UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Monitoramento Remoto e Diagnósticos Aplicados a Processos Industriais, Dentro do Conceito da Indústria 4.0

HUMBERTO JOSÉ GONZAGA

Itajubá, agosto de 2023

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

HUMBERTO JOSÉ GONZAGA

Monitoramento Remoto e Diagnósticos Aplicados a Processos Industriais na Indústria 4.0

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica como parte dos requisitos para obtenção do Título de Mestre em Ciências em Engenharia Elétrica.

Área de concentração: Automação e Sistemas Elétricos Industriais.

Orientador: Dr. Prof. Tales Cleber Pimenta.
Coorientador: Dr. Prof. Alexandre Baratella Lugli

Agosto de 2023
Itajubá - MG

AGRADECIMENTOS

Gostaria de agradecer em especial à minha esposa Maria Dara por sempre me incentivar e apoiar mesmo quando precisei estar ausente em diversos momentos, ao meu filho José Paulo que mesmo ainda pequeno já me serve de inspiração para buscar cada dia mais evoluir, ao ex-aluno do curso técnico Matheus Bruzamolin. Agradeço também aos meus ex-professores e sempre mestres/orientadores Alexandre Baratella e Wanderson Saldanha pela ajuda técnica e orientação no desenvolvimento deste trabalho.

Ao meu orientador, Prof. Dr. Tales Cleber Pimenta, pela paciência, apoio, incentivo e atenção todo o desenvolvimento deste trabalho, além do auxílio e dedicação durante o cumprimento das disciplinas referentes ao mestrado, bem como nas informações e conselhos que auxiliaram na elaboração do modelo final e seus estudos apresentados.

Agradeço imensamente a Deus que, em primeiro lugar, me deu o dom da vida e as capacidades necessárias para execução das atividades aqui apresentadas. Além de me guiar durante toda minha trajetória acadêmica e profissional, o que me permitiu enxergar o equilíbrio entre as necessidades pessoais e profissionais, dessa forma perceber o quanto o esforço é necessário para que esses dons apresentados possam ser usados em favor de minha evolução e na contribuição para uma sociedade e um mundo melhor.

A TODOS, A MINHA ETERNA GRATIDÃO.

DEDICATÓRIA

Dedico carinhosamente este trabalho à minha esposa, pela paciência, persistência, coragem e inspiração que ela me passou para enfrentar os grandes desafios deste trabalho. Obrigado!

RESUMO

No contexto da Indústria 4.0, o monitoramento remoto e diagnósticos aplicados a processos industriais é fundamental. Em plantas industriais em que as tecnologias são defasadas, esse processo pode ser realizado com a utilização de protocolos e ferramentas, tais como o MQTT, Modbus e *Node-RED*. Este trabalho aborda conceitos da Indústria 4.0 aplicados ao monitoramento remoto e diagnósticos, com o objetivo de criar uma interface que permita a comunicação com dispositivos de campo, que operem em protocolos como Modbus RTU, e seu usuário de forma remota. O trabalho proposto tem como objetivo criar uma interface que permita o acesso remoto aos dispositivos de campo em plantas industriais já instaladas e sem a tecnologia disponível. No contexto da Indústria 4.0, o monitoramento remoto e diagnósticos aplicados a esses processos são fundamentais para melhorias contínuas, tomadas de decisão, redução de custos, otimização da manutenção, melhoria da qualidade e outros fatores referentes a eficiência e segurança do processo produtivo. Para solucionar os problemas referentes a falta de tecnologia voltada para Indústria 4.0, foram realizados estudos para o desenvolvimento de uma interface capaz de realizar a conexão entre os dispositivos de campo desatualizados, que operem em protocolos como o Modbus RTU, e o acesso remoto. Foi feito um estudo do principal protocolo utilizado no chão de fábrica, que não possuí acesso direto as tecnologias para comunicação remota. A partir desses dados foram pesquisadas tecnologias e dispositivos capazes de comunicar com esse protocolo, realizar a troca de dados e possibilitar seu acesso de forma remota. Assim foi estudada uma ferramenta capaz de comunicar com essa interface, baseada no padrão TCP/IP, para fazer o acesso e controle remoto a partir de um computador, de qualquer lugar, conectado à rede de *internet*. Sendo assim, foi desenvolvida uma solução que realiza a troca de dados entre um dispositivo de campo e uma plataforma *online* através do protocolo MQTT, que opera sobre a arquitetura TCP/IP. Os resultados obtidos demonstram a possibilidade de adequação dos processos já existentes para o conceito da Indústria 4.0. Esse sistema permite o acompanhamento da eficiência e desempenho das operações, o que permite reduzir o tempo de resposta a eventos e falhas inesperadas no processo. Este estudo contribuí para o avanço do conhecimento nas aplicações relacionadas a adequação das plantas industriais dentro do conceito da Indústria 4.0, fornecendo uma solução prática e validada para execução das atividades mencionadas, o que permite sua utilização em futuras pesquisas em implementações relacionadas a atualização de processos industriais.

Palavras Chaves: Diagnósticos, Indústria 4.0, Modbus, Monitoramento Remoto, MQTT.

ABSTRACT

In light of the industry 4.0, remote monitoring and diagnosis applied to industrial processes is critical. In industrial plants where technologies are lagging, this process can be performed using protocols and tools such as MQTT, Modbus and Node-RED. This essay addresses concepts of industry 4.0 applied on remote monitoring and diagnosis, with the aim of create an interface that allows the communication with all field devices which operate with Modbus RTU protocols and the user, remotely. The proposed work aims to create an interface that allows the remote access to field devices in industrial plants already installed and without the available technology. In the context of Industry 4.0, remote monitoring and diagnostics applied to these processes are critical to continuous improvement, decision making, cost reduction, optimization of maintenance, quality improvement and other factors related to efficiency and safety of the production process. To solve the problems related to the lack of technology focused on Industry 4.0, studies were carried out to develop an interface capable of making the connection between outdated field devices, which operate on protocols such as Modbus RTU, and remote access. It was made a study of the main protocol used on the shop floor, which do not have direct access to technologies for remote communication. From these data were researched technologies and devices capable of communicating with this protocol, perform the exchange of data and enable their access remotely. Thus was studied a tool capable of communicating with this interface, based on the TCP/IP standard, to make access and remote control from a computer, from anywhere, connected to the internet network. Thus, a solution was developed that performs the exchange of data between a field device and an online platform through the MQTT protocol, which operates on the TCP/IP architecture. The results show the possibility of adequacy of existing processes for the concept of Industry 4.0. This system allows monitoring the efficiency and performance of operations, which allows it to reduce the response time to unexpected events and failures in the process. This study contributes to the advancement of knowledge in applications related to the adequacy of industrial plants within the concept of Industry 4.0, providing a practical and validated solution for the execution of the mentioned activities, which allows its use in future research in implementations related to updating industrial processes.

Keywords: Diagnostics, Industry 4.0, Modbus, MQTT, Remote Monitoring.

LISTA DE FIGURAS

LISTA DE TABELAS

APÊNDICES

ANEXOS

LISTA DE SIGLAS E SÍMBOLOS

ACK - Acknowledgment

ADU - Application Data Unit

AI - Artificial Intelligence

API - Application Programming Interface

ASCII - American Standard Code for Information Interchange

BDT - Big Data Technology

BROKER - Intermediário central na comunicação entre os dispositivos conectados

CLP - Controlador Lógico Programável

CPS - Cyber-Physical Systems

CTU - Central Terminal Unit

DUP - Duplicate Delivery

ESP32 - Placa de Desenvolvimento

GPS - Sistema de Posicionamento Global

HTTP - Hyper Text Transfer Protocol

I/O - Input/Output

IAAS - Infrastructure as a Service

IBM - International Business Machines

ID - Identity

IDE - Integrated Development Environment

IEEE - Institute of Electrical and Electronics Engineers

IHM - Interface Homem-Máquina

IIOT - Industrial Internet of Things

IO-LINK - Protocolo de Comunicação Digital

IOT - Internet of Things

IP - Internet Protocol

JSON - Javascript Object Notation

LSB - Least Significant Bit

M2M - Machine to Machine

MBAP - Modbus Application Protocol

MIXDES - Mixed Design

MODBUS – Protocolo De Comunicação de Dados

MQTT - Message Queuing Telemetry Transport

NODE-RED - Ferramenta de desenvolvimento

OPC UA - Open Platform Communications Unified Architecture

OSI - Open System Interconnection

PAAS - Platform as a Service

PDU - Unit Data Packet

QOS - Quality of Service

RFID - Radio Frequency Identification

RS - Recommended Standard

RS-232 - Padrão de comunicação serial

RS-485 - Padrão de comunicação serial

RTU - Remote Terminal Unit

SAAS - Software as a Service

SOA - Service Oriented Architecture

SSL - Secure Socket Layer

TA - Tecnologia da Automação

TCP - Transport Control Protocol

TCP/IP - Pilha de protocolos para comunicação a longa, média ou pequena distância

TI - Tecnologia da Informação

TLS - Transport Layer Security

TTL - Transistor-Transistor Logic

URL - Uniform Resource Locator

WI-FI - Wireless fidelity

SUMÁRIO

1 INTRODUÇÃO

A Indústria 4.0 representa uma nova era na automação industrial, onde a digitalização e a conectividade estão transformando a forma como as fábricas operam. Uma das principais áreas de aplicação dentro desse conceito é o monitoramento remoto e diagnóstico aplicados a processos industriais, onde o uso de tecnologias avançadas permite o acompanhamento em tempo real e análise de dados de sistemas produtivos de forma remota, visando a otimização da eficiência, qualidade e sustentabilidade dos processos. [1][2]

O monitoramento remoto permite a supervisão e o controle de sistemas industriais a partir de qualquer lugar, possibilitando uma visão em tempo real do desempenho e eficiência das operações. [1] O monitoramento remoto contribui para a tomada de decisões mais rápidas e assertivas, o que reduz o tempo de resposta a eventos e falhas. Por outro lado, os diagnósticos aplicados a processos industriais visam identificar anomalias, falhas e tendências que possam comprometer a eficiência e a produtividade das operações. [2][3]

Dois dos protocolos utilizados na Indústria 4.0 são o MQTT *(Message Queuing Telemetry Transport*) e o Modbus. O MQTT é um protocolo de mensagens, projetado para comunicação entre dispositivos *Internet of Things* (IoT), que permite a transmissão eficiente de dados em tempo real. Ele utiliza um modelo de publicação/assinatura (*publish/subscribe*), em que os dispositivos publicam mensagens em tópicos específicos e outros dispositivos se inscrevem nesses tópicos para receber as informações. O Modbus é um protocolo de comunicação utilizado em sistemas de automação industrial. Ele permite a troca de informações entre dispositivos, como sensores, atuadores e controladores. O Modbus opera sobre diferentes tipos de redes, como Ethernet e RS-485, e possui uma ampla gama de funções que permitem o acesso e o controle dos dispositivos conectados. [4][5][6]

O Modbus é um dos protocolos industriais mais conhecidos e populares devido a sua simplicidade e facilidade de implementação. Foi criado pela *Modicon Industrial Automation Systems* em 1979, empresa que anos antes lançou o primeiro Controlador Lógico Programável (CLP). A *Modicon Industrial Automation Systems*, posteriormente, foi adquirida pela *Schneider Electric* que tornou o protocolo como domínio público. [5]

O MQTT utiliza um modelo de comunicação no qual os dispositivos se comunicam por meio de um intermediário conhecido como broker. Os dispositivos que desejam enviar dados publicam mensagens em tópicos específicos no *broker*, e outros dispositivos interessados em receber esses dados se inscrevem nesses tópicos. Dessa forma, o MQTT permite que os

dispositivos se comuniquem de forma assíncrona, isso garante uma alta escalabilidade e flexibilidade na troca de informações. [7]

1.1 As revoluções industriais

As revoluções industriais, foram, sem dúvidas, os movimentos que deram origem a automação industrial e às demais áreas tecnológicas, sendo um dos acontecimentos mais importantes da nossa história. Pode-se observar na Figura 2 o desenvolvimento das comunicações dentro de cada período das revoluções industriais. [1]

No início do século XVIII, na Inglaterra, foram aplicados os primeiros passos na mecanização dos sistemas de produção e, a partir daí, deu-se início as revoluções industriais. Motivada, principalmente, pelas necessidades das indústrias de se obter maiores lucros com redução de custos e produção acelerada. Isso se deu, também, devido ao crescimento populacional, trazendo maior demanda de produtos. [1]

A primeira revolução industrial teve início no século XVIII com a utilização da máquina a vapor nas indústrias têxteis. Neste momento, as primeiras indústrias passam a substituir algumas atividades de trabalhadores, que exigiam esforço físico e alta frequência de repetição, por maquinários. Com isso, fez-se possível o aumento considerável das produções. [1] [2]

A comunicação neste período era realizada através do telégrafo, inventado pelo norte-americano Samuel Morse em 1837, que teve seu nome como inspiração para o código Morse. [1][2]

Na segunda metade do século XIX (1850-1870) deu-se início a segunda revolução industrial, que teve seu término durante a Segunda Guerra Mundial (1939-1945). Para o acontecimento desta revolução, a eletricidade teve um papel fundamental, já que foi o momento que esta passou a ser utilizada para o funcionamento de motores, maquinários de pequeno porte e, também, nas residências. Devido à falta de mão de obra para a época, foram criados os primeiros eletrodomésticos para suprir essa necessidade. [1][2]

Neste período, as indústrias dos segmentos químico, elétrico, petroleiro e aço passaram por uma série de evoluções. Com isso, os produtos passaram a se tornar mais acessíveis a população, devido ao surgimento da produção em massa e, consequentemente, a redução dos preços. [2]

Paralelo a este desenvolvimento industrial, a eletrônica também dava seus passos na evolução, tendo sido marcada pela criação do primeiro computador eletromecânico, Mark I, em 1944, por um professor da Universidade de Harvard. [2]

Os primeiros circuitos integrados também foram desenvolvidos durante esta revolução, sendo um deles em 1958, pelo engenheiro eletricista Jack St. Clair Kilby. Este *chip*, assim chamado, era composto por cinco componentes em uma peça de germânio, sendo um transistor, três resistores e um capacitor, com a função de implementar um oscilador. [2]

A terceira revolução industrial, que teve início em meados do século XX, foi responsável pela modernização da indústria. [1][2]

Quando se iniciou este período, o mundo já tinha acesso a rádios, televisores e automóveis. Porém, o marco desta época foi a utilização dos transistores, componente eletrônico que fez a substituição das válvulas. Foram criados também, os primeiros CLPs, que eram capazes de receber uma lógica de programação e executá-la quantas vezes fossem necessárias. Isso possibilitou a automatização dos processos produtivos, além da realização de tarefas matemáticas e cálculo de produção de forma automatizada. [1][2][3]

Esta época foi chamada de Era da Eletrônica quando os CLPs passaram a ser utilizados para realizar o controle das máquinas industriais. A Tecnologia da Informação (TI) também foi empregada nos processos de fabricação, bem como a introdução dos computadores industriais no chão de fábrica, para utilização dos sistemas de comunicação, supervisão e controle dos processos. A modernização dos processos automatizados com a utilização dos CLPs trouxe maior precisão, menores custos de implantação, além de exigir menor manutenção, quando comparado aos sistemas utilizados na época, pneumáticos e eletromecânicos. [1][2][3][8]

A década de 1970 teve um papel fundamental para o desenvolvimento na área de comunicação, tendo um avanço e possibilitando a troca de dados entre computadores, sendo que em 1973 houve a primeira conexão intercontinental entre Estados Unidos e Noruega. Ainda neste período, foi criado o Protocolo de Controle de Transmissão e Protocolo de *Internet* (TCP/IP), sendo um grande passo para à comunicação entre computadores. [2][8]

A evolução das tecnologias e da eletrônica foram fundamentais para as revoluções industriais, inclusive para o setor de automação industrial, com dispositivos e componentes cada vez menores, menor custo de produção e maior capacidade de processamento. [2]

Com o avanço da eletrônica, ocorrido durante a terceira revolução industrial, associado as evoluções do setor de telecomunicações, foi possível iniciar a Quarta Revolução Industrial, ou Indústria 4.0. Ao se associar as tecnologias de diferentes segmentos, tem-se a possibilidade da integração entre o mundo físico e o mundo digital. [2][3][8]

A Indústria 4.0 tem como desafio a comunicação entre as diferentes tecnologias, de modo que possibilite a criação de “plantas inteligentes”, com a utilização das tecnologias de telecomunicações, TI, sensores inteligentes, entre outras. Com isso, pode-se obter uma

produção sustentável e com menor desperdício, visando qualidade, baixo custo e melhor produtividade. [9]

A partir da década de 2000, foi observado que as informações que circulavam no chão de fábrica não eram utilizadas de forma adequada. Estes dados não chegavam aos níveis gerenciais, e dessa forma, existia um conflito entre as tecnologias de automação e da informação, focadas, respectivamente no processo industrial e no gerenciamento de informações do processo. Tem-se então a necessidade de convergência entre as tecnologias para a evolução da Indústria 4.0. [2]

Na Figura 1 pode-se observar a necessidade da convergência entre a tecnologia da informação (TI) e a tecnologia de automação (TA).

