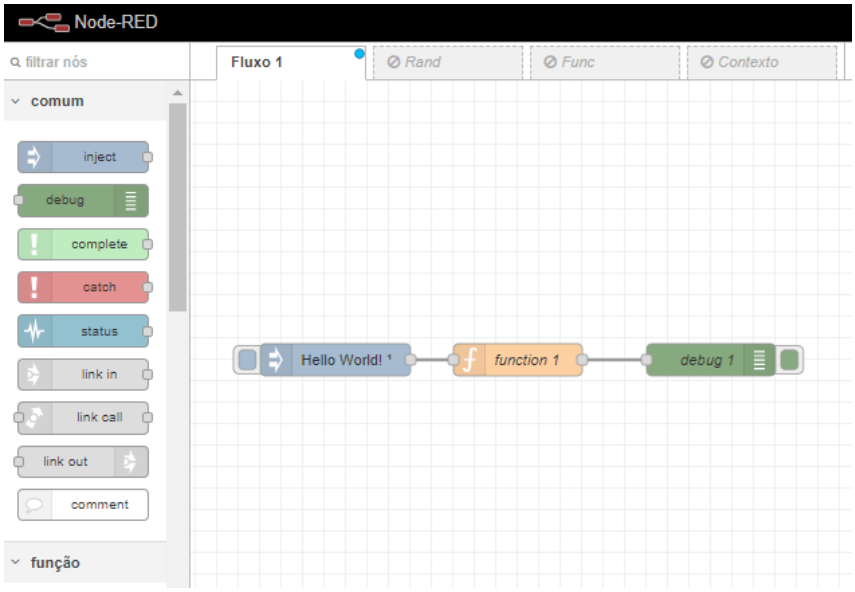
2.1 *Node-RED*

Node-RED é uma ferramenta amplamente utilizada, na qual a programação é baseada em fluxos e foi inventada em 2013 pelo time do grupo *IBM's Emerging Technology Services*. A programação baseada em fluxos foi criada em 1970 por J. Paul Morrison e descreve comportamentos com nós, criando uma representação visual acessível para diferentes usuários. Dessa maneira, torna-se possível criar soluções para problemas sem a necessidade de se aprofundar em todas as linhas de código.

De acordo com os criadores da ferramenta, ela recebe esse nome devido ao modelo baseado em nós (*nodes*) e fluxos, enquanto a parte do RED significa *Rapid Event Developer*. Atualmente pode ser usada em diversas aplicações, entretanto, iniciou como uma prova de conceito para visualizar e manipular mapeamentos entre tópicos MQTT (Node-RED, 2024).

Node-RED consiste em um ambiente de tempo de execução baseado em Node.js que aponta um navegador web para que o usuário acesse o editor de fluxo. Nele, é possível arrastar e soltar nós da paleta para uma área de trabalho para conectá-los, como pode ser observado na Figura 1.

Figura 1 – Interface do *Node-RED*.



Fonte: autoria própria (2024).

Por ser leve, o *Node-RED* pode ser executado em *hardware* como Raspberry Pi e na nuvem, utilizando Amazon Web Services, Microsoft Azure e outros. Por isso, tem sido bastante utilizado em aplicações IoT, como automação residencial (STEINMETZ et al., 2022).

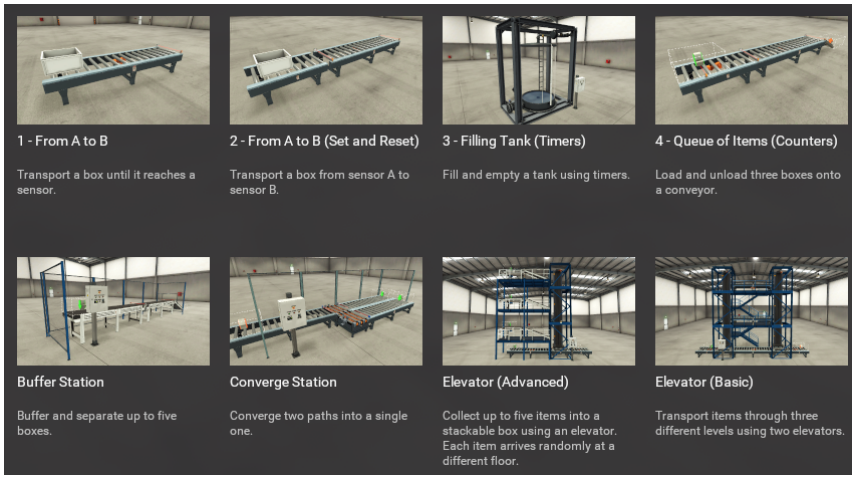
Além disso, funções escritas em JavaScript podem ser criadas para implementar diferentes funcionalidades. Essas funções podem ser salvas em uma biblioteca para posterior reutilização. Os fluxos criados no *Node-RED* podem ser armazenados em formato JSON e podem ser importados e exportados.

Node-RED possui uma comunidade ativa de usuários que constantemente desenvolve pacotes para adicionar funcionalidades. Esses pacotes ficam disponíveis para *download* em uma biblioteca online. Alguns deles são: *node-red-dashboard* e *node-red-contrib-modbus* (MUSUNGATE; TUNCAY, 2023).

2.2 *Factory I/O*

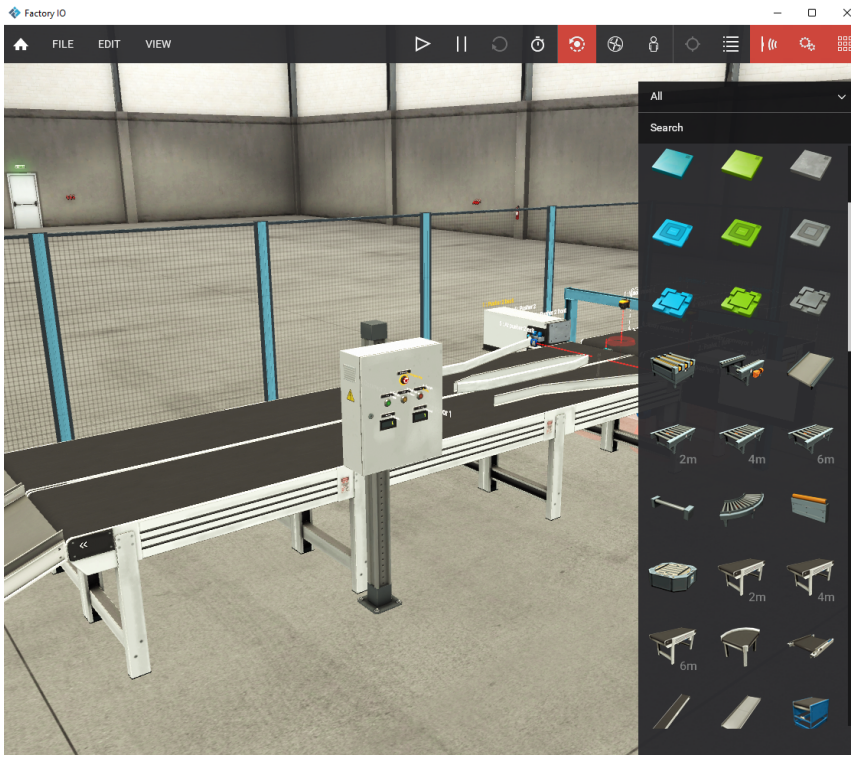
Factory I/O é um simulador 3D utilizado como ferramenta de ensino no campo da automação. É projetado para ser fácil de utilizar e inclui mais de vinte cenários inspirados em aplicações industriais reais com diferentes níveis de dificuldade (Figura 2) além de permitir que o usuário construa suas próprias cenas (Figura 3). O mais comum é utilizar o *Factory I/O* para o treinamento de Controladores Lógico Programáveis (CLPs), entretanto, pode ser utilizado com microcontroladores, SoftPLC, Modbus e outros (FACTORYIO, 2024).

Figura 2 – Biblioteca de Cenários no *Factory I/O*.



Fonte: autoria própria (2024).

Figura 3 – Exemplo de Cenário no *Factory I/O*.



Fonte: autoria própria (2024).

2.3 Protocolos de Comunicação

Esta seção aborda conceitos acerca dos protocolos de comunicação utilizados para o desenvolvimento deste trabalho.

2.3.1 Modbus TCP/IP

O Protocolo Modbus é uma estrutura de mensagens desenvolvida pela Modicon (atualmente Schneider Electric) em 1979. Utiliza arquitetura cliente-servidor para estabelecer a comunicação entre dispositivos inteligentes. O protocolo se tornou padrão na indústria para transferência de informações entre entradas e saídas analógicas ou digitais. Já TCP/IP é o protocolo de transporte de uso comum da internet (Modbus, 2024).

