Guilherme Balduino Lopes

Desenvolvimento e implementação de um sistema de controle e monitoramento remoto no contexto da Indústria 4.0

Uberlândia, MG

2022

Guilherme Balduino Lopes

Desenvolvimento e implementação de um sistema de controle e monitoramento remoto no contexto da Indústria 4.0

Trabalho de Conclusão de Curso de Engenharia de Controle e Automação da Universidade Federal de Uberlândia - UFU - Campus Santa Mônica, como requisito para a obtenção do título de Bacharel em Engenharia de Controle e Automação.

Universidade Federal de Uberlândia - UFU
Faculdade de Engenharia Elétrica - FEELT

Orientador Prof. Dr. Renato F. Fernandes Jr.

Uberlândia, MG
2022

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

L864 Lopes, Guilherme Balduino, 1997-
2023 Desenvolvimento e implementação de um sistema de controle e monitoramento remoto no contexto da Indústria 4.0 [recurso eletrônico] / Guilherme Balduino Lopes. - 2023.

Orientador: Renato Ferreira Fernandes Junior.
Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Uberlândia, Graduação em Engenharia de Controle e Automação.
Modo de acesso: Internet.
Inclui bibliografia.
Inclui ilustrações.

1. Processos de fabricação - Automação. I. Fernandes Junior, Renato Ferreira ,1971-, (Orient.). II. Universidade Federal de Uberlândia. Graduação em Engenharia de Controle e Automação. III. Título.

CDU: 67.02:681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:
Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

Este trabalho é dedicado ao meu falecido pai - Valdenes Lopes de Araújo que, durante toda a sua vida, almejou o sucesso dos filhos acima do próprio.

Agradecimentos

Os agradecimentos principais são direcionados aos meus pais Maguilânia Balduino Lopes e Valdenes Lopes de Araújo, juntamente ao meu irmão Gledson Balduino Lopes, pelo amor, apoio e incentivo aos estudos.

Aos meus professores, em especial ao professor Dr. Renato Ferreira Fernandes Junior, pela paciência, por todo conhecimento compartilhado e por sempre se dispor a ajudar e incentivar em minha evolução durante o desenvolvimento deste trabalho.

Aos meus colegas de curso e amigos por todo o apoio e por me darem forças durante todos estes anos.

E finalmente, à Universidade Federal de Uberlândia que, através de toda a comunidade acadêmica, proporcionou tanto para minha formação.

"What we endure in time will in the end just make us strong."
(Myles Richard Kennedy, 2018)

Resumo

A busca constante pela otimização de processos tem sido o principal agente no desenvolvimento de novas tecnologias que visam a melhoria do desempenho das linhas de produção industrial. Neste contexto, este trabalho propõe-se a abordar alguns dos conceitos que envolvem a chamada Indústria 4.0 (I4.0), como IoT, Big Data e Cloud Computing. Para mais, é apresentado o desenvolvimento de um sistema de controle e monitoramento remoto, seguro, escalável, não invasivo e de baixo custo, responsável pelo monitoramento de indicadores de desempenho em plantas industriais que possuam suporte ao protocolo Modbus TCP. Tem como objetivo principal, prover uma solução simples, porém robusta, capaz de integrar tanto sistemas modernos quanto legados à nova realidade da I4.0, e assim, auxiliar em tomadas de decisões que possam cooperar em uma maior eficiência dos equipamentos e, consequentemente, também melhorar o resultado final da produtividade e lucratividade na produção industrial. Este sistema possui ainda a capacidade de automatizar alguns processos como por exemplo, controlar atuadores ou enviar mensagens via e-mail/SMS quando algum parâmetro exceder um limite pré-determinado.

Palavras-chaves: Cloud Computing; Indústria 4.0; Industrial Internet of Things; Modbus; MQTT.