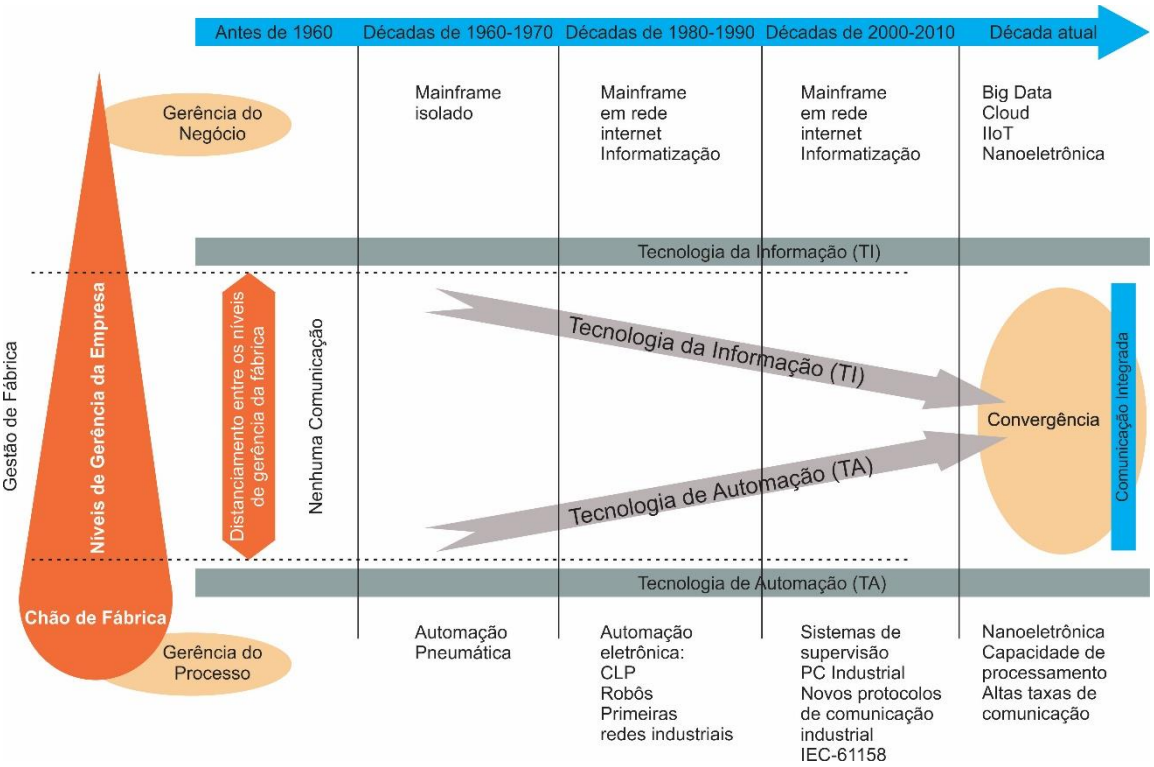

Figura 1: Convergência de tecnologias para a Indústria 4.0. [2]

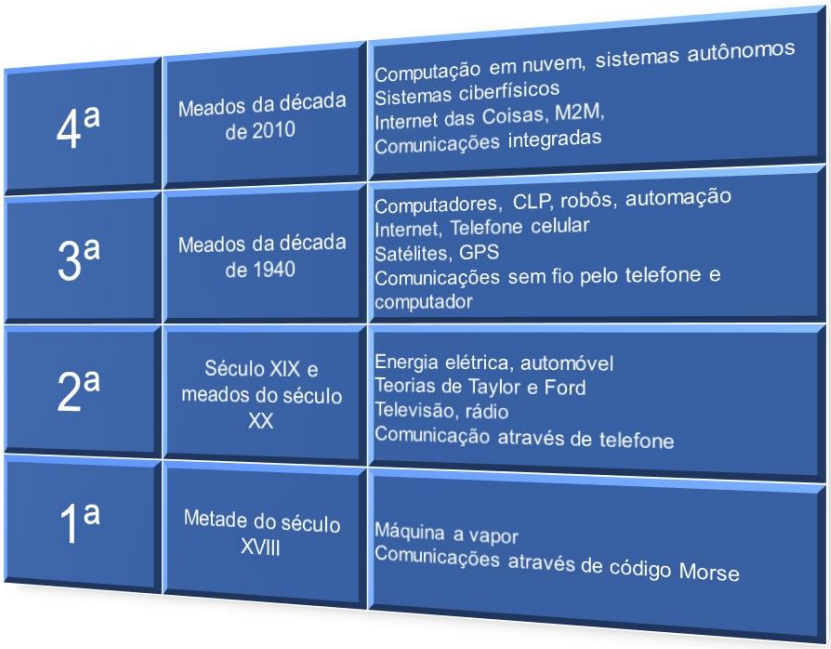

Figura 2: Revoluções industriais e o desenvolvimento das comunicações. [2]

1.2 Indústria 4.0

Na Indústria 4.0, efetivamente iniciada na segunda década do século XXI, há a convergência de níveis importantes de sensoriamento, controle e inteligência artificial ornamentados por requisitos de comunicação e intercomunicação de forma maciça. [2] Em 2013 foi proposta uma arquitetura, denominada plataforma *Industrie 4.0* (I4.0), como parte de um projeto de estratégia industrial do governo alemão. Ela representa a quarta revolução industrial impulsionada pela digitalização, automação e integração de tecnologias avançadas na indústria. [2][10]

Sendo uma transformação digital no modelo de fabricação e alavancada por tecnologias, é caracterizada pela interconexão de máquinas, equipamentos e sistemas por meio da *Internet of things* (IoT), o que permite a coleta, análise e compartilhamento de dados em tempo real. Essa conectividade possibilita a tomada de decisões de forma mais assertiva e a otimização dos processos produtivos, resultando em maior eficiência, flexibilidade e personalização na produção. [11][10]

Dentre as principais tecnologias utilizadas na Indústria 4.0, destacam-se:

- *Internet of Things*: É a base da Indústria 4.0, pois permite a conexão de objetos físicos, como sensores, atuadores e dispositivos, à *internet*, o que possibilita o monitoramento e controle remoto de processos industriais. [12][5]

Ao se tomar a IoT em uma perspectiva industrial, tem-se a *Industrial Internet of Things* (*IIoT*), que reúne diversos conceitos habituais do ambiente industrial, como supervisão, manutenção, coleta abundante de sinais de sensores, comunicação *Machine to Machine* (M2M) e tecnologias de automação. Entretanto, sua base reside na utilização de tecnologias atuais, como as máquinas inteligentes (capazes de interpretar e aprender), tratamento de grandes volumes de dados (*Big Data Technology* - BDT) e a interligação de diferentes redes de comunicação. [13][5]

- *Big Data*: Refere-se ao enorme volume de dados gerados pelos sensores, máquinas e sistemas na indústria, que podem ser coletados, armazenados e analisados para obter percepções e informações valiosas para a tomada de decisões.

  O volume de dados gerado pelas estruturas de comunicação não pode ser processado por um banco de dados convencional. O *Big Data* é uma grande estrutura construída para tratar dados estruturados e não estruturados, de forma que suporte processar as informações geradas pelas estruturas de comunicação. Normalmente essa estrutura é instalada em ambientes especializados.

  Enfim, tudo que está conectado à *internet* pode ou será uma fonte de dados para serem tratados. [2][14]
- *Cloud computing*: Permite o armazenamento e processamento de dados de forma remota, o que possibilita o acesso e análise de dados em tempo real de qualquer lugar e a qualquer momento. Associado ao grande volume de dados gerados pela Indústria 4.0, a computação em nuvem vem facilitar todo o processamento de informações e, ao mesmo tempo, disponibilizá-las para qualquer ambiente. [11][2]

  Existem três categorias de serviços: *Infrastructure as a Serv*ice (IaaS), *Platform as a Service* (PaaS) e *Software as a Service* (SaaS). Cada categoria define um conjunto de serviços disponíveis para o cliente, e essa é a chave para a nuvem: tudo é oferecido como um serviço. Isso é baseado no *Service Oriented Architecture* (SOA), em que os serviços da web foram usados para acessar as funções da aplicação. [2]
- *Artificial Intelligence* (AI): Utilizada para analisar grandes volumes de dados, identificar padrões, tomar decisões e otimizar processos de forma autônoma. [15]
- *Cyber-Physical Systems* (*CPS*): Refere-se à integração entre sistemas físicos e digitais, o que possibilita o controle e monitoramento de processos físicos em tempo real por meio de sistemas digitais. [16]

As perspectivas da Indústria 4.0 são promissoras, com a expectativa de trazer benefícios significativos para o monitoramento remoto e diagnósticos aplicados a processos industriais. Com a interconexão de dispositivos e sistemas, é possível obter dados em tempo real sobre o desempenho de máquinas e equipamentos, o que possibilita o monitoramento e diagnóstico remoto de falhas, manutenção preditiva e otimização de processos. Isso pode resultar em redução de custos, aumento da eficiência, maior qualidade dos produtos e redução do tempo de parada de máquinas e equipamentos. [9][2]

A utilização de tecnologias avançadas da Indústria 4.0 no monitoramento remoto e diagnósticos aplicados a processos industriais tem o potencial de melhorar a tomada de decisões, antecipar problemas, otimizar a manutenção e aumentar a produtividade, contribuir para a transformação digital das indústrias e sua busca por maior competitividade no mercado global. [11][10]

1.3 Monitoramento Remoto e Diagnósticos em Processos Industriais

O avanço da tecnologia e o surgimento da Indústria 4.0 têm impulsionado a transformação dos processos industriais em busca de maior eficiência, produtividade e competitividade. Nesse contexto, o monitoramento remoto e os diagnósticos desempenham um papel fundamental, o que permite o acompanhamento contínuo e a análise dos processos industriais à distância, bem como a detecção precoce de falhas e a implementação de ações corretivas imediatas. Com isso, o monitoramento remoto e diagnósticos são de importância para que se possa obter resultados referentes a:

- Otimização da eficiência operacional

O monitoramento remoto possibilita o acompanhamento em tempo real dos processos industriais, independentemente da localização física. Isso permite que os operadores tenham acesso às informações-chave sobre o desempenho dos equipamentos, variáveis de processo e indicadores de produção, o que resulta em maior eficiência operacional. Com dados atualizados em tempo real, é possível identificar problemas e gargalos, o que permite a tomada de decisões rápidas e a implementação de ações corretivas adequadas. [17][14]

- Redução de custos

A detecção precoce de falhas e anomalias por meio do monitoramento remoto permite a implementação de estratégias de manutenção preditiva. Dessa forma, é possível planejar intervenções de manutenção de forma mais eficiente, o que evita interrupções não planejadas e reduz o tempo de inatividade. Além disso, a identificação antecipada de problemas evita custos associados a reparos emergenciais, perda de produção e danos aos equipamentos. [18]

- Aumento da segurança

O monitoramento remoto contribui para a segurança nos processos industriais, o que permite a identificação de condições perigosas e riscos potenciais à distância. Os operadores podem monitorar continuamente variáveis críticas, como temperatura, pressão e fluxo, e identificar possíveis desvios que possam comprometer a segurança dos trabalhadores e a integridade dos equipamentos. A rápida detecção de situações de risco permite a implementação de medidas preventivas a fim de evitar acidentes e incidentes. [11][18]

- Otimização da manutenção

A análise contínua dos dados coletados por meio do monitoramento remoto facilita a implementação de estratégias de manutenção baseadas em condições reais. Em vez de realizar manutenção preventiva em intervalos fixos, é possível monitorar o estado dos equipamentos e realizar intervenções apenas quando necessário. Isso maximiza a vida útil dos equipamentos, reduz custos de manutenção e minimiza a interrupção da produção. [14][18]

- Melhoria da qualidade do produto

O monitoramento remoto permite o acompanhamento em tempo real de variáveis críticas nos processos industriais, o que garante que as especificações e os padrões de qualidade sejam atendidos. Qualquer desvio das metas de qualidade pode ser identificado prontamente, o que permite a tomada de medidas corretivas imediatas. Além disso, a análise dos dados coletados possibilita a identificação de causas de variações e falhas de qualidade, o que contribuí para a melhoria contínua dos processos produtivos. [2][11]

Para que os resultados citados possam ser obtidos é necessário que, dentro da Indústria 4.0, haja uma infraestrutura de comunicação eficiente para a realização do monitoramento remoto e diagnósticos em tempo real, quando necessário. São diversos os protocolos de comunicação utilizados para execução destas atividades, sendo alguns deles: *Message Queuing Telemetry Transport* (MQTT), *Open Platform Communications Unified Architecture* (OPC UA) e Modbus TCP. [14]

O protocolo MQTT tem sido adotado em uma variedade de setores, como indústria, automação residencial, cidades inteligentes e agricultura de precisão. Sua capacidade de comunicação eficiente, baixo consumo de recursos e suporte a ambientes com conectividade intermitente o tornam uma escolha popular para o monitoramento remoto, controle de processos e troca de dados em tempo real.

1.4 Metodologia

Este trabalho trata-se de uma pesquisa aplicada, necessária para que a solução apresentada fosse desenvolvida, os testes realizados e os resultados obtidos.

A motivação pala elaboração do trabalho em questão vem da necessidade de atualização de plantas industriais com tecnologias defasadas. Isso permite que essas industrias não tenham acesso as tecnologias da Indústria 4.0. Os processos industriais e laboratórios voltados para o desenvolvimento de conhecimento da área de automação, onde é necessário o estudo e acompanhamento dessa evolução, tem buscado uma solução de baixo custo para atender essas necessidades.

Sendo assim, este trabalho tem como objetivo geral pesquisar e desenvolver uma solução para realização do monitoramento remoto e diagnósticos aplicados a processos industriais, permitindo que estes se enquadrem nos conceitos da Indústria 4.0. Para isso, faz-se necessária a execução dos objetivos específicos, que são o de criar uma interface que permita o acesso e controle de dispositivos industriais de forma remota e desenvolver um sistema para validação dos conceitos abordados.

Nesta dissertação, iremos explorar o uso dos protocolos MQTT e Modbus RTU para o monitoramento remoto e diagnósticos aplicados a processos industriais, no contexto da Indústria 4.0. Serão abordados seus conceitos fundamentais, características e como eles podem ser implementados em sistemas de automação industrial. Será explorada a aplicação conjunta desses protocolos no monitoramento remoto para abordar aspectos como a coleta de dados em tempo real, a transmissão eficiente desses dados e a integração com sistemas de supervisão e controle. Existem estudos e aplicações relacionadas ao tema para utilização conjunta dos protocolos em comunicação com dispositivos. Porém, estes estudos não tem como finalidade a adequação dos processos industriais desatualizados, apenas a implementação dos conceitos da indústria 4.0 para comunicação com dispositivos e utilização dos conceitos de IoT.

Para o desenvolvimento deste trabalho foram realizadas pesquisas para possibilitar o desenvolvimento de um sistema capaz de realizar uma interface entre dispositivos de campo e o acesso remoto dos dados gerados. Considera-se para esta aplicação dispositivos com tecnologias defasadas do conceito da Indústria 4.0, tendo então como objetivo a utilização de recursos que façam uma ponte entre processos desatualizados e as necessidades atuais das indústrias.

Inicialmente foi realizado o estudo de protocolos industriais, o que permitiu considerar a utilização do Modbus RTU como base para estudo e desenvolvimento de uma interface que possibilite a troca de mensagens/dados deste protocolo de forma remota. Fez-se então o estudo

das tecnologias disponíveis para realizar o monitoramento remoto e troca de informações entre os dispositivos de campo que utilizam o protocolo Modbus RTU, o que possibilita o acesso e controle das informações de qualquer dispositivo conectado a uma rede de *internet*. Para isso, foi desenvolvida uma solução baseada no protocolo MQTT, que tem a função de realizar a troca de dados entre o dispositivo de campo e uma plataforma *online* para acesso remoto as informações, que é permitido pois o MQTT opera sobre o protocolo TCP/IP.

Uma solução foi implementada utilizando como referência, para coleta de dados, um inversor de frequência operando com protocolo Modbus RTU (RS-485), uma interface desenvolvida com a utilização de um conversor "RS-485 para TTL e TTL para RS-485" conectado a um controlador com acesso a rede de *internet* e uma ferramenta/interface para acesso *online* das informações, foi utilizado o *Node-RED* para criação da interface entre o usuário e o dispositivo. A partir dessa implementação pôde ser realizado um estudo e a criação de um sistema para realizar a troca de informações entre o dispositivo de campo e o aplicativo criado na ferramenta *Node-RED*, o que permite acesso remoto de qualquer lugar com um computador conectado à *internet*. Foi utilizado o inversor de frequência como o dispositivo de campo por se tratar de um equipamento comum nos processos industriais e possuir o protocolo definido para o desenvolvimento do trabalho.

Esta dissertação está organizada da seguinte forma:

- Conceitos teóricos: são abordados pelos capítulos um, dois e três. No capítulo um é desenvolvida uma introdução referente ao tema de desenvolvimento do trabalho, a metodologia aplicada e a carta de aceitação de artigo. Nos capítulos dois e três são discutidos os conceitos teóricos referentes as tecnologias e protocolos utilizados para implementação do estudo.
- Solução proposta e resultados: no capítulo quatro é estruturada a solução implementada e discutidos os pontos cruciais para validação dos resultados. São citados os passos realizados durante a implementação e discutidos os resultados obtidos em cada etapa.
- Conclusões e contribuições: são abordados no capítulo cinco deste trabalho.
- Trabalhos futuros: no capítulo seis são mencionadas propostas de implementações e melhorias que podem ser aplicadas para este trabalho.
- Referências bibliográficas: as referências utilizadas para elaboração e execução deste trabalho são mencionadas no capítulo sete.
- Apêndice: o artigo escrito e publicado durante o mestrado é anexado neste capítulo.

- Anexos: neste capítulo foram anexadas partes dos códigos utilizados para o desenvolvimento do trabalho. Ainda neste capítulo está anexado o *Best Paper Award*, recebido pelo trabalho publicado no MIXDES 2023.

1.5 Comunicação de Aceitação de Artigo

A Indústria 4.0 tem sido discutida como uma das principais tendências da manufatura avançada. Nesse contexto, a rede industrial *IO-Link* tem surgido como uma tecnologia-chave para a implementação de sistemas de produção mais inteligentes e eficientes.

O artigo aceito é referente ao protocolo *IO-Link*. Dentro dos conceitos da Indústria 4.0, troca de dados e digitalização, este é um protocolo de comunicação aberto utilizado para realizar a integração dispositivos de campo. Este protocolo permite, além da troca de informações entre sensores e atuadores, a troca de informações com o controlador.

Sendo assim, se trata de um estudo realizado a partir de uma tecnologia estruturada para utilização nos conceitos da indústria 4.0, que permite uma visão ampla das necessidades que devem ser supridas nos processos industriais para atender aos requisitos mínimos de troca de dados entre dispositivos.

No dia trinta de abril de 2023 foi recebida a carta de aceite do artigo de titulação "*Study and Application of the IO-Link Industrial Network in the Context of Industry 4.0"*, submetido ao *30th International Conference – Mixed Design of Integrated Circuits and Systems*, seção *Embedded Systems*.

Em 18 de julho de 2023 foi recebido o *Best Paper Award* do MIXDES 2023, conforme Anexo 1.

O artigo publicado está disponível, conforme publicação, no Apêndice 1 deste trabalho.

2 PROTOCOLO MQTT

O MQTT foi criado em meados de 1999 por Andy Stanford-Clark (IBM) e Arlen Nipper (Eurotech). Trata-se de um protocolo de mensagens baseado na arquitetura *publish/subscribe*, voltado para dispositivos restritos e redes inseguras, com baixa largura de banda e alta latência. Os princípios do design são minimizar os requerimentos de recursos de dispositivo e de largura de banda tentando garantir confiabilidade e garantia de entrega. [19]

> O MQTT está na mesma camada OSI que o *Hyper Text Transfer Protocol* (HTTP), porém a maior diferença entre eles é o tamanho do *payload*. No HTTP o *payload* é maior, o que inviabiliza o uso em conexões de baixa qualidade. Além disso o MQTT possui maior segurança, apresenta mais níveis de serviço, é menos complexo e permite uma comunicação de 1 para N se comparado ao HTTP que também é um protocolo utilizado na *Internet* das Coisas. [6]
>
> No MQTT o esquema de troca de mensagens é fundamentado no modelo *publish/subscriber*. O modelo *publish/subscriber* faz com que a comunicação entre as partes seja assíncrona, ele desacopla o emissor e o receptor da mensagem tanto no espaço quanto no tempo. Na arquitetura *publish/subscriber* do protocolo MQTT a identificação das mensagens se dá por meio de tópicos (*topics*). O tópico lembra o conceito de *Uniform Resource Locator* (URL) onde os níveis são separados por barras ("/"). Em aplicações envolvendo a *internet* das Coisas, o MQTT é um dos protocolos mais utilizados devido a sua definição de qualidade de serviço, especificações de segurança, implementação simples e garantia de utilização da banda de uso moderada. [6]
>
> As mensagens a serem transmitidas são publicadas para um endereço, chamado de tópico, assemelha-se a uma estrutura de diretórios em um sistema de arquivos. Clientes por sua vez podem se subscrever para vários tópicos, tornando-se assim capazes de receber as mensagens que outros clientes publicam neste tópico. [19]

A Figura 3 elucida o modo de funcionamento do protocolo MQTT.