Modbus TCP/IP usa Rede local Ethernet (do inglês *Local Area Network* - LAN) ou rede global de área ampla (do *inglês Wide Area Network* - WAN) para transportar os dados da estrutura de mensagens Modbus entre dispositivos compatíveis.

O Modbus é um protocolo da camada de aplicação que define regras para organizar e interpretar os dados transmitidos. O modelo de dados Modbus possui uma estrutura simples que apenas diferencia entre alguns tipos de dados. A Tabela 1 destaca um conjunto de funções padrão do Modbus. As funções são utilizadas para acessar os registradores descritos no mapa de registradores do Modbus.

Tabela 1 – Funções do Modbus

Função	Descrição
1	Ler status de uma bobina
2	Ler status de uma entrada
3	Ler registradores de escrita
4	Ler registradores de entrada
5	Forçar uma bobina
6	Predefinir um registrador
15	Forçar múltiplas bobinas
16	Predefinir múltiplos registradores

Fonte: (Modbus, 2024).

O campo de dados da solicitação do cliente fornece ao servidor qualquer informação adicional exigida pelo serivdor para completar a ação especificada pelo código de função na solicitação do cliente. O campo de dados normalmente inclui endereços de registro e valores de contagem.

O Protocolo de Controle de Transporte (do inglês *Transmission Control Protocol* - TCP) fica na Camada de Transporte que reside logo abaixo da Camada de Aplicação e é responsável pela transmissão, recepção e verificação de erros dos dados. O TCP é geralmente considerado a camada superior da plataforma IP que serve para garantir a transferência segura de dados.

2.3.2 MQTT

MQTT é o protocolo de mensagens mais comumente usado para IoT e IIoT. MQTT significa *Message Queuing Telemetry Transport*. Ele foi projetado por um time da IBM em 1999 para conectar sistemas de telemetria de oleodutos por satélite (MQTT, 2024).

O protocolo é orientado a eventos e utiliza um padrão *Publisher/Subscriber* no qual quem envia informações é o *Publisher*. O *Subscriber* pode escolher tópicos para ser "assinante", e ele receberá informações apenas do tópico que assinou. A conexão entre os dois é feita utilizando um *broker* MQTT que filtra as mensagens recebidas e as distribui corretamente. Existem diversos *brokers* MQTT disponíveis para usar para testes e para aplicações reais. Alguns são gratuitos, sendo os mais populares o Mosquitto e o HiveMQ.

Uma das maiores vantagens do protocolo MQTT é a comunicação bidirecional, diferentes dispositivos podem se inscrever e publicar em diferentes tópicos. MQTT também permite a entrega confiável de mensagens, pois especifica diferentes níveis de Qualidade de Serviço (QoS). No quesito segurança, um *broker* pode identificar os clientes a partir de nomes de usuários e senhas, IDs de clientes ou com certificados SSL.

Tabela 2 – Qualidade de Serviço

QoS	Descrição
0	No máximo uma vez
1	Pelo menos uma vez
2	Exatamente uma vez

Fonte: (MQTT, 2024).

QoS 2 descrito na Tabela 2 equivale ao maior nível de qualidade de entrega de mensagens. O QoS 1 é um nível intermediário no qual o remetente mantém uma cópia da mensagem até receber uma mensagem de confirmação de recebimento vinda do receptor. Já o QoS 0 é o nível mais baixo e não garante a recepção das mensagens.

2.4 Gêmeos Digitais

Uma definição geral de Gêmeos Digitais é: "a cópia digital de um objeto ou sistema físico, que está conectado e compartilha dados funcionais e/ou operacionais". Em outras palavras, um Gêmeo Digital é uma representação virtual que modela um objeto físico. Existe uma conexão bidirecional entre a representação virtual e sua versão física.

Segundo (PIRES et al., 2020), o desenvolvimento de Gêmeos Digitais possui 5 fases principais:

- Virtualização de um objeto físico;
- Estabelecimento da comunicação entre a parte física e o gêmeo digital;
- Implementação de mecanismos de monitoramento (Ex.: *Dashboards* no *Node-RED*);
- Implementar mecanismos de tomada de decisão com inteligência artificial;
- Controle de operações aplicado pelo Gêmeo Digital no sistema físico.

Em (STEINMETZ et al., 2022), um caso de uso de um sistema *Car-as-a-Service (CaaS)* foi implementado no contexto de uma *Smart City*. O *Node-RED* é utilizado para a modelagem e implantação de DTs, ou seja, tem como escopo a principal funcionalidade do CaaS: solicitar um carro e conduzir o usuário de um ponto A a um ponto B. O simulador Carla é usado para construção de cenários de casos de uso e reproduzir o mundo real.

Outro caso de uso descreve um experimento que realiza o monitoramento dinâmico e em tempo real das condições de uma correia transportadora. O experimento utiliza uma Raspberry Pi para controle. Utiliza-se uma *dashboard* implementada em *Node-RED*. As informações (estados do motor e dos sensores, *timestamps*, o nível da bateria e a corrente no motor) são coletadas por meio do protocolo MQTT (PIRES et al., 2020).

Em (SCHROEDER et al., 2021) é proposta uma metodologia abrangente de modelagem de DTs que pode ser aplicada em diferentes domínios. O caso de estudo deste artigo envolve um sistema de refinaria de óleo que possui 4 válvulas automatizadas e um *watchdog*. A linguagem Automation ML (linguagem de modelagem de dados orientada a objetos) é utilizada para modelar o DT, e o *Node-RED* é utilizado para criar a *dashboard* na qual as válvulas são monitoradas.

3 Proposta de Solução

Uma etapa indispensável durante o processo de desenvolvimento de uma aplicação é realizar a definição de requisitos funcionais e não-funcionais com base nos objetivos pré-definidos. Estes requisitos são utilizados para a validação do projeto.

Tabela 3 – Requisitos Funcionais do Usuário

ID	**Descrição**
RF-U1	O usuário deve ser capaz de realizar ações de controle remotamente
RF-U2	O usuário deve ser capaz de visualizar em tempo real os dados obtidos

Fonte: autoria própria (2024).

Tabela 4 – Requisitos Não-Funcionais do Usuário

ID	**Descrição**
RNF-U1	O usuário deve ter acesso à uma interface intuitiva

Fonte: autoria própria (2024).

Tabela 5 – Requisitos Funcionais do Sistema

ID	**Descrição**
RF-S1	O sistema deve garantir a transmissão confiável dos dados coletados
RF-S2	O sistema deve ser capaz de processar os dados obtidos
RF-S3	O sistema deve permitir que o usuário envie comandos de controle para a simulação

Fonte: autoria própria (2024).

Tabela 6 – Requisitos Não-Funcionais do Sistema

ID	**Descrição**
RNF-S1	A interface deve ser projetada com uma disposição lógica das informações e opções para oferecer uma experiência de usuário intuitiva
RNF-S2	O sistema deve ter um tempo de resposta baixo

Fonte: autoria própria (2024).

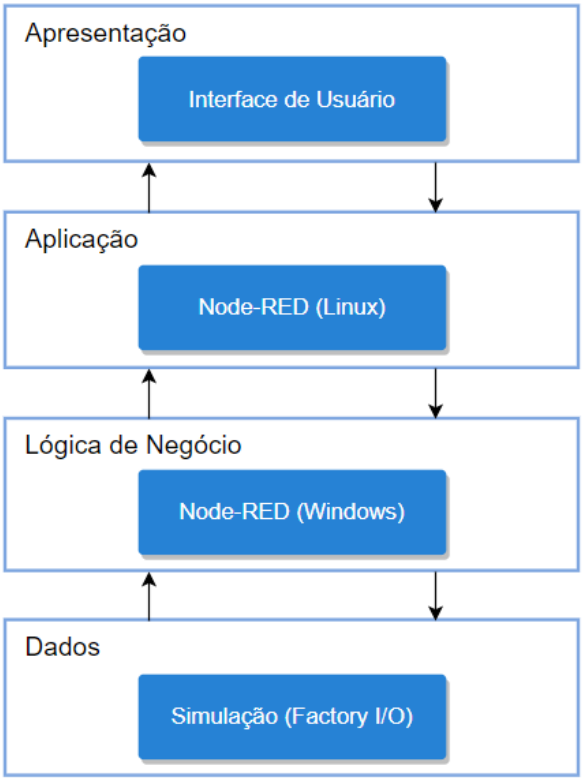
A abordagem proposta para atender os requisitos definidos baseia-se na utilização do *Node-RED* para criar uma interface de controle e monitoramento de um tanque de líquido em um ambiente industrial simulado, utilizando o *Factory I/O* como simulador. O *Node-RED* e o *Factory I/O* foram empregados devido às suas interfaces gráficas intuitivas e de simples uso.