Abstract

The constant search for process optimization has been the main agent in the development of new technologies aimed at improving the performance of industrial production lines. In this context, this work proposes to raise some important concepts of the so-called Industry 4.0 (I4.0), such as IoT, Big Data and Cloud Computing. Furthermore, it presents the development of a remote, secure, scalable, non-invasive and low-cost control and monitoring system, responsible for monitoring performance indicators in industrial plants that support the Modbus TCP protocol. Its main objective is to provide a simple but robust solution, capable of integrating both modern and legacy systems to the new reality of I4.0, and thus assist in decision-making that can cooperate with greater equipment efficiency and, consequently, also improve the end result of productivity and profitability in industrial production. This system also has the ability to automate some processes, such as controlling actuators or sending messages via e-mail/SMS when any parameter exceeds a predetermined limit.

Key-words: Cloud Computing; Industry 4.0; Industrial Internet of Things; Modbus; MQTT.

Lista de figuras

Lista de abreviaturas e siglas

4IR	4° Industrial Revolution - 4° Revolução Industrial
A2A	Application to Application - Aplicação para Aplicação
A2P	Application to Person - Aplicação para Pessoa
ADU	Application Data Unit - Unidade de dados do aplicativo
ARMv8	Advanced RISC Machine - Máquina RISC Avançada
ASCII	American Standard Code for Information Interchange - Código Padrão Americano para Intercâmbio de Informações
AWS	Amazon Web Service
BI	Business Intelligence - Inteligência de Negócios
BLE	Bluetooth Low Energy - Bluetooth de Baixa Energia
CIs	Critical Infrastructure - Infraestruturas Críticas
CLP	Controlador Lógico Programável
CNI	Confederação Nacional da Indústria
CoAP	Constrained Application Protocol - Protocolo de Aplicativo Restrito
CPSs	Cyber Physical Systems - Sistemas Ciber-Físicos
CRC	Cyclic Reduncany Check - Verificação de Redundância Cíclica
CSI	Camera Serial Interface - Interface Serial da Câmera
DA	Data Access - Acesso de Dados
DC	Direct Current - Corrente Contínua
DNP3	Distributed Network Protocol - Protocolo de Rede Distribuída
DP	Decentralized Peripherals - Periférico Descentralizado
DSI	Display Serial Interface - Interface Serial de Display
EC2	Elastic Compute Cloud - Nuvem de Computação Elástica
EIA	Electronic Industries Alliance - Aliança das Indústrias de Eletrônica

FF	Foundation Fieldbus
GPIO	General Purpose Input/Output - Entrada/Saída de Uso Geral
HAT	Hardware Attached on Top - Hardware Anexado na Parte Superior
HDMI	High-Definition Multimedia Interface - Interface Multimídia de Alta Definição
HMI	Human Machine Interface - Interface Homem-Máquina
HSE	High Speed Ethernet - Redes de Alta Velocidade
HTTPS	Hyper Text Transfer Protocol Secure - Protocolo de Transferência de Hipertexto Seguro
I4.0	Industry 4.0 - Indústria 4.0
IA	Inteligencia Artificial
IBM	International Business Machines Corporation - Corporação Internacional de Máquinas de Negócios
IEC61850	International Electrotechnical Commission's (61850 Standard) - Comissão Eletrotécnica Internacional (Norma 61850)
IED	Intelligent Electronic Device - Dispositivo Eletrônico Inteligente
IEEE	Institute of Electrical and Electronic Engineers - Instituto de Engenheiros Eletricistas e Eletrônicos
IIoT	Industrial Internet of Things - Internet das Coisas Industrial
IoT	Internet of Things - Internet das Coisas
IP	Internet Protocol - Protocolo de Internet
JSON	JavaScript Object Notation - Notação de Objeto JavaScript
LAN	Local Area Networks - Redes Locais
LASEC	Laboratório de Automação, Sistemas Eletrônicos e Controle
LPDDR2	Low-Power Double Data Rate
M2M	Machine to Machine - Máquina para Máquina
MBAP	Modbus Application Protocol
MD	Material Design - Design Material