Nas figuras 4 e 5 pode-se observar uma rede conectando três clientes com um *broker* central, onde os clientes B e C subscrevem para o tópico temperatura, Figura 4. Na Figura 5 pode-se observar o valor 22.5, que é publicado no tópico temperatura. O *broker* por sua vez encaminha a mensagem para os clientes subscritos naquele tópico, o que possibilita que os clientes se comuniquem. [6]

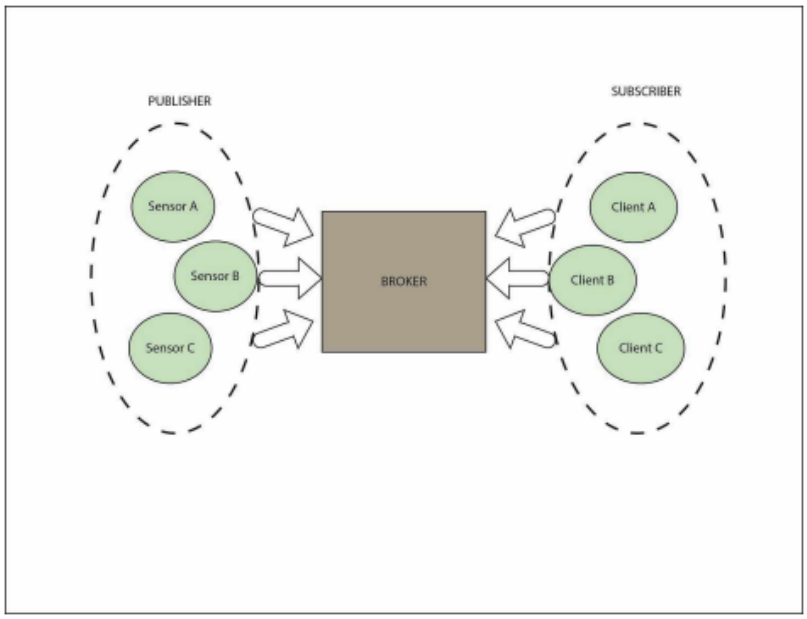

Figura 3: Arquitetura MQTT. **[6]**

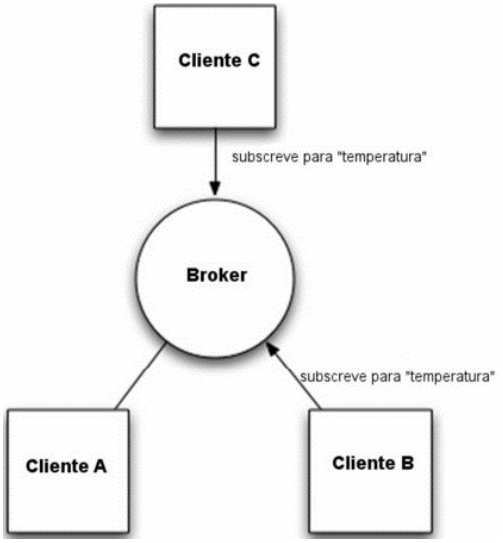

Figura 4: Subscrição de tópico ao *broker*. [19]

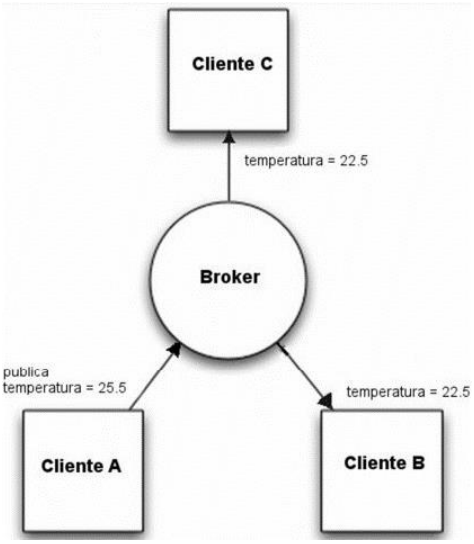

Figura 5: Publicação de mensagem no *broker*. [19]

A conexão dos clientes MQTT ao *message Broker* acontece via Protocolo de Controle de Transmissão (TCP). Na conexão se estabelece o login (usuário e senha) e o método de criptografia *Transport Layer Security*/*Secure Socket Layer* (TLS/SSL). No processo de conexão entre os elementos também se estabelece o nível de *Quality of Service* (QoS) que se deseja para o elemento conectado ao *Broker*. Este QoS define a garantia da entrega da mensagem e possui três níveis. [6]

Após a escolha do QoS e conexão com o *Broker*, o protocolo MQTT especifica que é responsabilidade do cliente garantir que o intervalo de tempo em que as mensagens demoram a ser enviadas não exceda um valor de *Keep Alive*. Caso o valor seja excedido, o cliente envia uma mensagem de controle PINGREQ. PINGREQ avisa ao *Broker* que ele ainda está vivo (*online*). Se o *Broker* não receber um PINGREQ ou qualquer mensagem de um cliente específico, o *Broker* fecha a sessão. O máximo intervalo de *Keep Alive* é 18h 12 min e 15 seg, se o valor for setado para 0, o recurso é desativado. Se o problema de conexão ainda persistir, a biblioteca iniciará o recurso chamado *Client Take Over*, fazendo com que o *Broker* feche a sessão e envie uma requisição de uma nova conexão com o cliente. [19]

2.1 Formato da Mensagem

Cada uma das mensagens no protocolo MQTT possuí um cabeçalho fixo, composto por dois bytes. A Tabela 1 mostra o formato de cabeçalho fixo.

bit	7	6	5	4	3	2	1	0
byte 1	Tipo da Mensagem				DUP Flag	Nível QoS		RETAIN
byte 2	Largura Restante							

Tabela 1: Mensagem MQTT - Cabeçalho fixo. [20]

O primeiro byte contém os campos Tipo de Mensagem e Sinalizadores (DUP, Nível de QoS e RETER). O tipo da mensagem é representado pelos bits de 7 a 4. Os quatro bits restantes dividem-se em quatro campos que servem de marcadores para indicar preferências definidas antes do envio da mensagem, são eles:

- *Duplicate Delivery* (DUP): Esse sinalizador é definido quando o cliente ou servidor tenta reenviar (duplicar) uma mensagem do tipo PUBLISH, PUBREL, SUBSCRIBE ou UNSUBSCRIBE que tenham o *Quality of Service* (QoS) maior que zero (0) e uma confirmação é necessária, *acknowledgment* (ACK). [21]
- QoS: Esse sinalizador indica o nível de garantia para a entrega de uma mensagem *PUBLISH*. Os níveis de QoS são mostrados na Tabela 2. [21]
    - QoS 0 (*at most once*): é o que possui o menor esforço, onde não são exigidas confirmações quando a mensagem é entregue. [21]
    - QoS 1 (*at least once*): neste nível existe a confirmação de entrega de uma mensagem, porém várias mensagens iguais são geradas, mas apenas uma terá o reconhecimento de chegada. [21]
    - QoS 2 (*exactly once*): garante que a mensagem seja entregue exatamente uma vez com envio de confirmação de recebimento e confirmações de recebimento. [21]
- *RETAIN*: quando um cliente envia uma mensagem *PUBLISH* ao servidor com este marcador ativado, ela deve ser retida no servidor mesmo depois de ser entregue aos assinantes. No evento de uma nova subscrição a um tópico, a última mensagem retida para este tópico deve ser enviada para o novo assinante caso este marcador esteja ativado. [19]

Valor QoS	Bit 2	Bit 1	Descrição		
0	0	0	Até uma vez	Disparar e esquecer	≤ 1
1	0	1	Ao menos uma vez	Entrega com ACK	≥ 1
2	1	0		Entrega garantida	1
3	1	1	Reservado		

Tabela 2: Níveis de QoS. [19]

O segundo byte do cabeçalho fixo é usado para representar a quantidade de bytes restantes na mensagem atual, dados no cabeçalho da variável e no *payload*.

Cabeçalho variável é um componente presente em alguns tipos de mensagem MQTT e está localizado entre o cabeçalho fixo e o *payload*. Usado principalmente nas mensagens *CONNECT*, este cabeçalho possui dois campos para nome e versão do protocolo,

respectivamente e mais uma série de marcadores que definirão algumas diretivas para a conexão entre cliente e servidor. [19][20]

O *payload* pode armazenar diferentes tipos de informação, a depender do tipo da mensagem transmitida. [19][20]

- *CONNECT*: irá conter ID do cliente.
- SUBSCRIBE: contém uma lista de nomes de tópicos aos quais o cliente pode se inscrever e o nível de QoS. [21]
- SUBACK: lista de níveis de QoS garantidos pelo servidor.

3 PROTOCOLO MODBUS RTU E TCP

A empresa Modicon introduziu no ano de 1979 o Modbus, que desde então vem sendo aperfeiçoado em suas aplicações, não apenas no mercado de automação industrial, mas também em outros setores. No momento de sua implementação, o Modbus foi concebido como o protocolo de comunicação ponto-a-ponto interno entre CLPs Modicon e painéis de programação usados para programar os controladores. A Modicon agora faz parte da AEG Schneider Automation e o protocolo Modbus permanece com constantes aprimoramentos, o que é uma característica por se tratar de um protocolo de sistema aberto, ou seja, pode ser usado livremente sem custos adicionais de *royalties*. Além de sua aplicação nos processos de automação industrial, o Modbus pode também ser aplicado em outros processos como a automação predial, por exemplo. O objetivo inicial da criação do protocolo foi o de comunicar um dispositivo mestre com outros dispositivos escravos, independentemente do tipo de rede utilizada. [22][23]

O protocolo Modbus opera com uma estrutura de mensagens que permite que os mais diversos tipos de dispositivos possam reconhecer, uma estrutura composta por bytes. [23] Em sua implementação, a Modicon utilizou o Modbus em conexões ponto-a-ponto com interfaces EIA RS-232C, sendo então um protocolo mestre-escravo que permite apenas um mestre e até 247 escravos. Apesar de operar sobre conexões seriais com padrão RS-232, também é utilizado como um protocolo da camada de aplicações de redes industriais, como o TCP/IP sobre *Ethernet.* [23][24]

O Modbus especifica um protocolo de comunicação serial para aquisições entre mestre e escravo, para isso é definido um protocolo de mensagens na camada de aplicação, de acordo com a camada 7 do modelo OSI, que proporciona comunicação cliente/servidor entre dispositivos conectados em diferentes tipos de barramentos ou redes. [5][23]

É um protocolo de requisição/resposta e oferece serviços especificados por funções. Na Figura 6 é possível perceber a existência de um único protocolo para a camada de aplicação, enquanto tem-se diferentes protocolos para as camadas mais baixas. [5][23]

O protocolo é implementado usando:

- *Transmission Control Protocol* (TCP)/*Internet Protocol* (IP) sobre Ethernet; [22]
- Transmissão serial assíncrona sobre variados meios físicos (*Recommended Standard* RS-232, RS-422, RS-485); [22]
- Modbus + (rede de alta velocidade baseada em *token passing*). [22]

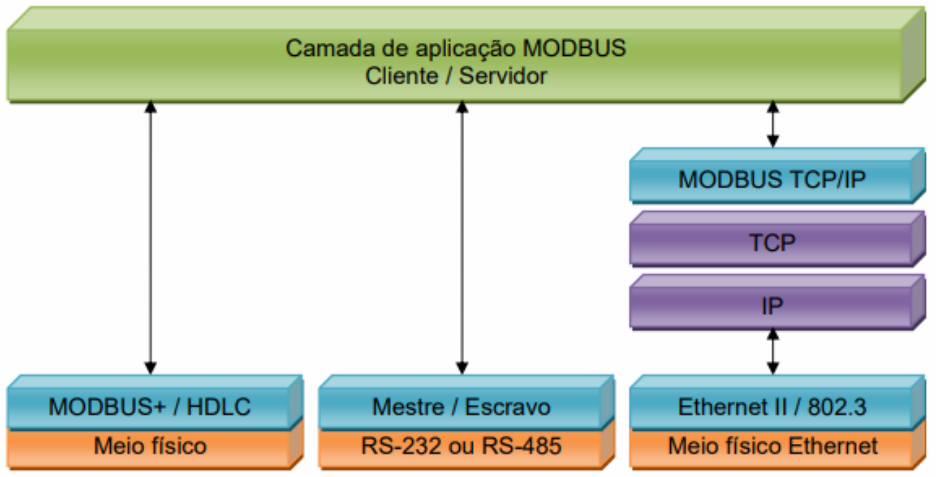

Figura 6: Pilha de comunicação. [22]

O modelo de comunicação utilizado pelo protocolo Modbus é baseado na comunicação mestre-escravo, ou seja, somente o mestre pode iniciar uma comunicação e os demais dispositivos, os escravos, responder a essa solicitação enviando os dados ou realizando alguma ação. [23] Nesse modelo todos os escravos recebem a mensagem do mestre, mas apenas o escravo com o endereço solicitado retorna à informação ou toma alguma ação de acordo com a mensagem do mestre. Os escravos são endereçados de 1 a 247, onde o endereço zero é reservado como um endereço de transmissão para todos os escravos. [22][23]

O mestre emite uma solicitação Modbus para os nós escravos em dois modos:

No modo *unicast*, o mestre se dirige a um escravo individual, como mostra a Figura 7. Depois de receber e processar a solicitação, o escravo retorna uma mensagem (uma resposta) para o mestre. [22][23]

Nesse modo, uma transação Modbus consiste em 2 mensagens: uma solicitação do mestre e uma resposta do escravo. [22][23]

Cada escravo deve ter um endereço exclusivo (de 1 a 247) para que possa ser endereçado independentemente de outros nós. [22][23][25]

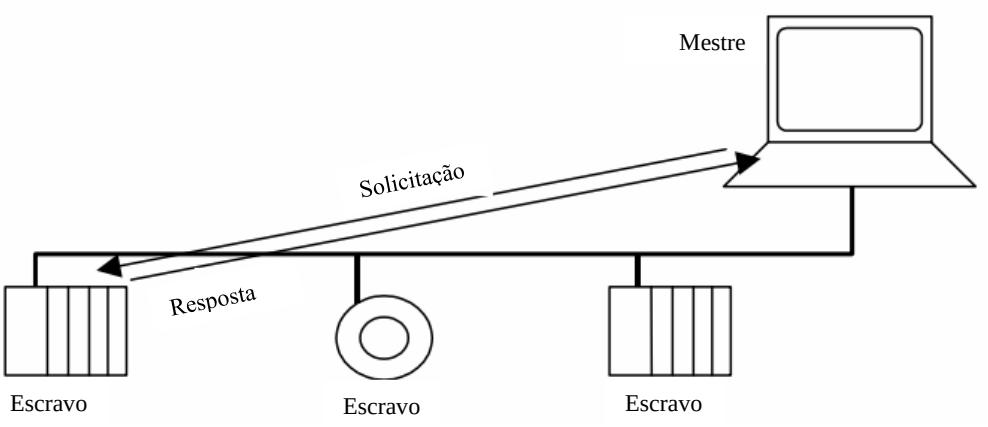

Figura 7: Troca de mensagens no modo *Unicast*. [26]

No modo *broadcast*, o mestre pode enviar uma solicitação a todos os escravos , como mostra a Figura 8.

Nenhuma resposta é retornada às solicitações de transmissão enviadas pelo mestre. As solicitações de transmissão são necessariamente comandos de escrita. Todos os dispositivos devem aceitar a transmissão para a função de gravação. [24][26]

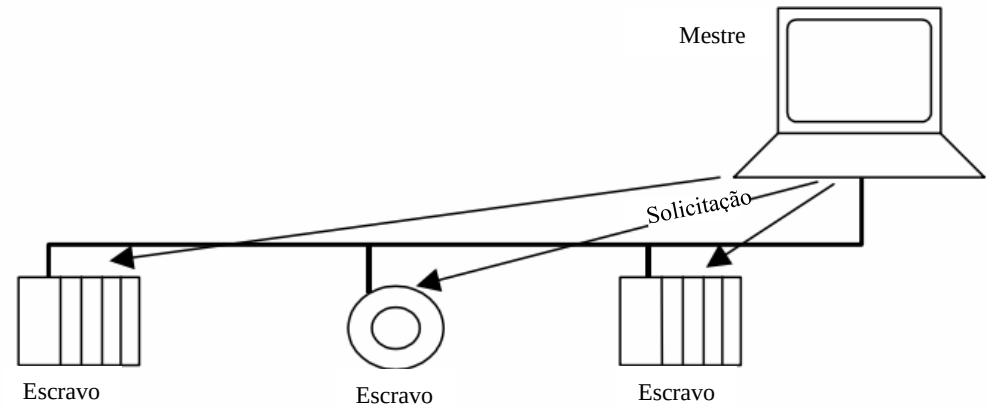

Figura 8: Troca de mensagens no modo Broadcast. [26]

A Tabela 3 mostra os endereços do Modbus, onde:

0	De 1 a 247	De 248 a 255
Endereço de Transmissão	Endereços individuais dos escravos	Reservado

Tabela 3: Endereços MODBUS. [26]

- O Endereço 0 é reservado como o endereço de transmissão. Todos os nós escravos devem reconhecer o endereço de transmissão. [25][26]
- O espaço de endereçamento Modbus compreende 256 endereços diferentes, de 1 a 247;
- Os endereços de 248 até 255 são reservados. [25][26]

3.1 Modbus ASCII e RTU

No protocolo Modbus existem dois modelos de transmissão serial, sendo o ASCII (*American Standard Code for Information Interchange*) e o RTU (*Remote Terminal Unit*). [24][26]

O termo RTU vem da indústria SCADA (*Supervisor Control and Data Acquisition*), onde o mestre, chamado de *Central Terminal Unit* (CTU), se comunica com várias RTUs em locais distantes. Essa configuração é semelhante à da implementação original do Modicon, com uma CTU se comunicando com RTUs usando modems em uma topologia estrela. O tempo das mensagens e o enquadramento podem ser afetados nos modos ASCII e RTU, mesmo que esses models não tenham relação com a topologia utilizada. Ao operar em links de comunicação serial, ambos os modos utilizam comunicações assíncronas com um caractere enviado por vez com enquadramento definido. [24] [27][28]

Na Figura 9 é possível observar o enquadramento de bits para os modelos ASCII-7 bits e RTU-8 bits.