Para exemplificar o acesso remoto às informações da simulação, algo essencial em aplicações IIoT, duas instâncias do *Node-RED* foram instaladas. Uma em um sistema operacional Windows, captura e trata os dados da simulação. A outra, em uma máquina virtual Linux, é responsável por construir a *dashboard* utilizada pelo usuário.

A *dashboard* fornecerá uma interface onde os dados do tanque serão apresentados de forma clara e acessível. Além disso, permitirá que o usuário controle as ações da máquina representada no *software* de simulação.

Para esta aplicação, optou-se pelo protocolo de comunicação Modbus TCP/IP, essa escolha se deve ao fato de o Modbus TCP/IP ser um padrão bem estabelecido na indústria. Além disso, o *Node-RED* oferece uma biblioteca de nós Modbus com suporte amplo e ótima avaliação pelos usuários. A Figura 4 abaixo apresenta uma visão arquitetural da solução.

Figura 4 – Visão lógica - diagrama de camadas



Fonte: autoria própria (2024).

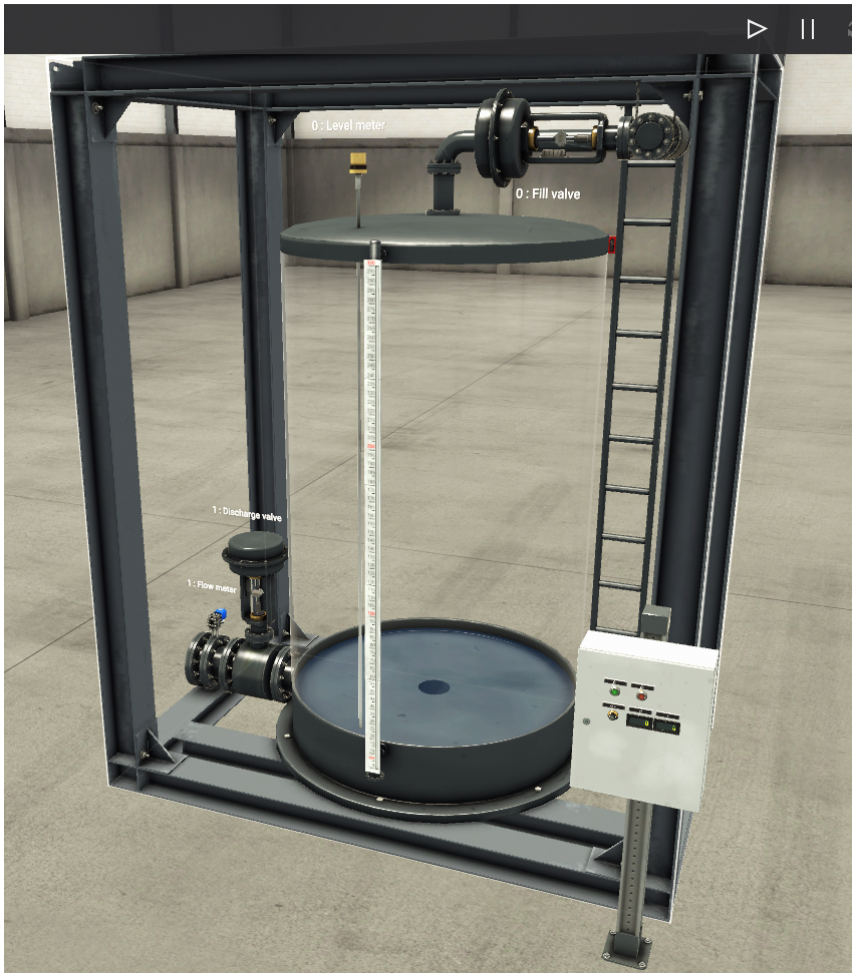
4 Desenvolvimento

Este capítulo apresenta os detalhes da implementação do trabalho descrito nos capítulos anteriores.

4.1 Cenário no *Factory I/O*

O cenário construido no *Factory I/O* (Figura 5) é referente ao enchimento de um tanque no qual o usário pode definir um valor de *setpoint* para o nível do líquido e escolher ligar ou desligar a válvula de enchimento de acordo com a necessidade. A Figura 6 exibe uma aproximação do painel de controle do tanque na simulação.

Figura 5 – Cenário simulado que representa o sistema físico.



Fonte: autoria própria (2024).

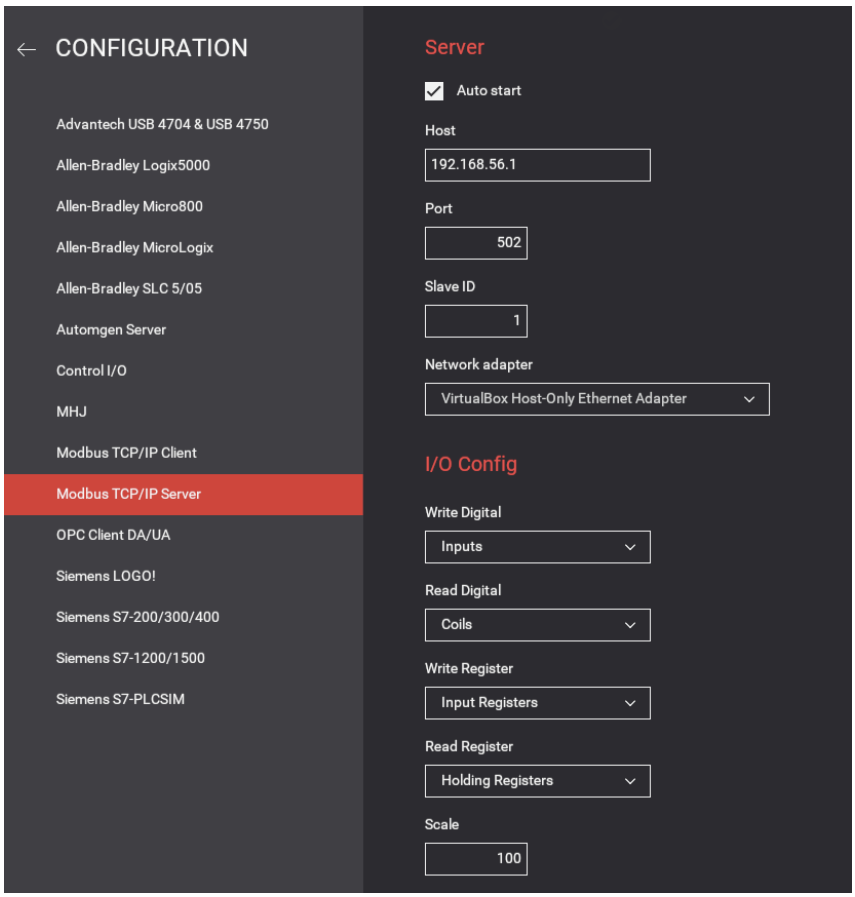
Figura 6 – Painel de controle do sistema "físico".

Fonte: autoria própria (2024).

Ao selecionar a opção *Modbus TCP/IP Server* nas opções de *drivers* do *Factory I/O*, as configurações do Servidor e das opções de Entrada e Saída são automaticamente definidas como na Figura 7.

Os sensores lidos nesse projeto são *Level Meter* (Nível) e *Setpoint*, ambos são registradores de entrada do tipo float e se encontram nos endereços 0 e 2 respectivamente. Os atuadores controlados são *Start Light* e *Stop Light* que são bobinas do tipo booleano e se encontram nos endereços 0 e 2 respectivamente. Por fim, também controla-se a *Fill Valve* (válvula de enchimento) e os *displays* digitais do *Setpoint* e do Nível que são registradores que podem ser do tipo float ou inteiro e se encontram nos endereços 0, 2 e 3 respectivamente (Figura 8).

Figura 7 – Configurações do servidor Modbus TCP/IP.



Fonte: autoria própria (2024).

Figura 8 – Sensores e atuadores do cenário.

Fonte: autoria própria (2024).