ML	Machine Learning - Aprendizado de Máquinas
MQTT	Message Queuing Telemetry Transport - Transporte de Telemetria de Fila de Mensagens
NoSQL	Not Only SQL - Não Só SQL
OPC	Open Platform Communications - Comunicações de Plataforma Aberta
PA	Process Automation - Automação de Processo
PDU	Protocol Data Unit - Unidade de Dados de Protocolo
PeD	Pesquisa e Desenvolvimento
PoE	Power over Ethernet - Alimentação pela Ethernet
QoS	Quality of Service - Qualidade de Serviço
RF	Radio Frequency - Radio Frequência
RRU	Read Request Units - Unidades de solicitação de leitura
RS	Recommended Standard - Padrão Recomendado
RTU	Remote Terminal Unit - Unidade Terminal Remota
SCADA	Supervisory Control And Data Acquisition - Sistema de Supervisão e Aquisição de Dados
SDRAM	Synchronous Dynamic Random-Access Memory - Memória de Acesso Aleatório Dinâmica Síncrona
SEBRAE	Serviço Brasileiro de Apoio às Micro e Pequenas Empresas
SD	Secure Digital - Digital Seguro
SDK	Software Development Kit - Kit de Desenvolvimento de Software
SQL	Structured Query Language - Linguagem de Consulta Estruturada
SMS	Short Message Service - Serviço de Mensagens Curtas
SNS	Simple Notification Service - Serviço de Notificação Simples
SQS	Simple Queue Service - Serviço de Fila Simples
SSL	Secure Sockets Layer - Camada de Soquetes Segura
TCP	Transmission Control Protocol - Protocolo de Controle de Transmissão

TI	Tecnologia da Informação
TLS	Transport Layer Security - Segurança da Camada de Transporte
TO	Tecnologia Operacional
UA	Unified Architecture - Arquitetura Unificada
UDP	User Datagram Protocol - Protocolo de Datagrama do Usuário
URL	Uniform Resource Locator - Localizador Padrão de Recursos
USB	Universal Serial Bus - Barramento Serial Universal
VPN	Virtual Private Network - Rede Privada Virtual
WRU	Write Request Units - Unidades de solicitação de gravação

Sumário

1 Introdução

O cenário industrial está passando por mudanças exponenciais. A crescente necessidade de otimizar os processos de produção, aumentando a eficiência dos equipamentos e a qualidade dos produtos, tem levado vários grupos de pessoas da área de PeD (Pesquisa e Desenvolvimento) a contribuir com o desenvolvimento de novas tecnologias capazes de auxiliar no aprimoramento do desempenho das linhas de produção em vários setores da indústria (PIVOTO et al., 2021).

Uma medida que se trata de uma técnica consolidada de otimização de utilidades na linha de produção e que vem se desenvolvendo em uma velocidade surpreendente nos últimos anos, é a implementação do conceito de Indústria 4.0 (I4.0), um movimento universal que está proporcionando uma profunda transformação das empresas por meio de tecnologias como IoT (Internet of Things), Computação na Nuvem (Cloud Computing), Big Data Analytics e Inteligência Artificial (CUNHA et al., 2016). Ela se caracteriza por utilizar essas tecnologias digitais para otimizar os processos, melhorar a qualidade de produtos e serviços, reduzir o consumo de energia e contribuir para uma gestão mais assertiva.

Segundo Pivoto et al. (2021), adotada como parte da High-Tech Strategy 2020 Action Plan em 2011, a I4.0 é uma iniciativa estratégica do governo Alemão desenvolvida para revolucionar os processos de manufatura através da união de uma série de pilares que permitem a fusão do mundo físico com o mundo digital. Esta união transforma uma simples planta industrial em um planta inteligente por meio de tecnologias embarcadas e softwares/firmwares capazes de monitorar, controlar e automatizar sistemas completos.