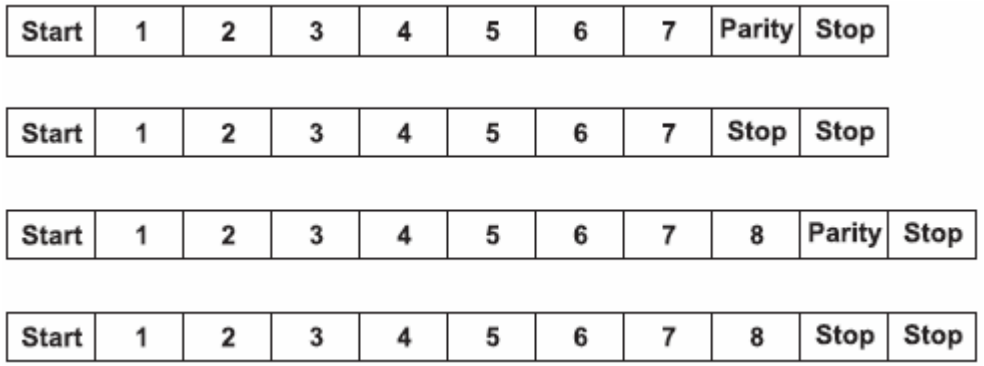

Figura 9: Enquadramento de caracteres para ASCII de 7 bits e RTU de 8 bits com ou sem paridade. [27]

A forma como um caractere é enviado usando a comunicação serial assíncrona é representada na Figura 9. Os caracteres são enviados de forma sequencial, como uma série de bits, sendo o tempo de envio conforme a taxa de transmissão. Por exemplo, para 9600 *baud rate* o tempo de bit é de 104,1μs. [23][24][27]

Um caractere de 7 bits (modo ASCII) ou 8 bits (modo RTU) é enviado entre eles, com o *Least Significant Bit* (LSB – Bit Menos Significativo) enviado primeiro. Depois do caractere vem um bit de paridade ou outro bit de parada. No modo ASCII são necessários dez bits para enviar um caractere, enquanto no modo RTU são necessários onze. Com comunicações assíncronas, os caracteres podem ser enviados de volta ou com algum atraso entre os caracteres. Uma série de caracteres formam mensagens com estruturas diferentes, dependendo se o modo ASCII ou RTU é pretendido. [5][23][24]

3.2 Modbus TCP

O Modbus.org criou o Modbus *Messaging on TCP/IP Implementation Guide V1.0b* com o propósito de permitir que o protocolo Modbus faça conexão com as redes *Ethernet*, de forma mais específica, para aplicações em redes baseadas no padrão *Ethernet*. Para o Modbus TCP foi usado um modelo de *internet* de cinco camadas, apresentado na Tabela 4, diferente do modelo usado para o Modbus Serial. [28][29]

Layer	ISO/OSI Function	Modbus Function
5,6,7	Application	Modbus Application Protocol
4	Transport	Transmission Control Protocol
3	Network	Internet Protocol
2	Data Link	IEEE 802.3
1	Physical	IEEE 802.3

Tabela 4: Modelo de cinco camadas do MODBUS TCP. [29]

Este protocolo baseia-se no padrão IEEE 802.3 do Instituto de Engenheiro Eletricistas e Eletrônicos. Não é definido o padrão para realizar a conexão física entre as estações e quais cabeamentos ou conectores utilizar, apenas é mencionado sobre como uma *Unit Data Packet* (*PDU*) Modbus é encapsulada em um protocolo de nível superior. No Modbus TCP, a PDU Modbus é encapsulada em um pacote de dados TCP/IP com o formato ilustrado na Tabela 5. [25][29]

MBAP (Modbus Application Header)	**PDU Modbus**
7 bytes	n bytes

Tabela 5: Pacote de dados TCP/IP para o PDU MODBUS. [25][29]

Onde:

- MBAP: É o cabeçalho de aplicação Modbus, que contém informações como o endereço IP do dispositivo escravo Modbus e o número de porta. [28]
- PDU Modbus: É a PDU Modbus original com o código de função e os dados. [28]

O pacote de dados Modbus TCP é então enviado pela rede usando o protocolo TCP/IP.

Uma outra consideração significativa está associada ao formato do barramento Modbus que, na verdade, se trata de um barramento IP. Em vez de ter um mestre fixado a vários escravos, o modelo cliente e servidor é usado. Os clientes podem ser IHMs ou CLPs, enquanto os servidores podem ser *racks* de entrada/saída. Como um mestre, os clientes iniciam comandos em um servidor. Como um escravo, os servidores respondem aos comandos do cliente. No entanto, a terminologia adequada com comunicações cliente/servidor é que os clientes iniciam solicitações com servidores fornecendo respostas, conforme descrito abaixo: [5]

- Uma solicitação é enviada pelo cliente para iniciar uma transação.
- Uma indicação é enviada pelo servidor para confirmar que uma solicitação foi recebida.
- Uma resposta é enviada pelo servidor para atender à solicitação do cliente.
- Uma confirmação é enviada pelo cliente para acusar o recebimento da resposta.

O que é significativo neste modelo é que vários clientes podem residir na rede IP e acessar um conjunto comum de servidores. Esta é uma mudança fundamental na forma como o protocolo Modbus funciona. Não há um único mestre controlando um conjunto definido de escravos, ou seja, diversos servidores podem ser acessados por clientes, podendo haver conflitos com clientes que façam solicitações contraditórias de um determinado servidor, como mostrado na Figura 10. [29]

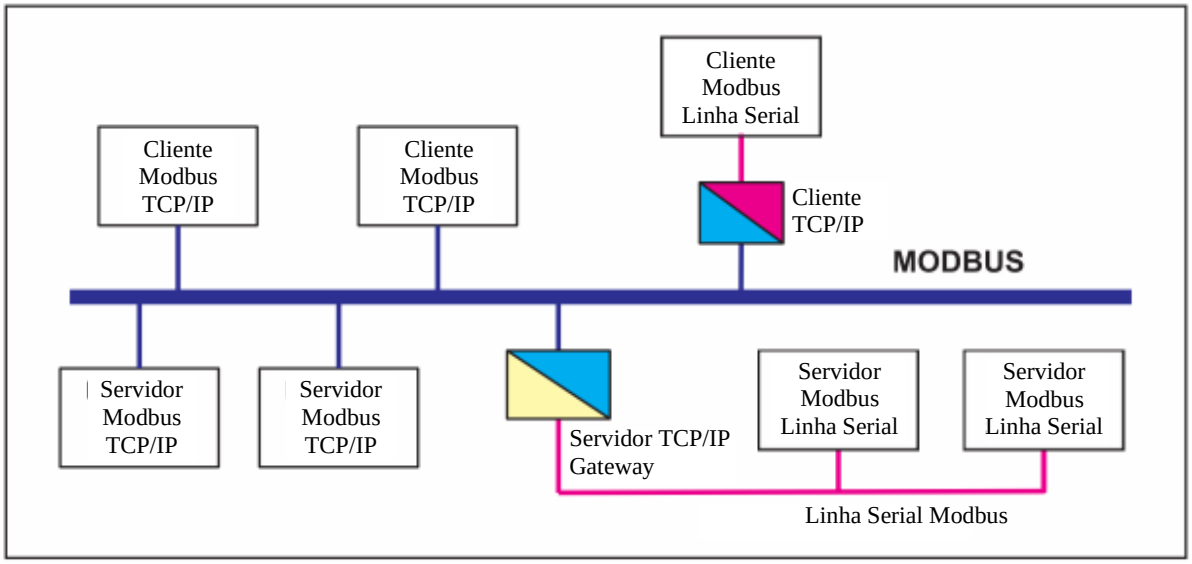

Figura 10: Modbus TCP utiliza clientes e servidores ao invés de mestre e escravos. [29]

A forma como uma nova *Application Data Unit* (ADU) Modbus TPC/IP é formada pode ser observada na Figura 11.

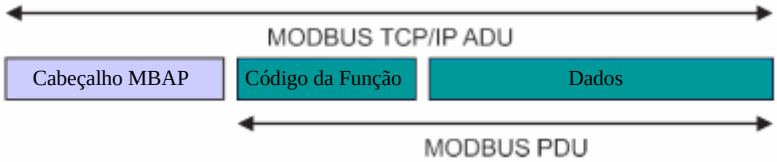

Figura 11: Cabeçalho *Modbus Application Protocol* adicionado à PDU Modbus. [28][29]

O PDU Modbus tradicional sobre o Modbus *Serial Line* e o código da função e suas definições de dados permanecem intactos, neste caso é anexado a esse PDU um cabeçalho MBAP (*Modbus Application Protocol*) distribuído, conforme mostra a Figura 12. [28][29]

Identificador
da Unidade

Figura 12: O cabeçalho MBAP possuí comprimento de sete bytes. [28][29]

O Identificador de Transação (*Transaction Identifier*) é fornecido pelo cliente e usado para acompanhar solicitações específicas, dessa forma é necessário que o servidor envie de volta o mesmo identificador com sua resposta, uma vez que o cliente tem permissão para enviar várias solicitações para um servidor sem aguardar respostas individuais. O Identificador de Protocolo (*Protocol identifier*) permite suporte para diversos protocolos, sendo que para o Modbus o valor é zero. O campo Comprimento (*Length*) identifica o comprimento de todos os campos restantes, que incluí os campos PDU Modbus. Finalmente, o Identificador de Unidade (*Unit identifier*) fornece o endereço de um escravo do *Modbus Serial Line* que deve ser acessado por meio de um *gateway*. [25][26][28]

O endereçamento é realizado utilizando endereços IP para clientes e servidores do Modbus TCP, dessa forma o endereço do *gateway* seria, então, um endereço IP. É necessária a utilização de uma porta TCP para enviar a ADU, sendo definida a porta 502 para este fim. [29][30]

4 SOLUÇÃO PROPOSTA E RESULTADOS

Os processos industriais já existentes possuem os recursos e tecnologias necessárias para realização da comunicação remota. Em processos mais antigos é comum que não haja atualizações para os dispositivos de campo, ou ainda, módulos que permitam o acesso as informações de controladores e demais instrumentos. O acesso as tecnologias da Indústria 4.0 é um desafio para essas industrias com tecnologias defasadas, o que não permite o acesso, monitoramento e diagnósticos do processo de forma remota. Sendo assim, aqueles que tem a possibilidade de adotar tecnologias atuais se destacam no mercado e evoluem em seus processos, os demais não conseguem se inserir na concorrência pela falta de informação e oportunidades de melhoria vindas dessas tecnologias.

Foi desenvolvido um sistema com o objetivo de realizar o acesso e controle de dispositivos industriais de forma remota, considerando instrumentos que não possuam essa tecnologia embutida ou disponibilidade para utilização de um módulo de expansão ou interface direta para realização desta função. Como a maioria dos dispositivos de campo (controladores) já operam com o protocolo Modbus RTU (RS232 ou RS485) [31], a solução preocupa-se em realizar a conversão dos sinais RS485 para TTL e TTL para RS485, dessa forma sendo possível a comunicação entre os controladores/instrumentos de campo e o controlador utilizado para fazer a interface para acesso remoto, ou seja, responsável por realizar a troca de informações entre os dispositivos de campo e um smartphone ou notebook de forma remota.

A atualização dos processos industriais, sem a tecnologia para acesso a Indústria 4.0, requer a substituição dos equipamentos quando não possuem um suporte para atualização ou módulos/interfaces disponíveis para este fim. A interface em estudo tem como objetivo permitir o acesso dessas plantas industriais às tecnologias da Indústria 4.0 sem a necessidade de um alto investimento ou substituição dos equipamentos que fazem parte da instalação.

O trabalho desenvolvido tem como objetivo contribuir para o avanço da Indústria 4.0, já que a interface permite que os processos industriais tenham acesso as tecnologias. Algumas das contribuições atribuídas são:

- Aplicação de Tecnologias: estudo e aplicação de tecnologias já existentes para execução da proposta e validação dos resultados.
- Melhoria na Eficiência Operacional: a Indústria 4.0 busca melhorar a eficiência operacional. A implementação de sistemas de monitoramento remoto e diagnóstico pode resultar em economia de recursos, redução de desperdícios e aumento da produtividade.

- Aumento da Confiabilidade e Disponibilidade: a solução proposta visa contribuir para o aumento da confiabilidade de processos industriais, o que permite mostrar como a detecção precoce de anomalias, através do monitoramento remoto, e manutenção preditiva podem reduzir o tempo de inatividade não planejado.
- Validação Experimental: os resultados obtidos a partir de experimentos práticos e validação da aplicação permitem demonstrar a eficácia das soluções propostas.
- Aplicabilidade Prática: as descobertas e soluções propostas podem ser aplicadas na indústria real, fornecendo diretrizes práticas para implementação.

Os fabricantes de dispositivos e instrumentos para processos industriais oferecem soluções para o problema em questão. Essas soluções envolvem a substituição dos equipamentos atuais, quando não possuem abertura para atualização ou expansão compatível com a tecnologia. Nesse contexto, a solução proposta visa a utilização de tecnologias disponíveis e acessíveis para o desenvolvimento de uma interface que permita o acesso destes dispositivos as tecnologias da Indústria 4.0.

Para realização da comunicação foi utilizado o protocolo MQTT, utilizado em aplicações que envolvem a Indústria 4.0 e IoT, possibilitando então a leitura e escrita dos parâmetros referentes aos dispositivos. Para realização do acesso remoto foi utilizado o *software Node-RED*, que é uma ferramenta de desenvolvimento baseada em fluxo para programação visual desenvolvida originalmente pela IBM (*International Business Machines Corporation*) para conectar dispositivos de *hardware*, *Application Programming Interface* (APIs) e serviços *online* como parte da *Internet* das Coisas, além de ser uma ferramenta gratuita de código aberto. Existem outros protocolos e ferramentas para comunicação e desenvolvimento da interface de acesso do usuário, foram utilizados o protocolo MQTT e o *Node-RED* devido a possibilidade de operação, facilidade de acesso e configuração e familiaridade com os recursos.

Na Figura 13 é apresentado um diagrama em blocos simplificado da aplicação. Na sequência pode-se observar a descrição e funcionalidade de cada elemento do diagrama.

Figura 13: Diagrama em blocos simplificado da aplicação realizada.

No primeiro bloco é apresentado o Inversor de Frequência. Trata-se de um dispositivo de campo muito comum nos processos industriais, tem como finalidade realizar o controle de motores. Este dispositivo de campo é conectado ao módulo Conversor Serial fisicamente com a utilização de cabos condutores.

O segundo bloco trata-se do Conversor Serial, este conversor é conectado fisicamente na Placa de Desenvolvimento ESP32 com a utilização de cabos condutores. É neste bloco que ocorre a conversão do sinal padrão RS485 para TTL e TTL para RS485. A conversão RS485 para TTL permite interpretar os parâmetros do Inversor de Frequência e mostrá-los na interface do usuário, para alterar os parâmetros configuráveis do dispositivo de campo a partir de um comando enviado pelo usuário é utiliza-se a conversão TTL para RS485. A definição das lógicas de leitura e escrita é realizada no terceiro bloco.

O terceiro bloco representa a Placa de Desenvolvimento ESP32, onde é implementado um algoritmo responsável pela comunicação e lógicas para envio e recebimento de dados entre o dispositivo de campo e a *internet*. Conectado neste bloco tem-se um *Display*, responsável por representar de forma local o estado atual da aplicação.

O último bloco do diagrama, *Node-RED* – Acesso Remoto, tem-se a interface do usuário. Neste bloco é onde ocorre a interação do usuário com o dispositivo de campo de forma remota, utiliza-se uma interface desenvolvida na ferramenta *Node-RED*. A comunicação remota entre a interface do usuário e o ESP32 é feita com a utilização do protocolo MQTT. Utiliza-se um *broker* para realizar a comunicação, um servidor responsável por receber todas as mensagens publicadas e redirecioná-las para o destino correto.

A Figura 14 representa a montagem final da aplicação, pode-se observar a esquerda da figura uma placa contendo o ESP32, o Conversor Serial e o *Display*. A direita tem-se o inversor de frequências. O ponto destacado em vermelho representa a conexão entre o Conversor Serial e os bornes de acesso à porta RS485 do Inversor de Frequência. Assim a Figura 13 pode ser representada e compreendida a partir da Figura 14.

4.1 Conversor Serial: TTL para RS485 e RS485 para TTL

Para realizar a comunicação foi necessário realizar a conversão dos sinais TTL/RS485 e RS485/TTL. Essa conversão é necessária devido aos padrões utilizados pelos dispositivos. O padrão RS485 é utilizado em dispositivos industriais, por este motivo foi escolhido um dispositivo com este padrão de comunicação para aplicação inicial. Pelo fato de estar sendo utilizada uma placa de desenvolvimento ESP32 programada pela IDE Arduino, a qual utiliza a comunicação serial TTL (*Transistor-Transistor Logic*), é necessária a utilização do conversor serial TTL/RS485 e RS485/TTL, possibilitando a comunicação entre o ESP32 e o Inversor de Frequência.

Foi utilizado o conversor serial industrial *TTL to RS485 module - C25B* para fazer a interface entre o ESP32 e o Inversor de Frequência, conforme mostra a Figura 14.

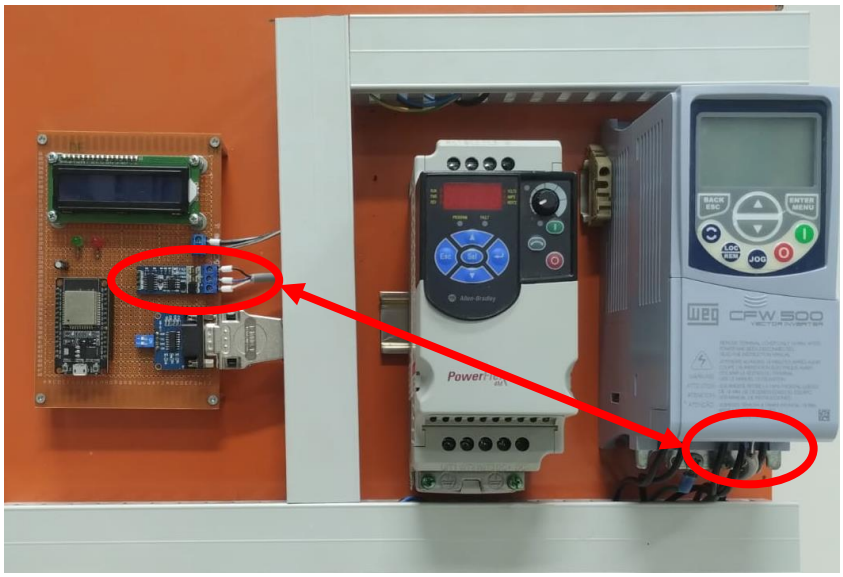

Figura 14: *TTL to RS485 module - C25B* conectado ao inversor de frequência e ao ESP32.

A utilização do conversor permite que seja realizada a comunicação entre o inversor de frequências e o ESP32.

4.2 Comunicação Serial RS485 entre o ESP32 e o Inversor de Frequência

Para que seja possível realizar a comunicação serial entre os dispositivos foi necessário realizar a parametrização do inversor de frequências e a definição das palavras de controle via código hexadecimal, conforme descrito abaixo:

- Parâmetro do Inversor de Frequência: P0682 – PALAVRA DE CONTROLE VIA SERIAL.
- Faixa de valores: 0000h a FFFFh.