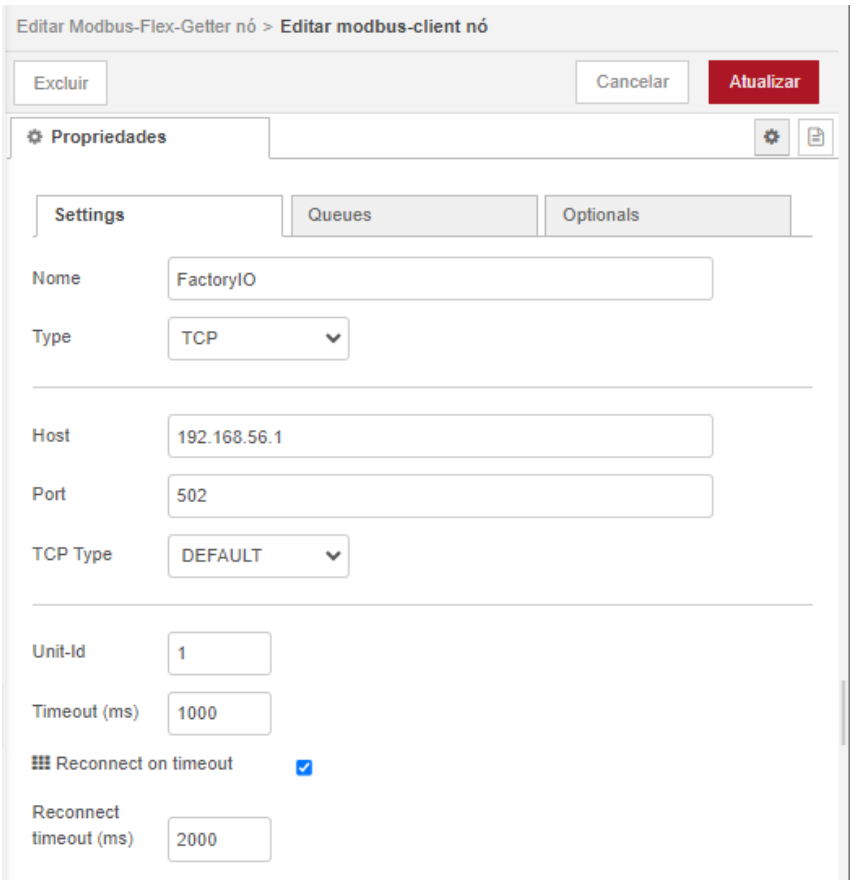
4.2 Leitura dos Dados da Simulação

O *Node-RED* possui uma biblioteca de nós denominada *node-red-contrib-modbus* que contém os nós necessários para utilizar o protocolo Modbus TCP/IP. Um deles é o nó de leitura *Modbus-Flex-Getter* que conecta-se a um servidor Modbus TCP para ler bobinas, entradas e registradores a medida que é acionado pelo nó Injetar característico do *Node-RED*.

O nó Injetar pode ser usado para acionar manualmente um fluxo clicando no botão do nó no editor ou para acionar fluxos automaticamente em intervalos regulares. A mensagem enviada pelo nó de Injetar pode ter seu *payload* e tópico definidos.

O nó *Modbus-Flex-Getter* aceita os códigos de funções descritos na Tabela 1. Ele pode ser configurado a partir de um nó de Função que pode passar os parâmetros unitid, *fc (function code), start address e quantity*. O nó tem as seguintes saídas: *array de dados*, *buffer* de resposta modbus e mensagem de entrada. Para o devido funcionamento do nó é necessário também configurar qual servidor está sendo acessado, no caso desse projeto o *Node-RED* atua como cliente e o *Factory I/O* como servidor (Figura 9).

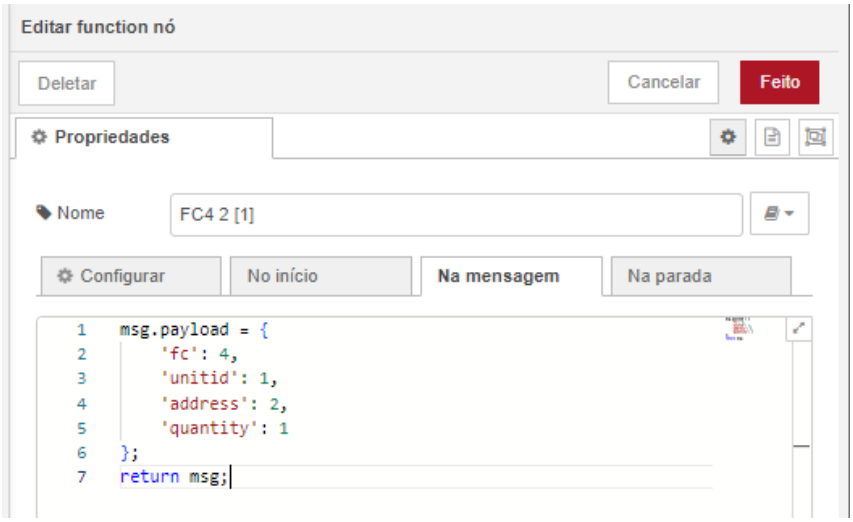
Figura 9 – Servidor acessado pelo cliente.



Fonte: autoria própria (2024).

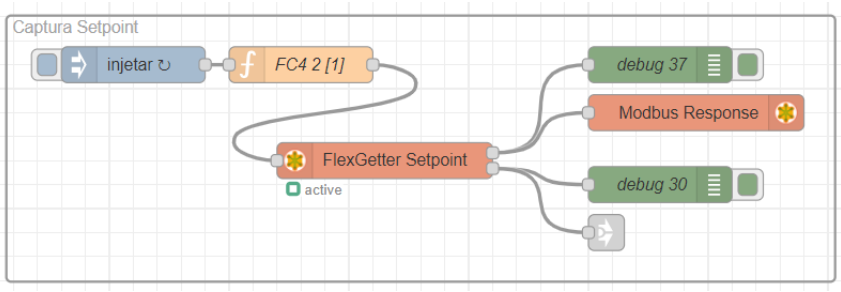
No fluxo descrito pela Figura 10 e pela Figura 11 é lido o valor único do *Setpoint* que é do tipo Registrador de Entrada e se encontra no endereço 2. O usuário pode ler essa informação ao observar o título da função que configura o *Modbus-Flex-Getter* (FC4 2 [1]).

Figura 10 – Configuração do nó de leitura.



Fonte: autoria própria (2024).

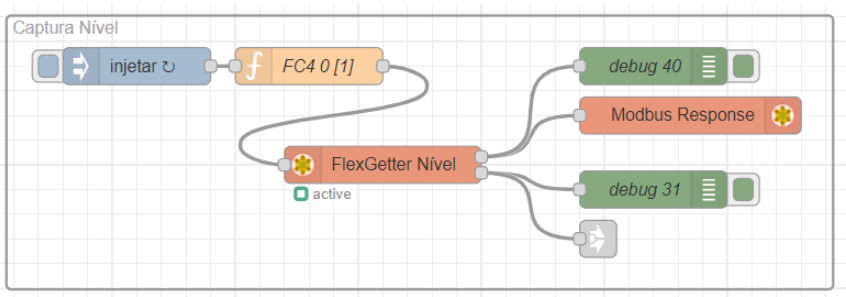
Figura 11 – Fluxo para captura do valor do *setpoint* na simulação.



Fonte: autoria própria (2024).

De maneira similar, pode-se fazer a leitura do valor do Nível que também é do tipo Registrador de Entrada e se encontra no endereço 0 (Figura 12) .

Figura 12 – Fluxo para captura do valor do nível na simulação.



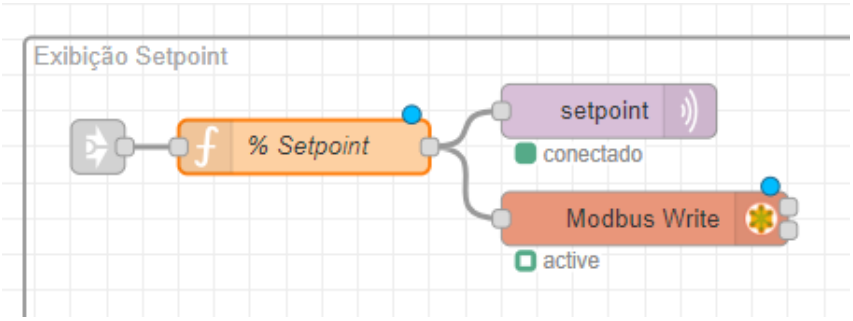
Fonte: autoria própria (2024).