A análise do consumo de energia nas indústrias é feita através de medições das variáveis elétricas, tensão e corrente instantânea, que se tornam essenciais para a obtenção de indicadores chave de desempenho (KPIs ou Key Performance Indicator) responsáveis por ajudar a medir a performance ou eficácia de alguma atividade realizada em uma empresa (ENTENDA. . . , 2017). Nas indústrias, onde é predominante o uso de motores elétricos, estes indicadores de desempenho ajudam a prever e identificar falhas, planejar manutenções preventivas, identificar queda de rendimento ou desgaste excessivo em equipamentos elétricos.

É comum encontrar indústrias que monitoram e automatizam seus processos através de diferentes protocolos de redes industriais. As redes *Fieldbus*, também chamadas de redes de campo, servem para controlar a distribuição de dados que segue em toda a planta industrial e para ajudar a gerenciar inúmeros processos de automação por meio de uma ágil e precisa troca de informações, comumente realizada entre sensores, computadores,

posicionadores, atuadores, etc. Existem ainda redes industriais baseadas em Ethernet, as quais fazem com que máquinas e diferentes dispositivos se conectem rapidamente e com precisão, e troquem informações e dados com segurança – utilizando a mesma linha. Além disso, é possível se conectar através de cabos ou não, como as redes sem fio (QUAIS..., 2018).

Neste contexto, o presente trabalho aborda de uma forma prática e de baixo custo, o desenvolvimento de um sistema SCADA (Supervisory Control And Data Acquisition) remoto e personalizável, capaz de se conectar com equipamentos industriais que possuam suporte ao protocolo Modbus TCP (protocolo baseado em Ethernet) e de convergir suas informações para a nuvem através de um protocolo baseado em IoT - o MQTT (Message Queuing Telemetry Transport), onde essas possam ser analisadas em tempo real ou posteriormente pelo setor estratégico da empresa de uma forma fácil, precisa e, principalmente, segura. Além disso, o sistema também é capaz de gerar alarmes que podem atuar em plantas e enviar e-mails/SMS personalizados.

1.1 Justificativas

A capacidade de obter lucro e eficiência em uma unidade produtiva está relacionada diretamente com o grau de otimização de cada processo produtivo. Além disso, de acordo com o 7º Congresso Brasileiro de Inovação da Indústria realizado pelo SEBRAE e CNI (2018), cerca de 42% da energia consumida nos processos produtivos é desperdiçada de alguma forma, sendo boa parte perdida em calor. Segundo Venturelli (2020), é comum se encontrar grandes linhas de produção, seja de manufatura ou processo, com grandes quantidades produzidas, mas que não obtêm a lucratividade esperada por unidade de produto e, muitos destes problemas, estão relacionados à otimização da linha produtiva.

Isso ocorre pois várias empresas ainda fazem acompanhamentos manuais através de planilhas eletrônicas e ainda realizam as medições de consumo de energia da maneira tradicional, o que geralmente requer o deslocamento de técnicos até cada ponto de consumo para realizar cada medição, implicando em um custo elevado, difícil acesso à informação e alta possibilidade de erro humano ao realizar as medições (CARVALHO et al., 2018). Sendo assim, surge a necessidade da implementação de novos recursos tecnológicos com o objetivo não só de aprimorar os processos de produção a fim de se obter uma melhor lucratividade, mas também de trazer soluções para problemas ambientais, melhorar a qualidade do ambiente de trabalho e, principalmente, diminuir o consumo desnecessário de recursos.

Entretanto, como afirma Villarim, Souza e Baiocchi (2021), apesar do rápido avanço da tecnologia, para que empresas possam se adequar à era da Indústria 4.0, é necessária a substituição de equipamentos legados por dispositivos inteligentes, o que pode acarretar

um custo financeiro considerável.