- Palavra de comando do drive via interface Modbus RTU. Este parâmetro somente pode ser alterado via interface serial.
- Para que os comandos descritos neste parâmetro sejam executados é necessário que o equipamento esteja programado para ser controlado via serial. Esta programação é feita através dos parâmetros P0105 e P0220 até P0228. Abaixo estão descritas as configurações dos parâmetros descritos para que seja realizada a comunicação com o dispositivo. [29]
    - P0105 - Seleção 1ª/2ª Rampa: 3 = Serial/USB
    - P0220 - Seleção LOC/REM: 6 = Serial/USB (REM)
    - P0222 - Sel. Referência REM: Ver opções em P0221
        - P0221 - Sel. Referência LOC: 9 = Serial/USB
    - P0226 - Seleção Giro REM: Ver opções em P0223
        - P0223 - Seleção Giro LOC: 5 = Serial/USB (H)
    - P0227 - Seleção Gira/Para REM: Ver opções em P0224
        - P0224 - Seleção Gira/Para LOC: 2 = Serial/USB
    - P0228 - Seleção JOG REM: Ver opções em P0225
        - P0225 - Seleção JOG LOC: 3 = Serial/USB

Conforme descrito no parágrafo acima, para que os comandos possam ser realizados via porta serial é necessário seguir as configurações e parametrizações do inversor de frequências.

A Tabela 6 representa os bits utilizados para comandos no equipamento. [29]

Bits	15 a 8	7	6	5	4	3	2	1	0
Função	Reservado	Reset de Falhas	Parada Rápida	Utiliza Segunda Rampa	LOC/REM	JOG	Sentido de Giro	Habilita Geral	Gira/Para

Tabela 6: Palavra apresentada em bits para execução dos comandos. [28]

As descrições das funções apresentadas na Tabela 6 podem ser observadas na Tabela 7.

Bits	Valores
Bit 0 Gira/Para	0: Para motor por rampa de desaceleração. 1: Gira motor de acordo com a rampa de aceleração até atingir o valor da referência de velocidade.
Bit 1 Habilita Geral	0: Desabilita geral o drive, interrompendo a alimentação para o motor. 1: Habilita geral o drive, permitindo a operação do motor.
Bit 2 Sentido de Giro	0: Sentido de giro do motor oposto ao da referência (sentido reverso). 1: Sentido de giro do motor igual ao da referência (sentido direto).
Bit 3 JOG	0: Desabilita a função JOG. 1: Habilita a função JOG.
Bit 4 LOC/REM	0: Drive vai para o modo local. 1: Drive vai para o modo remoto.
Bit 5 Utiliza Segunda Rampa	0: Drive utiliza como rampa de aceleração e desaceleração do motor os tempos da primeira rampa, programada nos parâmetros P0100 e P0101. 1: Drive utiliza como rampa de aceleração e desaceleração do motor os tempos da segunda rampa, programada nos parâmetros P0102 e P0103.
Bit 6 Parada Rápida	0: Não executa comando de parada rápida. 1: Executa comando de parada rápida. Obs.: quando o tipo de controle (P0202) for V/F ou VVW não se recomenda a utilização desta função.
Bit 7 Reset de Falhas	0: Sem função. 1: Se em estado de falha, executa o reset do drive.
Bits 8 a 15	Reservado.

Tabela 7: Funções dos bits para o parâmetro P0682. [28]

Com a Tabela 7, informada pelo fabricante, foi necessária a conversão dos bits de comando para o padrão hexadecimal, dessa forma sendo possível realizar o controle (leitura/escrita) dos parâmetros em questão. Para isso foi montada uma tabela para conversão dos sinais binários para hexadecimal considerando os oito primeiros bits das funções, como mostra a Tabela 8.

Binário								Hexadecimal Arduino IDE
7	6	5	4	3	2	1	0	
0	0	0	1	0	0	1	0	0x12
0	0	0	0	0	0	0	1	0x01
0	0	0	0	0	1	0	0	0x04
0	0	0	1	0	0	1	1	0x13
0	0	0	1	0	1	1	1	0x17
1	0	0	1	0	0	1	0	0x92
0	0	0	1	0	1	1	0	0x16

Tabela 8: Conversão dos bits de funções - binário para hexadecimal.

Após realizadas as conversões, os códigos em hexadecimal podem ser utilizados para programação do ESP32, que foi desenvolvida a partir do *Arduino Integrated Development Environment* (Arduino IDE). Analisando os códigos hexadecimais gerados na Tabela 8 é possível acompanhar na Tabela 6 quais os bits foram habilitados, consequentemente, quais comandos foram executados.

A programação foi estruturada da seguinte forma:

- Definição das bibliotecas utilizadas;
- Definição das variáveis utilizadas como referência no *Node-RED*;
- Definição das variáveis utilizadas no código;
- Identificação de sessão para o *broker*;

- Configuração da rede para conexão do ESP32 com a *internet*;
- Configuração/definição do *broker* utilizado;
- Desenvolvimento do algoritmo para cumprimento da proposta.

Algumas partes referentes ao código desenvolvido no Arduino IDE para troca de informações com o inversor de frequências foram disponibilizados em forma de anexos neste trabalho. Os comandos anexados permitem que o motor conectado ao inversor realize as seguintes funções, comandadas via serial: controle de velocidade, controle do sentido de giro (horário e anti-horário), comando para ligar ou desligar o motor, comando para realizar o reset de falhas no inversor de frequência.

Por exemplo, para que o motor gire no sentido horário (direto) é enviado o código hexadecimal 0x17 via serial (`au16data[0] = 0x17`). Dessa forma, são habilitados os bits 0, 1, 2 e 4 da palavra enviada para o inversor, sendo eles:

- Bit 0 – Quando em 1, gira o motor de acordo com a rampa de aceleração.
- Bit 1 – Quando em 1, habilita o drive e permite a operação do motor.
- Bit 2 – Quando em 1, sentido de giro direto do motor.
- Bit 4 – Quando em 1, drive vai para o modo remoto.

Da mesma forma, para que o motor gire no sentido anti-horário (reverso), é enviado o código hexadecimal 0x13, sendo:

- Bit 0 – Quando em 1, gira o motor de acordo com a rampa de aceleração.
- Bit 1 – Quando em 1, habilita o drive e permite a operação do motor.
- Bit 2 – Quando em 0, sentido de giro reverso do motor.
- Bit 4 – Quando em 1, drive vai para o modo remoto.

No Apêndice 2 tem-se uma parte do código desenvolvido na IDE Arduino que representa os comandos abaixo:

- 0x17 – Ligar o motor no sentido direto de giro.
- 0x16 – Desliga o motor a partir de uma rampa de desaceleração.
- 0x13 – Sentido de giro reverso do motor.
- 0x92 – Comando para realizar *reset* de falhas do motor.

Para realizar a comunicação entre o ESP32 e o Inversor de Frequências foi necessário utilizar a biblioteca *ModbusRtu.h*.

Como o ESP32 possuí dois núcleos, o primeiro (*Task1code*) foi utilizado apenas para realizar a sinalização da placa, fazendo com que um led realize uma sinalização intermitente

com intervalos de um segundo. O segundo núcleo (*Task2code*) foi utilizado para realizar todos os demais comandos relacionados a comunicação e troca de dados entre os dispositivos.

No Apêndice 3 é apresentada uma parte do código desenvolvido que representa os comandos referentes as mensagens disponibilizadas no *display*, a escolha do tipo de acesso que será realizado (celular ou *notebook*) e aos comandos de controle do motor conforme escolha do usuário. Pode-se observar no "case 2" do código apresentado a parametrização para escrita de informações no escravo `0x01`.

4.3 Comunicação entre o ESP32 e o *Node-RED* via MQTT

Para possibilitar a comunicação entre o ESP32 e o *Node-RED*, onde foi criada a interface remota para monitoramento e controle do dispositivo, foi utilizado o protocolo de comunicação MQTT. O *Node-RED* se destaca como sendo uma solução para integração e processamentos dos dados gerados e transmitidos por dispositivos conectados à *internet*.

O *Node-RED* tem como uma de suas principais características sua interface gráfica que é baseada em nós (nodes), que podem ser conectados para criar fluxos de trabalho. Cada nó representa uma função ou serviço específico, como leitura de sensores, processamento de dados, interações com bancos de dados, entre outros. [34][35]

Além da facilidade de construção de fluxos, o *Node-RED* possui uma biblioteca de nós pré-configurados que permitem a integração com diversos dispositivos e serviços. Isso inclui protocolos de comunicação como MQTT e HTTP, bancos de dados como MongoDB e MySQL, serviços de nuvem como AWS e Azure, plataformas de mídia social, entre outros. [34][36]

Outro aspecto importante do *Node-RED* é a capacidade de expansão por meio do desenvolvimento de nós personalizados. Isso significa que é possível criar nós com funcionalidades específicas que não estão disponíveis na biblioteca padrão. Essa flexibilidade permite que os usuários personalizem e estendam o *Node-RED* de acordo com suas necessidades específicas, tornando-o uma plataforma altamente adaptável. [35][36]

Para o projeto em questão foi desenvolvida uma interface no *Node-RED*, conforme pode ser observada na Figura 15.

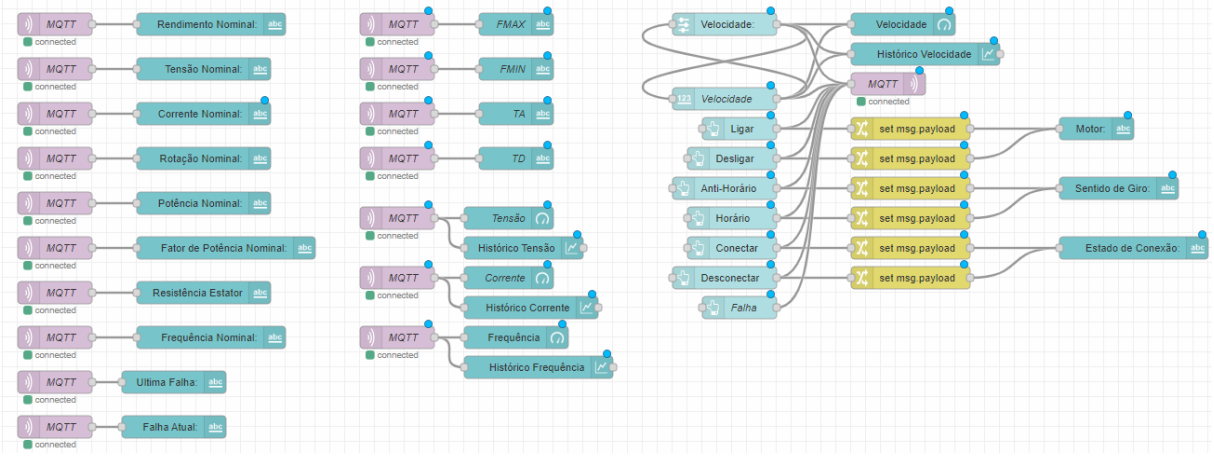

Figura 15: Tela da aplicação desenvolvida no *Node-RED*.

Conforme pode ser observado na Figura 15, cada nó possui uma função específica, abaixo estão representados alguns dos nós utilizados e suas funções.

O *download* do código desenvolvido na plataforma pode ser feito em sua forma de nós através da função *Export nodes*, conforme exibido na imagem acima, ou ainda no formato *Javascript Object Notation* (JSON). No Apêndice 4 é possível observar uma pequena parte da estrutura do código em JSON convertido em JSON.

A Figura 16 representa uma parte do código onde o *Node-RED* recebe informações via MQTT do dispositivo de controle e transfere essas informações para a interface do usuário.

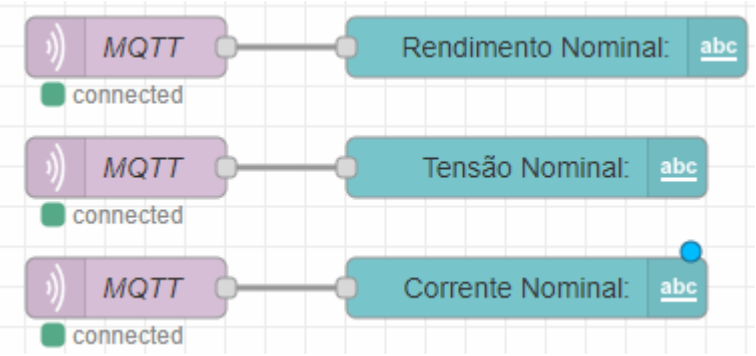

Figura 16*: Node-RED* – Nó MQTT IN recebendo informações do dispositivo.

O MQTT-IN é responsável por receber os dados do dispositivo e então apresentá-lo na interface do usuário. Cada um desses "nós" deve ser configurado de acordo com as configurações do controlador que está realizando a interface entre o *Node-RED* e o dispositivo de campo. Na Figura 17 é possível observar as telas de configuração do nó MQTT IN, que é responsável por receber as informações e do nó *TEXT*, que é responsável por mostrar a informação recebida na interface do usuário.

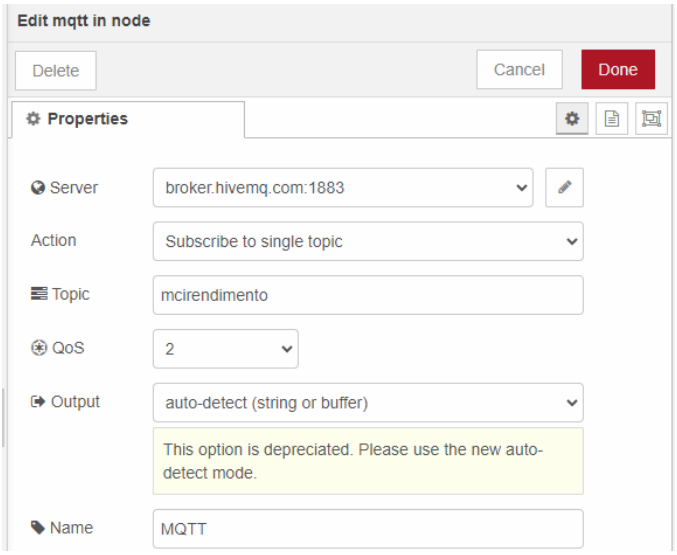

Figura 17: Tela de configuração do nó MQTT IN.

Na Figura 17 é possível observar a necessidade de realizar uma configuração correta do *broker* que será utilizado, sendo este mesmo configurado também no controlador que está fazendo a comunicação. Essa configuração no ESP32 pode ser observada através das linhas de configuração a seguir, ilustradas pela Figura 18 e pelos comandos abaixo.

```
//#### CONFIGURACAO DA REDE DE INTERNET PARA CONEXAO DO ESP32
const char* SSID = "Familia";
const char* PASSWORD = "Pazdecristo";
//#### BROKER UTILIZADO
const char* BROKER_MQTT = "broker.hivemq.com";
int BROKER_PORT = 1883;
```

Na Figura 18 é possível observar as configurações referentes ao que será mostrado na interface do usuário. Ou seja, o nó MQTT IN recebe os dados e os transfere para um padrão de visualização compreensível ao usuário final.

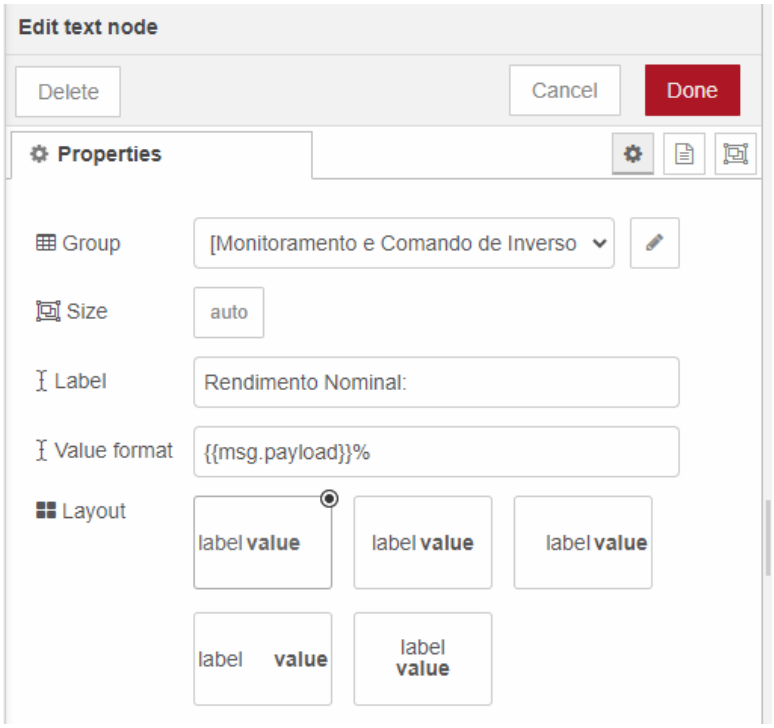

Figura 18: Tela de configuração do nó *TEXT*.

A Figura 19 representa parte da estrutura desenvolvida na ferramenta de *Node-RED*. O nó MQTT, destacado na cor roxa e com símbolo de *wi-fi* a sua direita, é chamado de nó MQTT OUT e tem como função receber os comandos gerados pelo usuário e transmitir via MQTT para o controlador. Todos os nós responsáveis pela alteração ou envio de algum parâmetro para o controlador são conectados no MQTT OUT.

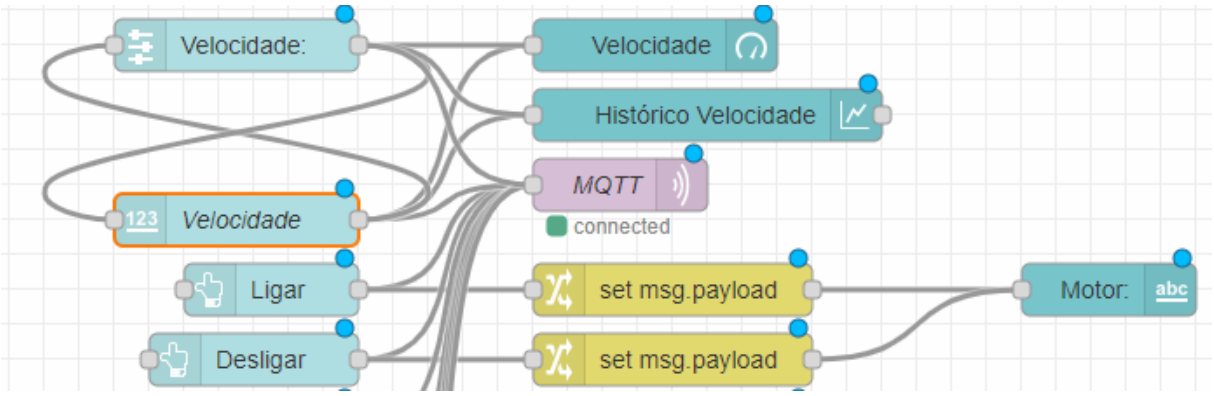

Figura 19: *Node-RED* – Nó MQTT OUT enviando informações para o dispositivo.