4.3 Tratamento e Exibição dos Dados da Simulação

Tanto o valor de *Setpoint* quanto o valor do Nível são exibidos em *arrays* de dados na saída do nó *Modbus-Flex-Getter*. Entretanto, os valores lidos não estão formatados com vírgulas para separar as casas decimais. Por exemplo, se o dado lido é 529 ele é equivalente a 52,9%.

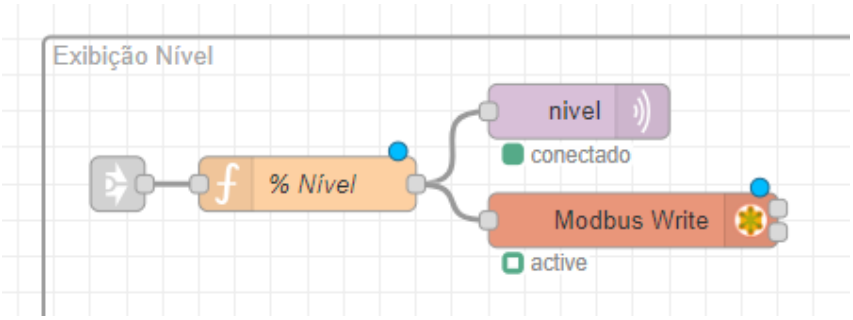
Para exibir essas informações em formato de porcentagem na *dashboard* é necessário dividir o valor da saída por 10 como pode ser observado no código do nó função "% Nível" (Figura 15). O valor tratado é escrito no *display* digital do Nível (Figura 14) e o mesmo acontece para o *Setpoint* (Figura 13).

Figura 13 – Fluxo para exibição do valor do setpoint na simulação.



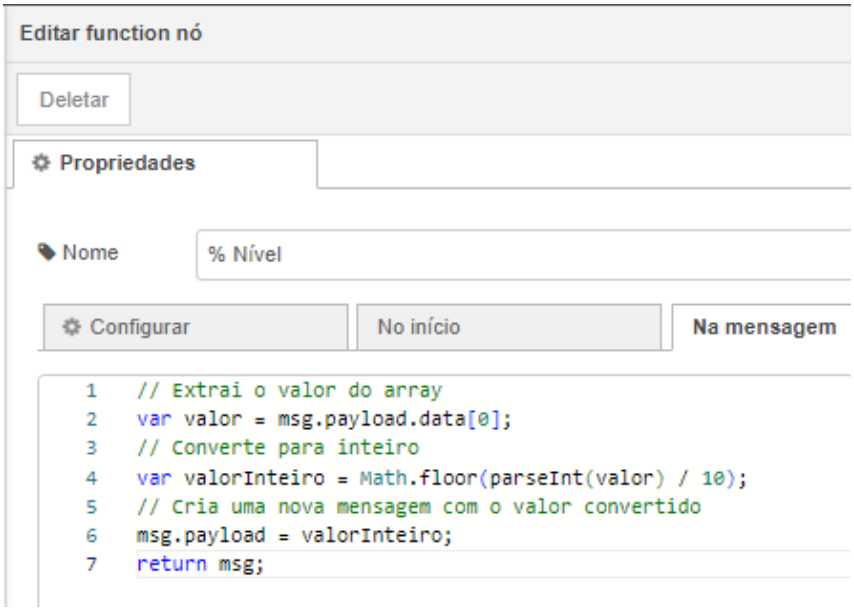
Fonte: autoria própria (2024).

Figura 14 – Fluxo para exibição do valor do nível na simulação.



Fonte: autoria própria (2024).

Figura 15 – Função para converter o valor do nível em porcentagem.



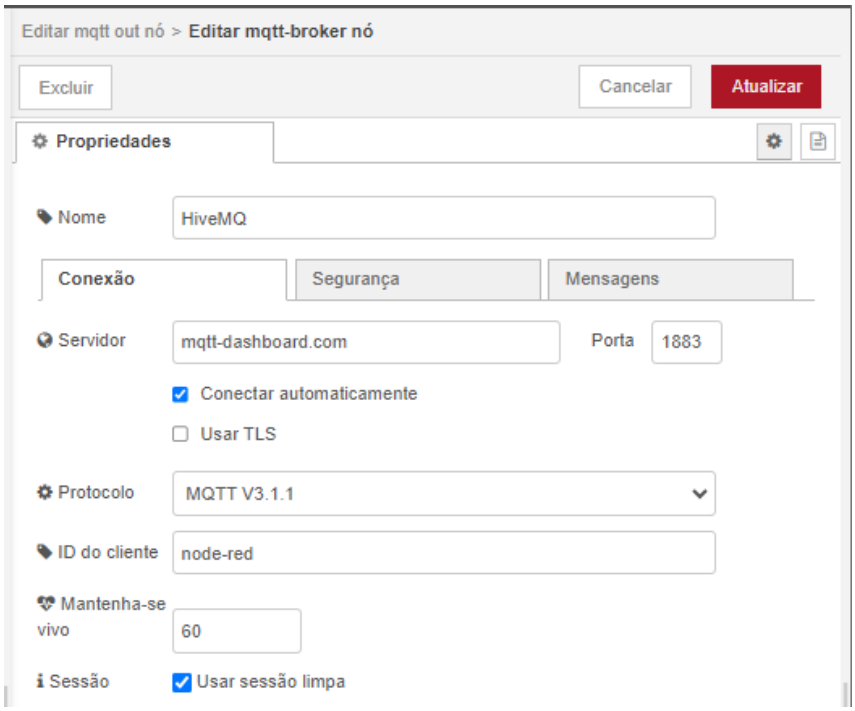
Fonte: autoria própria (2024).

Esses valores são enviados via protocolo MQTT para outro fluxo *Node-RED* em uma máquina virtual, demonstrando assim a possibilidade de visualização dessas informações remotamente e em outra máquina. No nó *MQTT-out*, é necessário selecionar o *broker* a ser utilizado, o nome do tópico de publicação e a Qualidade de Serviço (Figura 16). Na configuração do *broker*, é necessário definir a URL, a porta de acesso e o ID do cliente para identificação (Figura 17). Existem também opções extras de segurança, como a definição de um nome de usuário e senha.

Figura 16 – Nó MQTT-out.

Fonte: autoria própria (2024).

Figura 17 – Configuração do broker MQTT.

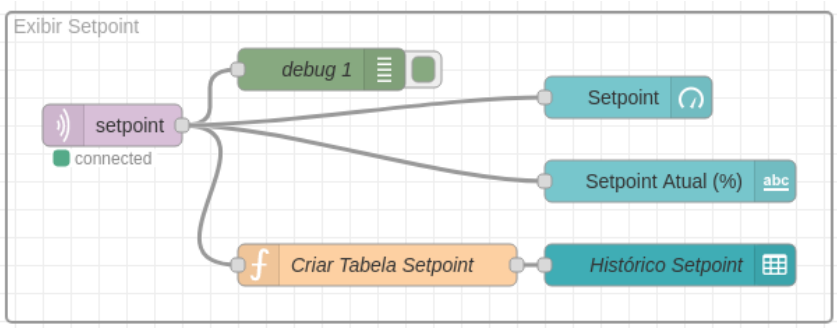


Fonte: autoria própria (2024).

No fluxo *Node-RED* da máquina virtual um nó MQTT-in que está inscrito nos tópicos *Setpoint* e Nível recebe os dados publicados. Com o uso dos nós da biblioteca

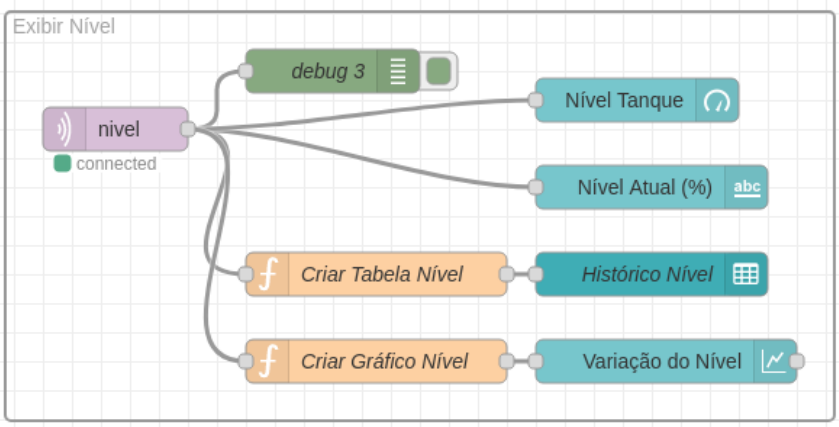
node-red-dashboard uma *dashboard* foi criada para a visualização desses dados (Figuras 18 e 19).

Figura 18 – Fluxo para exibição do valor do valor do *Setpoint* na *dashboard*.



Fonte: autoria própria (2024).

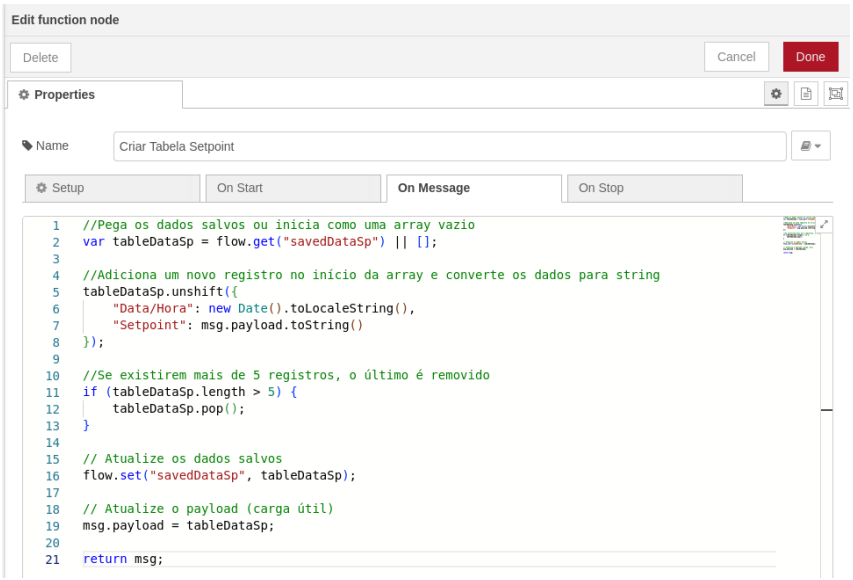
Figura 19 – Fluxo para exibição do valor do valor do Nível na *dashboard*.



Fonte: autoria própria (2024).

Para o *Setpoint*, é exibido um gráfico e um campo de texto com o valor atual em porcentagem, além de uma tabela contendo os cinco valores mais recentes, conforme a frequência de injeção do *payload*. O nó de função, ilustrado na Figura 20, permite a remoção dos valores antigos da exibição. O mesmo procedimento é realizado para o Nível, com o acréscimo de um gráfico de linha que demonstra a variação do Nível ao longo do tempo.

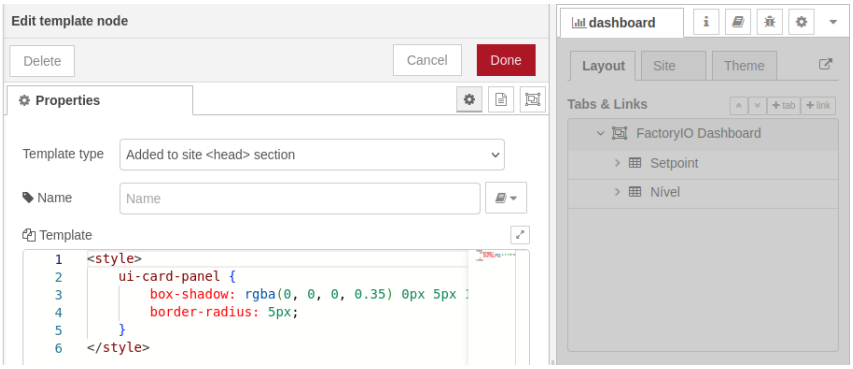
Figura 20 – Nó função para exibição do histórico de valores.



Fonte: autoria própria (2024).

O *Node-RED* permite customização do *Front-End* com o nó *Template* que pode conter quaisquer diretivas HTML e Angular válidas. Ele pode ser utilizado para criar uma interface de usuário dinâmica, no projeto ele é utilizado para melhor separar as informações acerca do Nível e do *Setpoint* (Figura 21).

Figura 21 – Nó template para customização da interface.



Fonte: autoria própria (2024).

4.4 Controle da Simulação

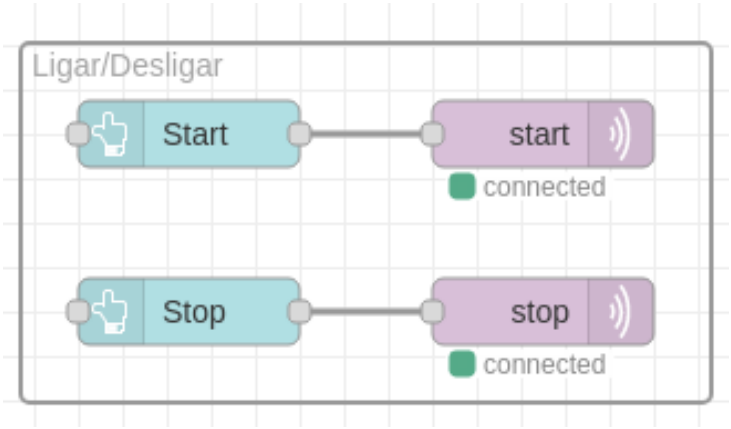
Para ligar e desligar a válvula de enchimento na simulação foi utilizado o nó *Modbus-Write* que permite acessar quatro tipos de código de função.

- FC 5: Forçar uma única bobina;

- FC 6: Predefinir um único registrador;
- FC 15: Forçar múltiplas bobinas;
- FC 16: Predefinir múltiplos registradores.

Na *dashboard* do *Node-RED* o usuário tem acesso a um botão de *start* e um botão de *stop* que são responsáveis por ligar e desligar a válvula de enchimento. Como pode ser visto na Figura 22, o valor booleano dos botões é transmitido utilizando protocolo MQTT para o fluxo *Node-RED* inicial e a partir de lá é utilizado para controlar o comportamento da simulação.

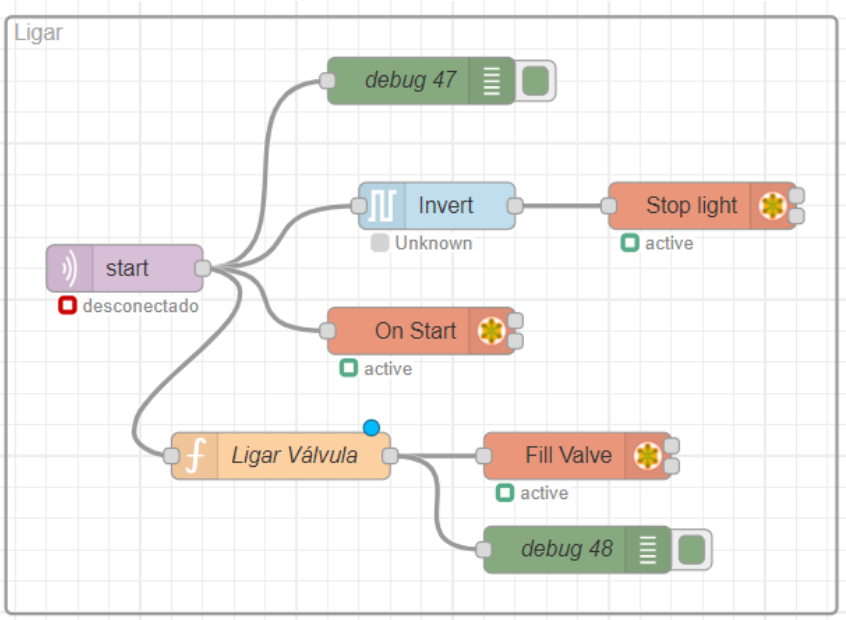
Figura 22 – Fluxo para envio da informação.



Fonte: autoria própria (2024).

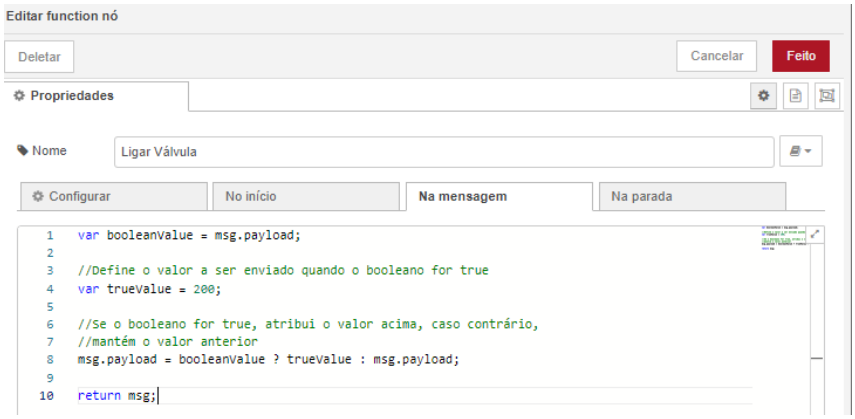
Se o botão *start* for pressionado, o valor booleano *true* é enviado, a luz do botão é acesa na simulação e a válvula de enchimento é forçada com um valor predefinido no nó função "Ligar Válvula"(Figura 24). O valor booleano *true* é invertido e enviado para a luz do botão *stop* desligando-a (Figura 23).