O MQTT OUT é responsável pelo envio da informação do *Node-RED* via MQTT para o dispositivo. Através da interface do usuário podem ser enviados comandos para o dispositivo via protocolo MQTT, esses comandos podem ser discretos (ligar ou desligar um equipamento) ou então analógicos, onde é possível realizar o controle de velocidade ou temperatura, por exemplo. Nas figuras 20 e 21 pode-se observar as telas de configuração do MQTT OUT e de

uma das funções executadas. Sem a utilização do nó MQTT OUT não é possível realizar o envio de dados da interface do usuário, criada no *Node-RED*, para o dispositivo.

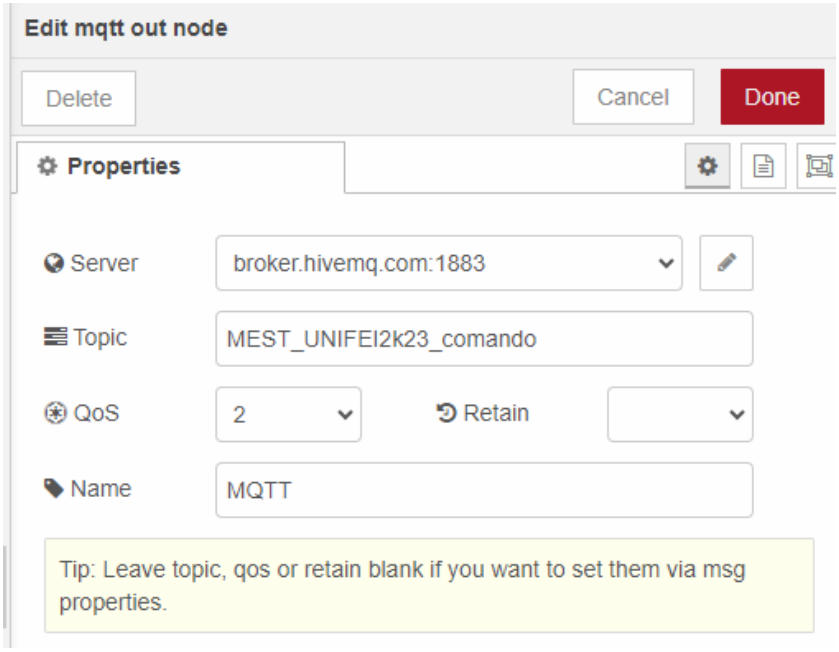

Figura 20: Tela de configuração do nó MQTT OUT.

O campo *Topic* é preenchido de acordo com a programação do ESP 32, onde são criadas as variáveis para receber e interpretar os dados enviados pelo *Node-RED* via MQTT.

Abaixo, nas linhas de comandos a seguir, é possível observar algumas das variáveis utilizadas no desenvolvimento do trabalho.

```
//#### DEFINICAO DAS VARIAVEIS QUE SERAO UTILIZADAS COMO REFERENCIA NO NODE-
RED ####
#define TOPICO_SUBSCRIBE        "MEST_UNIFEI2k23_comando"

#define PUBLISH_CONEXAO         "mciconexao"

#define PUBLISH_TENSAO          "mcitensao"
#define PUBLISH_CORRENTE        "mcicorrente"
#define PUBLISH_FREQUENCIA      "mcifrequencia"

#define PUBLISH_FREQUENCIAMAX   "mcifrequenciamax"
#define PUBLISH_FREQUENCIAMIN   "mcifrequenciamin"
#define PUBLISH_TEMPOA          "mcitempoa"
#define PUBLISH_TEMPOD          "mcitempod"
```

Na Figura 21 pode-se observar a configuração de um nó do tipo *BUTTON* (botão), este nó é responsável por enviar uma informação, ou um pulso, para o controlador, que irá interpretar este dado e então realizar uma função designada. Neste caso o objetivo é o de ligar o motor através do *Payload* “habilita”, que será reconhecido pelo ESP 32. A palavra “Ligar” está preenchendo o campo *Label*, portanto será parte da interface do usuário.

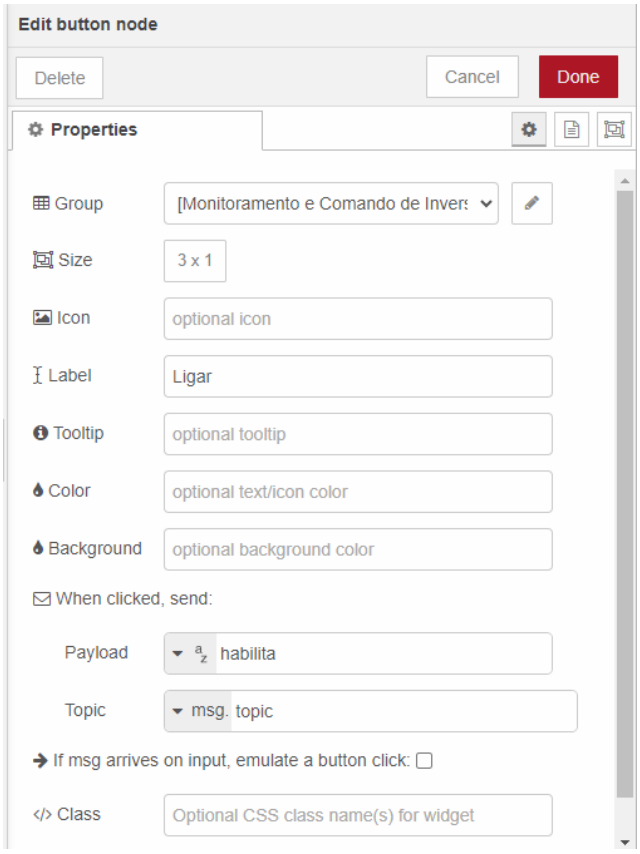

Figura 21: Tela de configuração do nó *BUTTON*.

Abaixo, nas linhas de comando a seguir, está uma parte do código desenvolvido no ESP32 que tem a função de interpretar quando é dado o comando “Liga” e então realiza uma função, que é a de ligar o motor através do inversor de frequência.

```
if(comando == "habilita")
     {
      if(aux_sentido == 2 or aux_sentido == 0) au16data[0] = 0x13;
      if(aux_sentido == 1) au16data[0] = 0x17;
      aux_comando = 1;
     }
```

O comando para habilitar/ligar o motor se dá pelo código hexadecimal “0x17”.

Na Figura 22 é possível observar a interface do usuário a partir do código desenvolvido no *Node-RED*. Essa interface pode ser acessada de forma remota, possibilitando o monitoramento, acompanhamento de operações e variações nos equipamentos, dessa forma realizar os diagnósticos e, quando necessário, realizar intervenções no equipamento ou processo de forma remota.

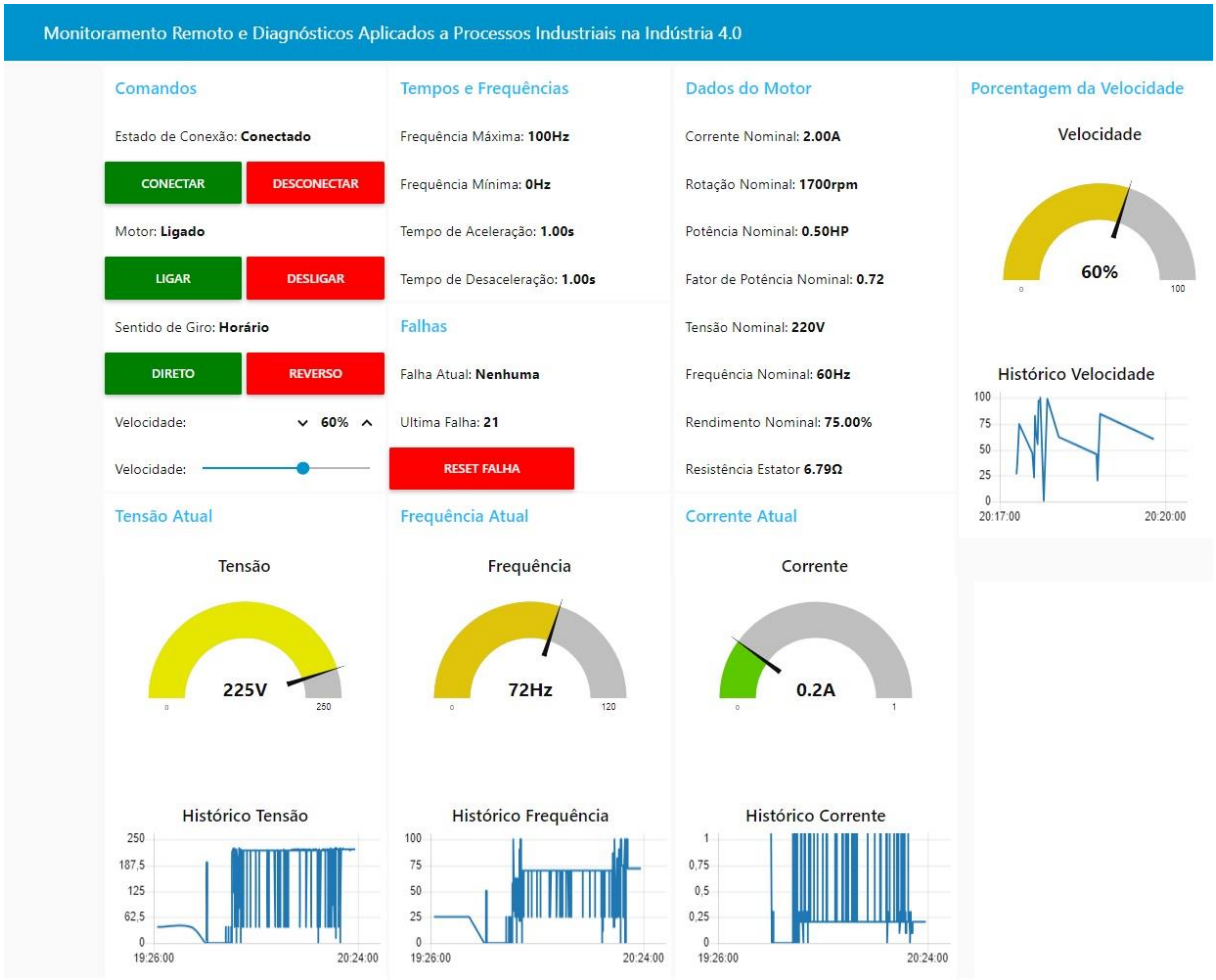

Figura 22: Interface do usuário no Node-RED.

Após todos os testes realizados e validados, pode-se acessar remotamente o dispositivo de campo. Os principais resultados obtidos foram:

- Possibilidade de acesso remoto ao dispositivo de campo;
- Utilização de tecnologia acessível para desenvolvimento da interface;
- Interface do usuário que permite leitura e configuração dos parâmetros do dispositivo de forma remota.

Na interface do usuário é possível realizar a leitura e parametrização do dispositivo de campo. A configuração da interface do usuário permite que estas funções sejam executadas conforme descrito abaixo.

- Estado de conexão: pode-se acompanhar através interface do usuário ou *display* o estado de conexão da aplicação, ou seja, se o usuário está conectado ou não com o dispositivo de campo.
- Status do motor: o motor pode ser ligado, desligado ou ter o sentido de giro alterado a pelo usuário.

- Velocidade do motor: foi construída uma matriz para configuração de frequência do motor, o que permite ao usuário ajustar a frequência de operação em uma escala de 0 a 100%, de acordo com as frequências mínima e máxima ajustadas no inversor de frequência.
- Dados do motor: foram coletados os dados do motor a partir do inversor de frequências. Essas informações permitem ao usuário conhecer as condições do equipamento em funcionamento e fazer a tomadas de decisão mais assertivas durante o processo. Seguem os dados coletados:
    - Frequência máxima;
    - Frequência mínima;
    - Tempo de aceleração;
    - Tempo de desaceleração;
    - Última falha apresentada;
    - Corrente nominal;
    - Rotação nominal;
    - Potência nominal;
    - Fator de potência nominal;
    - Tensão nominal;
    - Frequência nominal;
    - Rendimento nominal;
    - Resistência do estator.
- Gráficos: foram definidos gráficos para apresentar os históricos e valores momentâneos das seguintes grandezas:
    - Frequência;
    - Corrente;
    - Tesão;
    - Velocidade.

Dessa forma o usuário tem acesso as informações e configuração dos parâmetros dos dispositivos de campo de forma remota. Apenas com acesso à *internet* é possível fazer a leitura de um processo industrial e intervenção, quando necessário. Isso permite a otimização dos processos e tomadas de decisão mais assertivas.

5 CONCLUSÕES E CONTRIBUIÇÕES

Com o auxílio do monitoramento remoto é possível que o usuário tenha acesso a supervisão e controle de sistemas industriais. Esse acesso permite o acompanhamento da eficiência e desempenho das operações, reduz o tempo de resposta a eventos e falhas de forma a contribuir para tomadas de decisões mais rápidas e assertivas, ainda que a implementação, em fase de projeto, seja com a utilização de um dispositivo não robusto para chão de fábrica.

Na implementação realizada, o acesso aos parâmetros, a comunicação e o controle do dispositivo de campo foram alcançados. Os resultados obtidos a partir da interface realizada entre um dispositivo de campo (inversor de frequência), que possui apenas o protocolo Modbus RTU, com o *Node-Red* mostram claramente as possibilidades de implementação dos conceitos da Indústria 4.0 em plantas industriais equipadas com instrumentos sem a tecnologia para este fim. Os resultados obtidos no capítulo 4 evidenciam a possibilidade de obter informações, parâmetros dos dispositivos e realizar intervenções no processo a partir da interface do usuário.

O desenvolvimento do sistema para implementação dos conceitos da Indústria 4.0 foi baseado em tecnologias disponíveis e acessíveis, além de serem plataformas abertas ao usuário. Dentro da proposta inicial, o objetivo era de atingir os resultados levando em consideração não apenas o acesso a tecnologia, mas chegar neste resultado de forma viável e acessível aos usuários.

A validação do sistema proposto tomou como referência um inversor de frequências, comum nas plantas industriais. É possível o controle e acesso de outros dispositivos, como os CLPs, o que permite o acesso a um nível de dados superior em uma planta industrial. Para isso é necessário que os parâmetros do dispositivo em questão sejam analisados para que possa ser configurado e realizada a interface com o *Node-RED*.

A necessidade de implementação de um sistema de monitoramento remoto voltado para Indústria 4.0 para plantas industriais foi atendida, através dos resultados apresentados no Capítulo 4. Neste primeiro momento não foram realizados os ensaios referentes a possíveis interferências geradas por máquinas industriais, a montagem e simulação foram realizadas em ambiente não hostil.

Outra contribuição para os processos industriais “desatualizados” é que essa interface pode ser implementada em outros padrões disponíveis em campo. O Modbus é utilizado em dispositivos industriais, não sendo um protocolo recente, está presente em parte dos equipamentos instalados. No caso citado, a interface foi realizada para comunicação com o

padrão RS-485. Essa mesma aplicação pode ser implementada para o padrão RS-232, comum em CLPs e controladores de campo, utilizando um conversor RS-232 para TTL.

Assim, mesmo que não exista a possibilidade de atualização de uma planta industrial para dispositivos com tecnologias recentes, é possível que os conceitos da Indústria 4.0 sejam aplicados aos processos. O que possibilita o monitoramento remoto e diagnóstico para melhor eficiência dos processos.

5.1 Trabalhos futuros

Para continuação do trabalho realizado, é possível implementar algumas funções e ensaios para validação da solução em plantas com processos hostis.

- Implementação de uma interface para dispositivos que operam com o padrão RS-485.
- Levantamento de protocolos industriais que não possuem acesso as tecnologias da Indústria 4.0 e que possam, através da interface apresentada, serem conectados ao sistema remoto.
- Ensaios e utilização de circuitos robustos para aplicação da solução em ambientes industriais hostis.
- Possibilidade de implementação de uma interface que permita a conexão e configuração de dispositivos de campo sem a necessidade de acesso/alteração do código desenvolvido no controlador (que faz a interface) e *Node-Red*, sempre que um novo dispositivo tiver de ser monitorado ou que a solução seja aplicada à uma nova planta.

Com a implementação das funções e ensaios listados, a solução proposta pode ser aplicada de forma segura e eficiente aos processos e plantas industriais e permite uma interface simplificada dos dispositivos de campo para o usuário.

6 REFERÊNCIAS BIBLIOGRÁFICAS

[1] CAPELLI, A. Automação industrial: controle do movimento e processos contínuos, São Paulo, Editora Érica, 3ª ed., 2013, 240p.

[2] STEVAN JR., S. L.; LEME, M. O.; SANTOS, M. M. D. Indústria 4.0: fundamentos, perspectivas e aplicações, São Paulo, Editora Érica, 1ª ed., 2018, 200p.

[3] LUGLI, A. B.; SANTOS, M. M. D. Redes Industriais: Características, Padrões e Aplicações, São Paulo, Editora Érica, 1ª ed., 2014, 128p.

[4] TORRES, A. et al. Análise de Desempenho de Brokers MQTT em Sistemas de Baixo Custo. In: XXXVI Congresso da Sociedade Brasileira de Computação, p 2804-2815, 2016.

[5] Site da Internet: MQTT – Protocolos para IoT, visitado em 12/2022. https://embarcados.com.br/mqtt-protocolos-para-iot/

[6] Souza, D. S. Estudo da aplicação de um sistema IoT baseado no protocolo de comunicação MQTT a área da robótica industrial, 2018. 63 p. Dissertação (Mestrado em Engenharia Elétrica) Universidade Federal de Uberlândia, Minas Gerais, 2018.

[7] Site da Internet: O protocolo MQTT - Leve e simples. Perfeito para IoT e sistemas embarcados, visitado em 08/2022. https://www.gta.ufrj.br/ensino/eel878/redes1-2018-1/trabalhos-vf/mqtt/#

[8] Site da Internet: Redes Industriais. Artigo técnico de fabricante, visitado em 03/2022. https://www.smar.com/brasil/artigo-tecnico/redes-industriais

[9] Site da Internet: UNCTAD. (2022). ECN16/2022/2, Comisión de Ciencia y Tecnología para el Desarrollo - Industria 4.0 para el desarrollo inclusivo, visitado em 07/2022. https://unctad.org/system/files/official-document/ecn162022d2_es.pdf

[10] BAENA, F. et al. Learning Factory: The Path to Industry 4.0. In: Procedia Manufacturing, V.9, p. 73-80, 2017.

[11] SALTIÉL, R. M. F.; NUNES, F. A indústria 4.0 e o sistema hyundai de produção: suas interações e diferenças. V SIMEP-Simpósio de Engenharia de Produção. Anais... Joinville, 2017.

[12] JAIKAR, S.; IYER, K. R. A Survey of Messaging Protocols for IoT Systems. In: International Journal of Advanced in Management, Technology and Engineering Sciences, V. 8, n. 2, p. 510-514, 2018.

[13] Site da internet: GIBBS, M. Node-RED, wiring the Raspberry Pi to the IoT, visitado em 01/2023.
https://www.networkworld.com/article/3075329/node-red-wiring-the-raspberry-pi-to-theiot.html

[14] Site da Internet: Indústria 4.0: Entenda seus conceitos e fundamentos, visitado em 06/2022.
https://www.portaldaindustria.com.br/industria-de-a-z/industria-4-0/

[15] OLIVEIRA, J. D. Como podemos utilizar inteligência artificial no e-commerce? E-Commerce Brasil, 19 fev. 2020. Disponível em: <https://www.ecommercebrasil.com.br/artigos/como-podemos-utilizar-inteligencia-artificialno- e-commerce/>. Acesso em: 10 março 2021.

[16] Lima, P. B. SECURITY OF CYBER-PHYSICAL SYSTEMS AGAINST ACTIVE AND PASSIVE NETWORK ATTACKS: A DISCRETE EVENT SYSTEM APPROACH, 2021. 97 p. Tese (Doutorado em Engenharia Elétrica) Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2021.

[17] Ribeiro, K. S. Virtualização de uma estação de trabalho de um sistema de manufatura automatizado, 2022. 60 p. Projeto Final de Conclusão de Curso (Bacharel em Engenharia de Controle e Automação) – Universidade Estadual Paulista, Sorocaba, 2022.

[18] Righetto, S. B. Manutenção Preditiva 4.0: Conceito, Arquitetura e Estratégias de Implementação, 2020. 87 p. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) Universidade Federal de Santa Catarina, Florianópolis, 2020.

[19] Site da Internet: Zem, J. L.; Martins, I. R. ESTUDO DOS PROTOCOLOS DE COMUNICAÇÃO MQTT E COAP PARA APLICAÇÕES MACHINE-TO-MACHINE E INTERNET DAS COISAS, 2015, visitado em 01/2023.
http://ric-cps.eastus2.cloudapp.azure.com/bitstream/123456789/107/1/estudo_protocolos_comunicacao_mqtt_coap_aplicacoes_machine_internet_coisas.pdf

[20] Site da Internet: MQTT: The Standard for IoT Messaging, visitado em 10/2022.
https://mqtt.org/

[21] Site da Internet: MQTT V3.1 Protocol Specification, visitado em 01/2023.
https://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html

[22] Strack, G. Módulo de I/O Remoto MODBUS, 2011. 88 p. Projeto de Diplomação - Universidade Federal do Rio Grande do Sul, Porto Alegre, 2011.
https://www.lume.ufrgs.br/bitstream/handle/10183/33101/000787155.pdf

[23] Site da internet: Protocolo de Comunicação Modbus RTU/ASCII – VERSÃO 1.0, visitado em 06/2022.
https://www.dca.ufrn.br/~affonso/FTP/DCA447/modbus/modbus_manual.pdf

[24] Site da internet: Modicon Modbus Protocol Reference Guide, visitado em 10/2022.
https://modbus.org/docs/PI_MBUS_300.pdf

[25] Site da internet: MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b3, visitado em 10/2022.
https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

[26] Site da internet: MODBUS over serial line specification and implementation guide V1.02, visitado em 03/2022.
https://modbus.org/docs/Modbus_over_serial_line_V1_02.pdf

[27] Site da Internet: Introduction to the Modbus Protocol, visitado em 01/2023.

https://www.ccontrols.com/pdf/Extv9n4.pdf

[28] Site da internet: MODBUS MESSAGING ON TCP/IP IMPLEMENTATION GUIDE V1.0b, visitado em 03/2022.
https://modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf

[29] Site da Internet: Introduction to Modbus Serial and Modbus TCP, visitado em 01/2023.
https://www.ccontrols.com/pdf/Extv9n5.pdf

[30] Site da internet: MODBUS/TCP Security - Protocol Specification, visitado em 10/2022.
https://modbus.org/docs/MB-TCP-Security-v36_2021-07-30.pdf

[31] Site da Internet: Protocolo Modbus: Fundamentos e Aplicações, visitado em 08/2022.
https://embarcados.com.br/protocolo-modbus/

[32] HELD, Gilbert. **Ethernet Networks**, London, Wiley Editor, 2nd ed., 2000, 458p.

[33] BRANCO, M. V; et al. Aspectos de Diferenciação entre Laboratórios Remotos e Simuladores. In: XLV Congresso Brasileiro de Educação em Engenharia, 2014, Joinville, Brasil. Anais. Joinville, 2017.

[34] LOURENÇO, R. S. Laboratórios remotos-um estudo para a puc-rio. 2014. Disponível em: <http://www.pucrio.br/Pibic/relatorio_resumo2014/relatorios_pdf/ctc/ELE/ELERodrigo%20da%20Silva%20Louren%C3%A7o.pdf>.

[35] LOWE, D. et al. 'rating remote laboratory management systems (rlmss) for more efficient sharing of laboratory resources. Computer Standards & Interfaces, Elsevier, v. 43, p. 21–29, 2016.

[36] KUCGANT, M. Quando começou a integração entre automação e informação. **Controle & Instrumentação**, n. 143, p. 86-89, 2008.

[37] POPP, Manfred; WEBBER, Carl. **The Rapid Way to PROFINET**. PROFIBUS Nutzeorganisation, order 4182, 2004, 244p.

[38] LUGLI, A. B. SOFTWARE PARA ANÁLISE DE TOPOLOGIA E TRÁFEGO PARA REDES ETHERNET INDUSTRIAIS. 2007. 85 p. Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal de Itajubá, Itajubá, 2007.

[39] Site da Internet: Mapping of Modbus Registers to BACnet® Objects Using the BAS Remote, visitado em 11/2022.
https://www.ccontrols.com/pdf/Essentials1009.pdf

[40] Site da internet: Modbus RTU CFW500 – Manual do usuário, visitado em 01/2023.
https://www.servicedrive.com.br/wp-content/uploads/Manual-do-Usu%C3%A1rio-Modbus-RTU-CFW500-ServiceDrive-19-3012-6360.pdf

[41] Site da internet: Inversor de Frequência CFW500 V1.8X - Manual de Programação, visitado em 01/2023.
https://static2.weg.net/medias/downloadcenter/hee/h0b/WEG-cfw500-manual-de-programacao-10001469555-1.8x-manual-portugues-br.pdf

[42] Dias, C. X. S. SmartNode Dashboard: um framework front-end baseado em Node-RED para criação de City Dashboards, 2019. 71 p. Dissertação (Mestrado em Engenharia de Software) - Universidade Federal do Rio Grande do Norte, Natal, 2019.

[43] Site da internet: CONWAY-JONES, D. Node-red-dashboard, visitado em 10/2022.
https://github.com/node-red/node-red-dashboard

[44] BLACKSTOCK M.; LEA R. FRED: A Hosted Data Flow Platform for the IoT built using Node-RED. In: International Workshop on Web of Things. p. 1-5, 2016.

[45] Site da Internet: C Palmer and S. Shenoi (Eds.): Critical Infrastructure Protection III – Chapter 6: DESIGN AND IMPLEMENTATION OF A SECURE MODBUS PROTOCOL, 2009, visitado em 12/2022.
https://link.springer.com/chapter/10.1007/978-3-642-04798-5_6

[46] Site da Internet: JAFFEY, Toby. MQTT and CoAP, IoT protocols. 2014, visitado em 01/2023.
https://www.eclipse.org/community/eclipse_newsletter/2014/february/article2.php

[47] Site da Internet: Yuan, M. Getting to know MQTT, 2017, visitado em 08/2022.
https://developer.ibm.com/articles/iot-mqtt-why-good-for-iot/?mhsrc=ibmsearch_a&mhq=MQTT

[48] FABREGAS, E. et al. Developing a remote laboratory for engineering education. In: . [s.n.], 2011. v. 57, n. 2, p. 1686 – 1697. ISSN 0360-1315. Disponível em: http://www.sciencedirect.com/science/article/pii/S0360131511000716

[49] Site da Internet: POSTEL, Jon. **RFC 791 – Internet Protocol, DARPA Internet Program Protocol Specification**. University of Southern Califórnia, USA, September 1981, visitado em 03/2023.
https://www.rfc-editor.org/rfc/rfc791

[50] TANENBAUM, Andrew S. **Redes de Computadores**, São Paulo, Editora Campus, 3ª ed., 1997, 923p.

[51] FOROUZAN, Behrouz A. Protocolo TCP/IP, Porto Alegre, Editora Mc Graw Hill, 3ª ed., 2009, 896p.

7 APÊNDICES

Apêndice 1: Study and Application of the IO-Link Industrial Network in the Context of Industry 4.0.

Mixed Design of Integrated Circuits and Systems – MIXDES 2023

Study and Application of the IO-Link Industrial Network in the Context of Industry 4.0

Humberto Jose Gonzaga
Universidade Federal de Itajubá
Itajubá - Brazil
humbertogonzaga91@gmail.com

Tales C Pimenta
Universidade Federal de Itajubá
Itajubá - Brazil
tales@unifei.edu.br

***Abstract*—This study presents the concepts, definitions and applications of IO- Link System, highlighting its main characteristics, its implementation, rates of transmission and protocol. We have used the IEC 61131-9 standard as the main reference which considers *IO-Link* as the first I/O technology for communication with sensors and actuators as an international standard. We also discuss, as a case study, its use on a company dedicated to packaging machinery. It reduced the breakdown and downtime of machines by detecting their own maintenance needs before a failure occurs, thus improving productivity.**

***Keywords*—IO-Link, IO-Link Master, Smart Plants.**

I. Introduction

There is an increasingly quest for efficient and independent processes to meet the demands of industries for fast production as it has been the case for a long time. The first industrial revolution came with the steam machines, soon after started the concept of scale production, followed by the automation era, in which the first Programmable Logic Controllers – PLC were used. In this way, it was possible to have automated processes, thus promoting huge gains in product quality and process efficiency. The fourth industrial revolution has already begun, which is also be called "Industry 4.0" or "Industrial Internet of Things". This revolution aims to create "smart plants", which is possible with the use of smart sensors, which collect significant process data and send them directly to the system controller. The collected data can be used in real time, thus enabling the decision-making of the machine based on the information obtained [1, 2, 3].

Therefore, the Industry 4.0 makes it possible the implementation of intelligent and independent processes, energy reduction, quality control throughout the production process and also the reduction of breakdown and downtime of machines, since they can detect their own maintenance needs before a failure occurs [3, 4].

This work aims to study the concepts, definitions and applications of the *IO-Link* system, highlighting its main characteristics, means of implementation and applications. We have carried out a case study in the packaging machines of the company EDSON, in order to verify the improvements presented of using the *IO-Link* system.

The remaining of this work is divided into four chapters, the Definitions Concepts, IO-Link Protocol, Case Study and Conclusion.

II. Concepts and Definitions

Currently, sensors are becoming more intelligent thanks to the use of microprocessors [4] [9] [10]. Until short ago, the data acquired on a plant were not entirely used, due to the lack an easy and intelligent route to take them to the control system. In this context, the *IO-Link* technology was developed, which makes this traffic possible. It collects the data provided by those intelligent sensors through the existing cabling. It also offers the advantage of being an independent fieldbus. That is possible due to the computing resources, increasingly powerful microprocessors and wide bandwidth networks prepared for the transport of this data [4, 9, 10].

The *IO-Link* technology can also be connected to most existing control architectures, as the existing cabling can be used and since it does not require a specific network. It can be connected to the most communication protocols, such as PROFIBUS, PROFINET and EtherNet/IP, since it requires only one IO-Link master to communicate. The *IO-Link* parameterization can be done remotely and its information is stored in the master. In case of a replacement, the parameters of the replaced device will be automatically redefined. Another advantage of *IO-Link* is that it takes a single A/D conversion, thus avoiding conversion losses, while a standard (analog) system present at least three conversions, thus generating a loss of information [4, 9, 10].

A. Definition

IO-Link may be considered as the first I/O technology for communication with sensors and actuators to be adopted as a standard by the international standard IEC 61131-9. This point-to-point communication is based on the long-range connection of the sensor and the 3-wire actuator without the need for additional requirements related to cable material [9].

It is an open communication protocol used to integrate I/O devices, thus allowing not only the exchange of information between sensors and actuators, but also with the controller. In this way, there is an overview of the system and the status of the devices in real time, therefore controlling situations considered abnormal for operations, and consequently keeping them stable. It is possible because *IO-Link* devices are connected to an *IO-Link* master, which reads the status information of each device, stores it and, when necessary, sends it to higher-level communication systems, which upon receiving the data, treats them appropriately. Fig 1 depicts the pyramid of the automation system [7, 9, 10].

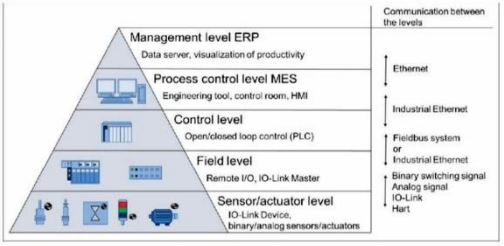

Fig. 1. Automation system pyramid [7].

B. I/O Link Components and Interface

An *IO-Link* system consists of [8]:

- *IO-Link* Master;
- *IO-Link* devices (sensors, actuators or both);
- Three- or five-conductor *unshielded* cable.

The *IO-Link* master is responsible for establishing the connection between the *IO-Link* devices and the control process architecture. This system can be connected to the most communication protocols, and can be installed both in the control cabin and remotely [8, 9].

During startup, the *IO-Link* master switches the ports to user-selected modes, which can be INACTIVE, DI (Digital Input), DO (Digital Output), FIXEDMODE or SCANMODE. When a communication to a device is requested, the master uses an activation pulse to start the process, then it automatically adjusts the baud rate for the communication ports COM1 (4.8 kbit/s), COM2 (38.4 kbit/s) or COM3 (230.4 kbit/s) and checks the parameters of the connected device through VendorID and DeviceID. If there is any inconsistency between the parameters of the device and those stored in the master, the parameters of the device will be overwritten or the ones contained in the master will be updated. Fig 2 shows an example of IO-Link architecture [9, 10].

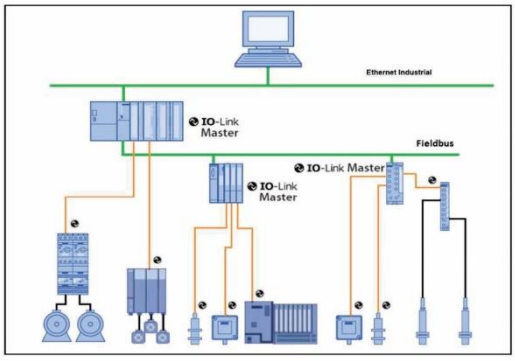

Fig. 2. IO-Link architecture [8].

IO-Link devices can be selected with two connection options, namely: Port Class A and Port Class B.

The Port Class A is a connector consisting of four pins, in which just three of them are used, two for powering the device (+24 V, 0 V) and the other for communication between the device and the master. This type of connection allows a maximum output of 200 mA. The fourth pin can be used as an additional signal line provided for in the IEC 61131-2 standard. Fig 3 illustrates the Port Class A connector [7, 9, 10]. Fig 4 shows the connection between an *IO-Link* Port Class A Master and a Port Class A device.

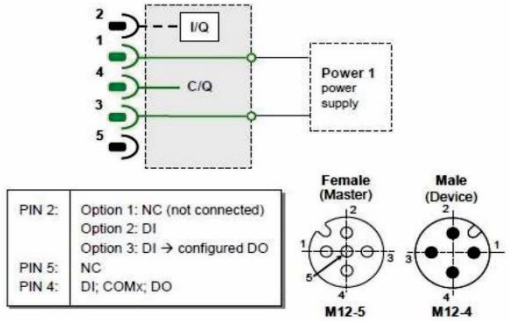

Fig. 3. *Port Class* A [7, 9].

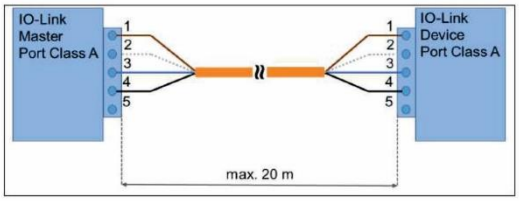

Fig. 4. Connection between master and *IO-Link* Port Class A device [7].

Port Class B is a five-pin connector that is specified for devices that require an additional power supply. In this model, besides the three conductors of Port Class A, it also has two other conductors responsible for supplying an additional power to the device. The maximum consumption allowed for this model depends on the type of connection used. In the case of the M12 connection, the maximum is 3.5 A.

The maximum cable distance used between the devices and the master is 20 meters, without the need to use shielding. Fig 5 illustrates the Port Class B connector [7, 9, 10]. Fig 6 shows connection between master and *IO-Link* Port Class B device.

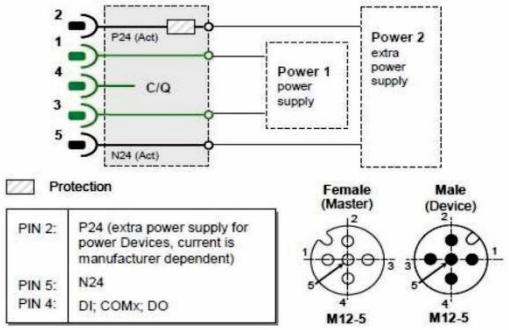

Fig. 5. Port Class B [7, 9].

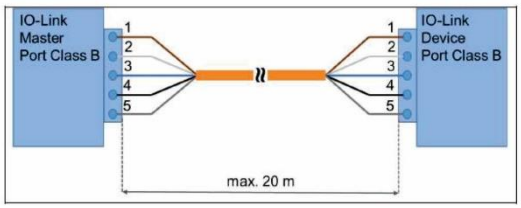

Fig. 6. Connection between master and IO-Link Port Class B device [7, 9].

If a Port Class A device is connected to a Port Class B master via a 4 or 5 conductor cable, it must be ensured that the PIN 2 conductor (white wire) is not connected, otherwise the device may malfunction [7, 9, 10].

C. *Maximum Device Capacity*

The *IO-Link* technology is based on the point-to-point connection and is connected to a superior network, as seen in Fig 2. Thus, the number of devices in a network is defined by the superior network. For instance, the PROFIBUS DP allows up to 126 devices (masters or slaves) per bus. In this case, the *IO-Link* master behaves like a slave on the PRROFIBUS DP network.

III. IO-Link Protocol

This section describes the *IO-Link* Protocol, highlighting information such as operating modes, transmission rates and data types, among others.

A. *Operation Modes*

The *IO-Link* ports of the master can basically work in four modes of operation: IO-Link, DI, DQ and Deactivated.

In *IO-Link* mode, the port is used for communication between the master and the devices. DI mode is used so that the port behaves like a digital input, being recognized in this way by the master. DQ allows the master to recognize the port as a digital output, and finally, Deactivated mode is enabled for unused ports [5, 6, 10].

B. *Transmission Rates*

Although the master can support all baud rates, the devices can operate only one baud rate. Therefore, the master automatically adapts to the baud rate supported by the device, which is defined by the manufacturer. The transmission rates are as follows [5, 6, 10]:

- COM 1: 4.8 kbit/s;
- COM 2: 38.4 kbit/s;
- COM 3: 230.4 kbit/s (optional, per Specification 1.0).