Figura 23 – Fluxo para ligar a válvula de enchimento.



Fonte: autoria própria (2024).

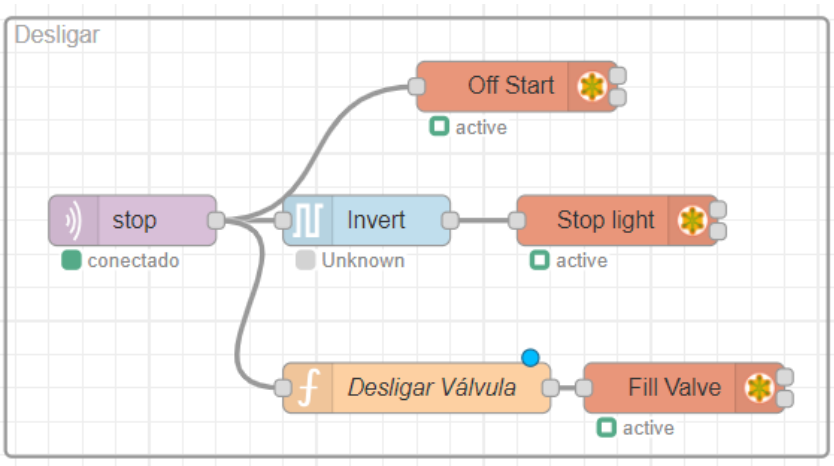
Figura 24 – Nó função para ligar a válvula de enchimento.



Fonte: autoria própria (2024).

Caso o botão *stop* seja pressionado, o valor booleano *false* é invertido, enviado para a luz do botão que é acesa na simulação e a válvula de enchimento é forçada com o valor 0 predefinido no nó função "Desligar Válvula". O valor booleano *false* é enviado para a luz do botão *start* desligando-a (Figura 25).

Figura 25 – Fluxo para desligar a válvula de enchimento.



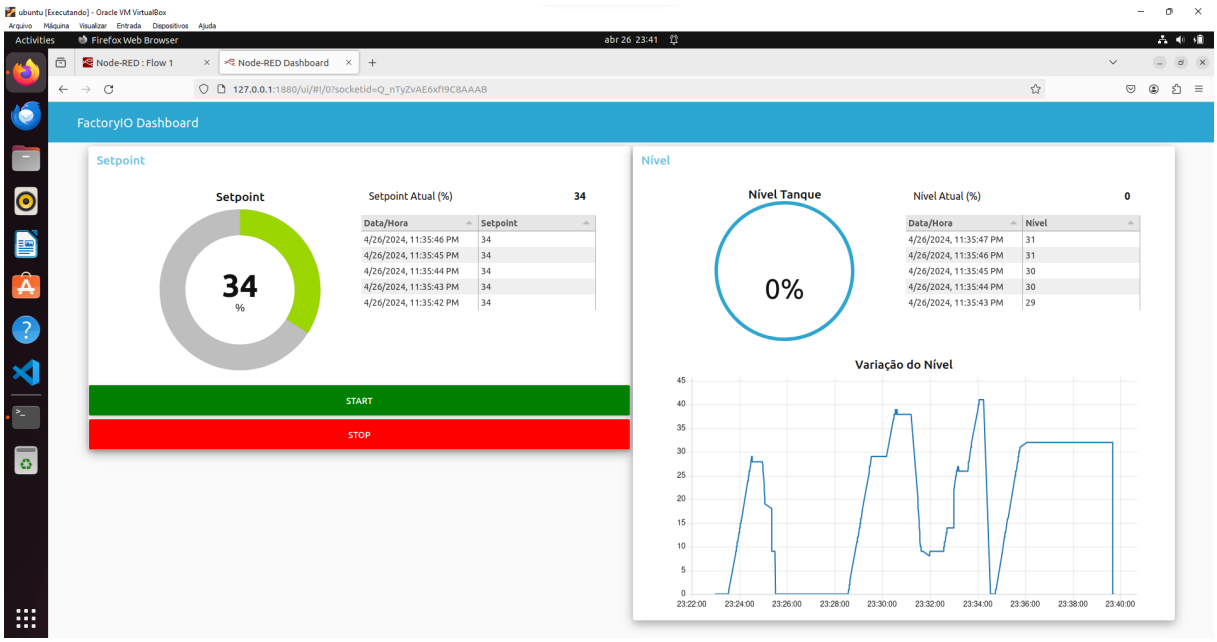
Fonte: autoria própria (2024).

5 Validação

Ao fim do projeto foi possível alcançar os objetivos específicos definidos no início deste trabalho. Uma conexão bidirecional entre o *Node-RED* e um ambiente simulado foi realizada por meio de uma *dashboard* que permite ao usuário interagir remotamente com um sistema "físico".

Na Figura 26 é possível visualizar a *dashboard* implementada. No lado esquerdo tem-se os dados relativos ao *Setpoint*, o valor atual e os valores históricos com data e hora e também os botões de *start* e *stop*. No lado direito tem-se uma animação com o Nível atual do tanque, os valores históricos e um gráfico que representa a variação do nível no tempo.

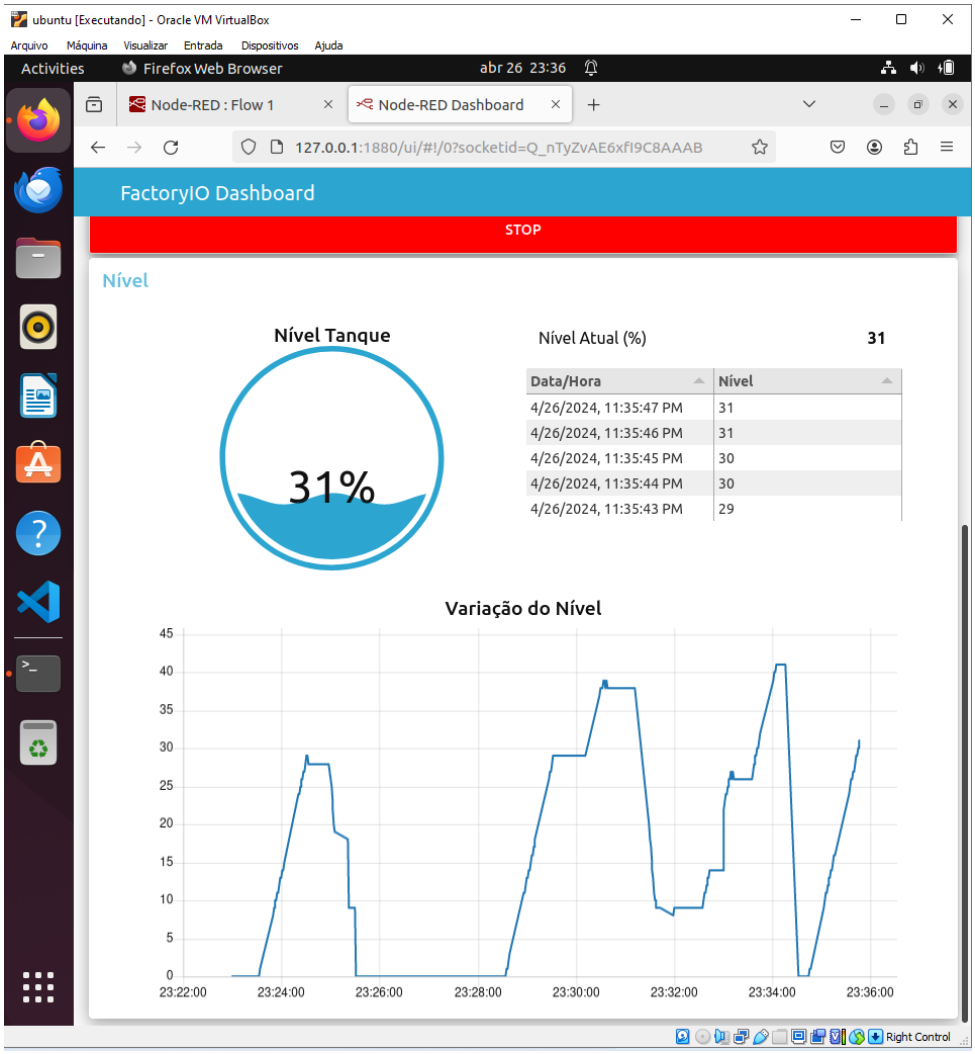
Figura 26 – Dashboard desenvolvida.