C. *System Response Time*

The system response time consists of several factors, including the baud rate and the minimum cycle of each device, which is provided by the IO Device Description (IODD). This value has great influence on the response time, in addition to the master's internal processing time, which is also included in this calculation [6, 10].

Devices with different minimum cycle times can be configured on the same master, so the system response time can vary considerably according to the devices connected to the network.

D. *Data Exchange*

In general, three types of data exchange are performed for the *IO-Link* system:

- Cyclic Process Data: are transmitted in a data frame, where the size is specified by the device. Data from 0 to 32 bytes are possible for each input and output [6, 10].
- Device Parameters (acyclic data): can be parameters, identification data and diagnostics information. Those data can be read or written to the devices and are only exchanged upon request from the *IO-Link* master [6, 10].
- Events (acyclic data): can be diagnostic and error messages and warnings or maintenance data. Those messages are read and transmitted to a controller or a Human Machine Interface by the *IO-Link* master. They can also be transmitted by the master itself, reporting, for example, broken wires and communication failures. While an event is being read, no parameter data can be changed [6, 0].

Each port of a master has its own data link layer. System management identifies the presence of connected devices, adjusts their ports and, if necessary, adjusts the devices themselves, so that they match the predetermined settings and characteristics of each connected device [10]. Fig 7 shows the data exchange.

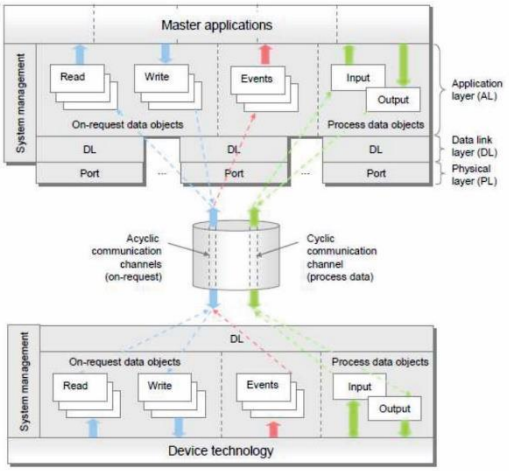

Fig. 7. Data exchange [10].

E. *Transmission Characteristics*

The physical topology is point-to-point, between the devices and the master, and connections can be made through Port Class A and Port Class B.

The baud rates for the *IO-Link* V1.1 are4.8 kbps, 38.4 kbps, and 230.4 kbps. Version V1.0 of the specification was prepared at first, then improved and new functions were added leading to version V1.1 [6, 10]. Important additions made to version V1.1 are:

- Parameter assignment server function (data storage).
- Data baud rate of 230.4 kbps (mandatory for the *IO-Link* master).
- Process data size per port is 32 bytes.

IO-Link V1.0 masters are only compatible with V1.0 devices. V1.1 masters are compatible with both V1.1 and V1.0 devices. The "parameter assignment server" function and the baud rate of 230.4 kbps can only be used if supported by the device [10].

The data exchange between a master and its device is performed through a sequence of messages, called "M-sequence". This "M-sequence" comprises a message from the master followed by a message from the device. Data must be transmitted as a "big-endian" sequence, in which the most significant octet (MSO) must be sent first, followed by the least significant octets, in descending order, with the least significant octet (LSO) sent last [10].

According to each specific need of an actuator or sensor, a type of "M-sequence" can be defined, and the length of the messages can vary according to the type of message and the direction of data transmission [10].

Fig 8 depicts the exchange of data from the master, making it clear that this exchange is understood by a message from the master followed by a message from the device.

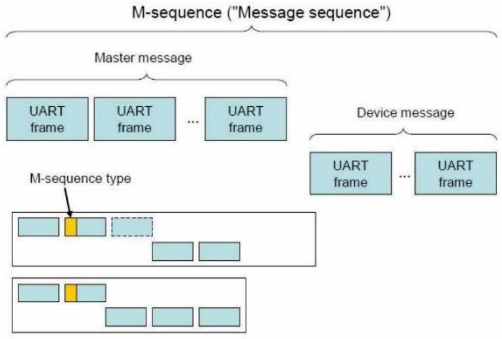

Fig. 8. Message sequence [10].

Fig 9 shows that the main message starts with the octet MC (M-sequence Control), followed by CKT (Check/Type) and optionally followed by PD (Process Data) and/or OD (On-request). Date). The device message optionally starts with the PD and/or OD octets, followed by the CKS (Check/Start) octet. There is also an overview of the types of M-sequence. Fixed M-sequence types consist of TYPE_0, TYPE_1_1, TYPE_1_2 and TYPE_2_1 through TYPE_2_6, the variables are TYPE_1_V and TYPE_2_V. [10].

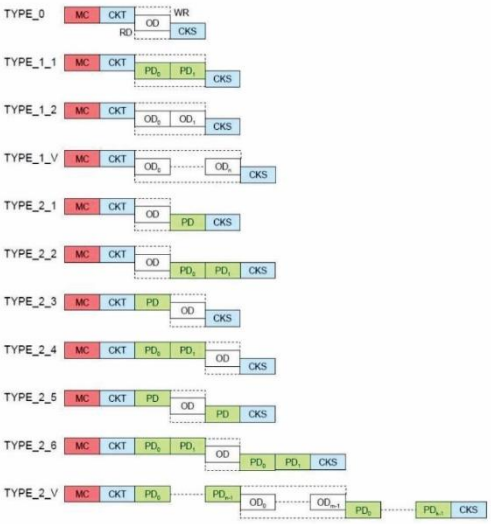

Fig. 9. Overview of M-sequence types [10].

An UART frame is used for encoding "data octet". The Single-Drop Digital Communication Interface – SDDCI UART frame is a structured string as shown in Fig 10 [10].

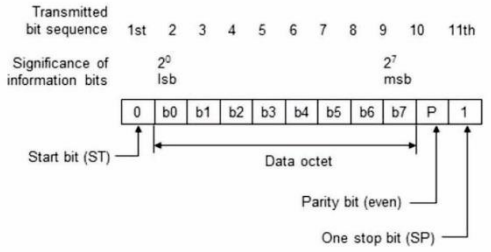

Fig. 10. Format of an SDCI UART frame [10].

F. IO Device Description - IODD

It is responsible for storing device information for system integration. Each device has its IODD and stores communication properties, device parameters with range of values and default value, identification, process and diagnostic data, device data, text description, device illustration and logo.

The IODD structure is standardized, regardless of the manufacturer. This structure is also represented in the same way by the configuration tools of the main manufacturers. In this way, it is possible to guarantee adaptation and handling for all IO-Link devices. Different versions of IODD for V1.0 and V1.1 devices are available from the manufacturer [9, 10].

IV. Case Study

EDSON is a company dedicated to packaging machinery. It observed that, with the wear and tear of the machines, it was difficulty to change and configure the sensors, which had to be

done at its location. A new sensor would require multiple wires be incorporated into a control cabinet, usually under a limited space [11].

By using *IO-Link*, a single standard sensor cable can be connected to an RFID reader. Each reader has the possibility to handle more than 255 tags, at data rate higher than old installations, and does not require extra spaces in the control cabinets. The devices can still be installed up to 20 meters from the master, so the distance not an obstacle [11].

The *IO-Link* technology allows the controller to track the cycles of each implanted device in order to monitor the correct time for preventive maintenance and avoid unwanted downtime. The inclusion and/or replacement of devices has also been optimized. If any of them fails, the controller immediately detects and alerts the maintenance team via a human-machine interface (HMI), as diagnostics are a great advantage of *IO-Link*. Furthermore, the configuration of the sensors is conducted automatically, and their configurations are protected by means of an HMI, which prevents incorrect configurations made by the installer and also guarantees the correct parameterization for each specific application [11].

Edson uses *IO-Link* laser sensors. The precision provided is about 1 mm, which was previously not economically viable to be achieved. They can also detect variations in slope in a stack of products. That improves the process leading to a more precise handling of products, and eliminates the congestion caused by inaccurate distance readings, in addition to congestion alerts [11].

The packaging machines also uses *IO-Link* photoelectric sensors located in places susceptible to dust accumulation, which previously was not possible to detect until there was a malfunction. By using *IO-Link* sensors, these dust accumulations can be monitored and an alert signal for maintenance can be issued before the machine stops. Thus, the machine remains operating longer without reducing the performance due to unwanted failures or maintenance, as it identifies the exact failure device [11].

The main advantages observed by using *IO-Link* in Edson machines are [11]:

- Real-time smart switching. The master takes care of configuring a replaced device;
- Self-diagnosis alerts operators of a problem and identifies the exact device;
- Wide variety of sensors of greater capacity than conventional sensors;
- Reduces the work of the operator to replacing devices and check for failures;
- Sensor settings cannot be changed by a field operator;
- Easy and economical updating of devices;
- *IO-Link* and non-IO-Link devices can work together, enabling a balance between cost and performance.

Fig 11 and 12 illustrate the applications carried out in the Edson machine.

Fig. 11. Photoelectric IO-Link sensors in an EDSON packaging machine.

Fig. 12. Photoelectric IO-Link sensors in an EDSON packaging machine.

V. Conclusions

The IO-Link system is a technology that has brought a range of advantages to industrial applications, being a system that can operate in plants already installed, since it can interact with different communication protocols, in addition to being installed in the same network that already uses equipment with other technologies, different from IO-Link.

The concept of Industry 4.0 can be clearly observed in the applications of this technology, promoting the installation of intelligent plants capable of performing a self-diagnosis and communicating the maintenance system, performing the collection and storage of important data for the continuous improvement of processes and, still, offers practicality when replacing and configuring devices, where, once parameterized and having their information inserted in the PLC that controls the plant. Thus, it is enough to replace a device with another, of the same model, and the parameterization is performed automatically, without risk of incorrect configurations, which can affect the correct functioning of the process.

In this way, IO-Link technology has been expanding and gaining more and more space in the most diverse applications, having its range of devices increasingly diversified and supplied by different manufacturers, which is not a problem, since the structure of the IODD is standardized, with each device having its own. Thus, the concept of Industry 4.0 has been introduced

in large and small companies, still in a timid way, however, with great potential still to be explored and offered with the help of the IO-Link protocol.

ACKNOWLEDGMENT

This work was supported by CAPES, CNPq and FAPEMIG.

REFERENCES

[1] Gilchrist, *A. Industry 4.0 - The Industrial Internet of Things*. 1st ed., Apress Berkeley.

[2] Industrial Networks, <https://www.smar.com/en/industrial-networks>, Acessed on 02 August 2022.

[3] Learn the Fundamentals of IO-Link Technology, <https://www.youtube.com/watch?v=_JxLmz-g2XM>, Acessed on 05 August 2022.

[4] IO-Link in context of smart factory, <https://www.youtube.com/watch?v=-fkjQgEzvmQ>, Acessed on 06 August 2022.

[5] IO-Link Technology and Application - System Description, <https://www.profibus.com/download/io-link-technology-and-application-system-description>, Acessed on 09 August 2022.

[6] IO-Link System Description - Technology and Application, <http://io-link.com/share/Downloads/At-a-glance/IO-Link_System_Description_eng_2018.pdf>, Acessed on 15 August 2022.

[7] IO-Link Design Guideline, <http://io-link.com/share/Downloads/Planung/IO-Link_Design_Guideline_eng_2018.pdf>, Acessed on 22 August 2022.

[8] Protocol IO Link, <https://www.citisystems.com.br/io-link/#:~:text=O%20IO%20Link%20%C3%A9%20um,atuadores%20no%20ch%C3%A3o%20de%20f%C3%A1brica. >, Acessed on 22 August 2022..

[9] IO-Link Interface and System – Specification, <www.io-link.com/share/Downloads/Spec-Interface/IOL-Interface-Spec_10002_V112_Jul13.pdf>, Acessed on 22 August 2022..

[10] International Electrotechnical Commission. IEC IEC 61131-9, 2013.

[11] Making a "case" for IO-Link, <http://profinews.com/2017/08/making-a-case-for-io-link/>, Acessed on 25 August 2022.

Apêndice 2: Comandos para resetar falhas, ligar, desligar e controlar o sentido de giro do motor.

```
// #### LIGAR O MOTOR NO SENTIDO DIRETO DE GIRO ####
if(comando == "habilita")
{
  ir(aux_sentido == 2 or aux_sentido == 0) au16data[0] = 0x13;
  if(aux_sentido == 1) au16data[0] = 0x17;
  aux_comando = 1;
}

// #### DESLIGAR O MOTOR POR RAMPA DE DESACELERAÇÃO ####
if(comando == "desabilita")
{
  if(aux_sentido == 2 or aux_sentido == 0) au16data[0] = 0x12;
  if(aux_sentido == 1) au16data[0] = 0x16;
  aux_comando = 2;
}

// ###### DEFINIÇÃO DO SENTIDO DE GIRO DO MOTOR ######

// #### SENTIDO DIRETO ####
if(comando == "horario")
{
  if(aux_comando == 2 or aux_comando == 0) au16data[0] = 0x16;
  if(aux_comando == 1) au16data[0] = 0x17;
  aux_sentido = 1;
}

// #### SENTIDO REVERSO ####
if(comando == "anti")
{
  if(aux_comando == 2 or aux_comando == 0) au16data[0] = 0x12;
  if(aux_comando == 1) au16data[0] = 0x13;
  aux_sentido = 2;
}

// ###### COMANDO PARA RESET DE FALHA DO INVERSOR ######

if(comando == "reset")
{
  au16data[0] = 0x92;
}
```

Apêndice 3: Comandos para definição do tipo de acesso, mensagens no display e controle do motor.

```
// ###### Task2code ######
void Task2code( void * pvParameters ){

  for(;;){

  //  #### MENSAGENS APRESENTADAS NO DISPLAY ####
  lcd.setCursor(0,0);
  lcd.print("Comando:");
  lcd.print(comando);
  lcd.print("        ");
  lcd.setCursor(0,1);
  lcd.print("Enviando Dados:");

  //  #### TIPO DE ACESSO - NOTEBOOK OU CELULAR ####
  if(comando == "notebook")    controle = 1;
  if(comando == "celular"){    controle = 2;
    MQTT.publish(PUBLISH_CONEXAO, "Conectado no Celular");
  }
  if(comando == "desconectar") controle = 0;

    switch(u8state) {

      case 0:
     if(millis() > (u32wait + 100)) u8state++;
     break;

     // #### CASO PARA COMANDO DO MOTOR ####
     case 1:

     au16data[0] = 0x12;

// #### COMANDOS PARA CONTROLE REALIZADO PELO NOTEBOOK ####
     if(controle == 1)
     {
       // #### CONTROLE DE VELOCIDADE DO MOTOR ####
       velocidade = comando.toInt();
       if(velocidade == porcentagem[velocidade-1])
       {
         velocidade = map(velocidade, 0, 100, 0, 16382);
         au16data[1] = velocidade;

         if(aux_comando == 1)
         {
      if(aux_sentido == 2 or aux_sentido == 0) au16data[0] = 0x13;
      if(aux_sentido == 1) au16data[0] = 0x17;
         }
       }

       // #### LIGAR O MOTOR NO SENTIDO DIRETO DE GIRO ####
         aux_sentido = 2;
       }

       // ###### COMANDO PARA RESET DE FALHA DO INVERSOR ######

       if(comando == "reset")
       {
         au16data[0] = 0x92;
       }
     }
```

```
// #### COMANDOS PARA CONTROLE REALIZADO PELO CELULAR ####
     if(controle == 2)
     {
       // #### CONTROLE DE VELOCIDADE DO MOTOR ####
       velocidade = comando.toInt();
       if(velocidade == porcentagem[velocidade-1])
       {
         velocidade = map(velocidade, 0, 100, 0, 16382);
         au16data[1] = velocidade;

         if(aux_comando == 1)
         {
      if(aux_sentido == 2 or aux_sentido == 0) au16data[0] = 0x13;
      if(aux_sentido == 1) au16data[0] = 0x17;
         }
       }

       // #### LIGAR O MOTOR NO SENTIDO DIRETO DE GIRO ####
       if(comando == "habilita")
       {
         ir(aux_sentido == 2 or aux_sentido == 0) au16data[0] = 0x13;
         if(aux_sentido == 1) au16data[0] = 0x17;
         aux_comando = 1;
       }

       // #### DESLIGAR O MOTOR POR RAMPA DE DESACELERAÇÃO ####
       if(comando == "desabilita")
       {
         if(aux_sentido == 2 or aux_sentido == 0) au16data[0] = 0x12;
         if(aux_sentido == 1) au16data[0] = 0x16;
         aux_comando = 2;
       }

       // ###### DEFINIÇÃO DO SENTIDO DE GIRO DO MOTOR ######

       // #### SENTIDO DIRETO ####
       if(comando == "horario")
       {
         if(aux_comando == 2 or aux_comando == 0) au16data[0] = 0x16;
         if(aux_comando == 1) au16data[0] = 0x17;
         aux_sentido = 1;
       }

       // #### SENTIDO REVERSO ####
       if(comando == "anti")
       {
         if(aux_comando == 2 or aux_comando == 0) au16data[0] = 0x12;
         if(aux_comando == 1) au16data[0] = 0x13;
         aux_sentido = 2;
       }

       // ###### COMANDO PARA RESET DE FALHA DO INVERSOR ######
       if(comando == "reset")
       {
         au16data[0] = 0x92;
       }
     }
     u8state++;
     break;

// #### DEFINICAO DOS PARAMETROS PARA LEITURA OU ESCRITA ####
     case 2:
     telegram[0].u8id       = 0x01;        // endereço do escravo
     telegram[0].u8fct      = 0x10;        // código de função para escrever
     telegram[0].u16RegAdd  = 0x2AA;       // endereço de inicio da escrita
     telegram[0].u16CoilsNo = 2;     // numero de registradores a escrever
     telegram[0].au16reg = au16data;       // ponteiro para o vetor de memória
     master.query(telegram[0]);      // envia a solicitação ao escravo
     u8state++;
     break;
```

Apêndice 4: Código JSON.


```
{
  "id": "0428b83f09cde19e",
  "type": "tab",
  "label": "Flow 1",
  "disabled": false,
  "info": "",
  "env": []
},
{
  "id": "74f77114f9ac6106",
  "type": "tab",
  "label": "CFW500",
  "disabled": true,
  "info": ""
},
{
  "id": "d3c6c02d74025e30",
  "type": "tab",
  "label": "CFW500",
  "disabled": false,
  "info": ""
},
{
  "id": "1b7aa63716cd294a",
  "type": "airspace",
  "z": "74f77114f9ac6106",
  "name": "spacer",
  "group": "5dbedb5118c46940",
  "order": 1,
  "width": 1,
  "height": 1
},
{
  "id": "d63c613bcec0993d",
  "type": "ui_spacer",
  "z": "74f77114f9ac6106",
  "name": "spacer",
  "group": "5dbedb5118c46940",
  "order": 1,
  "width": 1,
  "height": 1
},
```

8 ANEXOS

Anexo 1: Best Paper Award.