Fonte: autoria própria (2024).

Os dados são atualizados com uma frequência de um segundo que foi a frequência escolhida para a repetição do nó de injeção. Todas as informações são atualizadas em tempo real na simulação e na *dashboard*.

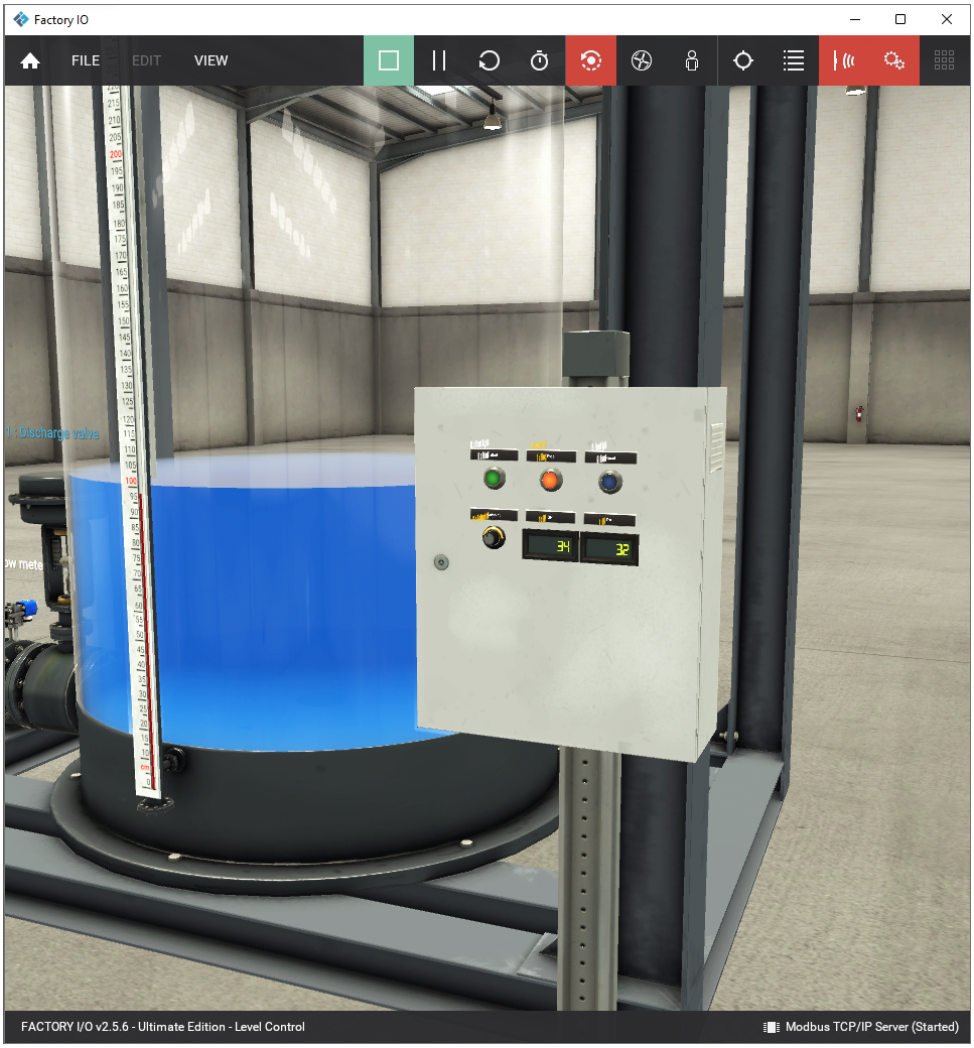
Figura 27 – Visualização na *dashboard*.



Fonte: autoria própria (2024).

Na Figura 27, o tanque estava sendo enchido, enquanto na Figura 28, o botão *stop* havia sido pressionado, interrompendo o enchimento. O usuário pode observar no *Factory I/O* a luz vermelha acesa, sinalizando a interrupção do processo, e visualizar nos *displays* digitais os valores de *Setpoint* e Nível.

Figura 28 – Visualização na simulação.



Fonte: autoria própria (2024).

A Tabela 7 abaixo descreve os casos de teste utilizados para a validação desse projeto.

Tabela 7 – Tabela de Casos de Teste

ID	**Título**	**Pré-condição**	**Critérios de Aceitação**	**Rastreabilidade**
CT-01	Ações de controle	O sistema está em funcionamento.	Ao acionar uma ação de controle remotamente, o sistema executa a ação desejada	RF-U1, RF-S3
CT-02	Acesso aos dados	O sistema está em funcionamento	Os dados obtidos são atualizados em tempo real na interface do usuário	RF-U2, RF-S2
CT-03	Teste de interface intuitiva	O sistema está em funcionamento	O usuário compreende a disposição das informações na interface	RNF-U1, RNF-S1
CT-04	Transmissão confiável de dados	O sistema está em funcionamento.	Os dados coletados são transmitidos de forma confiável entre as diferentes partes do sistema	RF-S1
CT-05	Processamento de dados	O sistema está em funcionamento	Os dados coletados são processados corretamente	RF-S2
CT-06	Tempo de resposta	O sistema está em funcionamento	O tempo de resposta do sistema para solicitações do usuário é rápido	RNF-S2

Fonte: autoria própria (2024).

Os testes de aceitação foram realizados comparando os valores exibidos na *dashboard* e na simulação com o valor obtido pelo nó *Modbus Response* e também com os valores do sistema de *tags* do *Factory I/O*. *Tags* são usadas para vincular valores de atuadores e sensores a um controlador. No entanto, ao fixa-las na simulação o usuário pode visualizar os valores atualizarem.

6 Considerações finais

A partir do desenvolvimento desse trabalho, foi possível estabelecer um arcabouço para integração do *Node-RED* com ambientes simulados industriais. No contexto de IIoT esse arcabouço se torna interessante por demonstrar uma solução na qual o usuário da interface desenvolvida pode visualizar remotamente diferentes informações acerca de uma planta industrial e implementar com sucesso ações de controle em tempo real.

Algumas competências necessárias para a realização do projeto compreendem lógica de programação, conhecimento acerca do funcionamento de protocolos de comunicação e compreensão sobre aplicações industriais de controle.

Por ser uma ferramenta muito ampla, o *Node-RED* permite a conexão entre diferentes dispositivos de *hardware* com os mais diversos serviços. Em trabalhos futuros o arcabouço desenvolvido pode ser utilizado para que o *Node-RED* opere juntamente de dispositivos de controle industrial como CLPs. Outra sugestão é implementar a integração utilizando o protocolo OPC-UA que vem se tornando cada vez mais aplicado.

Referências

FACTORYIO. 2024. <https://factoryio.com/>. Acesso em: 27/04/2024. Citado na página 16.

Modbus. 2024. <https://modbus.org/>. Acesso em: 27/04/2024. Citado 3 vezes nas páginas 12, 18 e 19.

MQTT. 2024. <https://mqtt.org/>. Acesso em: 27/04/2024. Citado 3 vezes nas páginas 12, 19 e 20.

MUSUNGATE, B. N.; TUNCAY, E. A simple node-red implementation for digital twins in the area of manufacturing. *Trends in Computer Science and Information Technology*, v. 8, n. 2, p. 50–54, August 2023. Citado 2 vezes nas páginas 12 e 16.

Node-RED. 2024. <https://nodered.org/>. Acesso em: 27/04/2024. Citado na página 15.

PIRES, F. et al. Digital twin experiments focusing virtualisation, connectivity and real-time monitoring. In: *2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS)*. [S.l.: s.n.], 2020. p. 309–314. Citado na página 20.

PIRES, F. et al. Digital twin based what-if simulation for energy management. In: *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*. [S.l.: s.n.], 2021. p. 309–314. Citado 2 vezes nas páginas 12 e 14.

SCHROEDER, G. N. et al. A methodology for digital twin modeling and deployment for industry 4.0. *Proceedings of the IEEE*, v. 109, n. 4, p. 556–567, April 2021. Citado na página 21.

STEINMETZ, C. et al. Digital twins modeling and simulation with node-red and carla. *IFAC-PapersOnLine*, v. 55, n. 19, p. 97–102, 2022. ISSN 2405-8963. Citado 2 vezes nas páginas 16 e 20.