> É provável que grandes indústrias e empresas possam custear essas despesas, mas para as de pequeno ou médio porte, essa realidade torna-se um obstáculo, principalmente em países com economias em desenvolvimento. Uma possível solução para permitir que essas indústrias tenham a oportunidade de participar da 4ª Revolução Industrial é introduzir nelas um processo de "IoTização" de seus equipamentos (VILLARIM; SOUZA; BAIOCCHI, 2021).

Ainda segundo Villarim, Souza e Baiocchi (2021), um processo de "IoTização" pode ser definido como um processo para introduzir em equipamentos já existentes na indústria características de sensoriamento, comunicação e processamento digital de informações. Desta forma, é possível que empresas com um orçamento um pouco menor ainda consigam se manter competitivas no mercado.

Este trabalho tem como principal motivação a aplicação, no contexto da Indústria 4.0, de conhecimentos adquiridos através de matérias do curso de Engenharia de Controle e Automação como Informática Industrial, Redes Industriais, Sistemas Embarcados, Sistemas Distribuídos, entre outras.

1.2 Objetivos

O principal objetivo do trabalho é apresentar o desenvolvimento de um sistema de baixo custo, capaz de permitir o monitoramento e o controle de equipamentos que possuam suporte ao protocolo Modbus TCP, de forma remota, segura, escalável e não invasiva para que estes possam ser melhor aproveitados dentro do conceito da nova indústria.

1.2.1 Objetivos Específicos

- Desenvolvimento de um gateway de protocolos de comunicação Modbus/MQTT (nomeado "Modbus2MQTT Gateway"), o qual consiste em uma aplicação responsável por fazer a comunicação entre a planta industrial e a nuvem, através de um protocolo baseado em IoT;
- Construção de uma Infraestrutura Cloud para armazenamento dos dados adquiridos pelo gateway, em um banco de dados do tipo valores-chave NoSQL na nuvem de forma redundante, escalável e com criptografia em repouso;
- Desenvolvimento da Infraestrutura na nuvem para que além de armazenar e apresentar os dados, ela gere alarmes capazes de enviar notificações (e-mail/SMS) e/ou atuarem nas plantas.

- Desenvolvimento de uma interface SCADA, a qual disponibiliza o monitoramento e o controle da planta através da internet de forma totalmente segura;
- Implementação e validação do sistema final em planta didática de medição de consumo de uma esteira transportadora de cargas do LASEC.

1.3 Trabalhos Publicados

Durante o desenvolvimento do projeto, foram escritos e publicados 3 artigos científicos em anais de eventos periódicos e 1 em uma revista, todos com coautoria do professor orientador Dr. Renato F. Fernandes Jr., sendo:

- **DESENVOLVIMENTO DE UM SISTEMA DE MONITORAMENTO E ANÁLISE DE DADOS PARA EFICIÊNCIA ENERGÉTICA EM INDÚSTRIAS 4.0**
  Apresentado na XIX Conferência de Estudos em Engenharia Elétrica - XIX CEEL - realizado on-line no período de 13 a 17 de dezembro de 2021, pela Universidade Federal de Uberlândia. ISSN 2596-2221.
- **A REMOTE MQTT-BASED DATA MONITORING SYSTEM FOR ENERGY EFFICIENCY IN INDUSTRIAL ENVIRONMENTS**
  Redigido em Inglês e apresentado no XXIV Encontro Nacional de Modelagem Computacional (XXIV ENMC) e XII Encontro de Ciência e Tecnologia de Materiais (XII ECTM), realizado on-line no período de 13 a 15 de outubro de 2021. DOI: 10.29327/154013.24-2. ISBN: 978-65-86084-34-4.
- **DESIGN AND IMPLEMENTATION OF A MODBUS TO MQTT GATEWAY DEVELOPED WITH PYTHON AND AMAZON WEB SERVICES**
  Redigido em Inglês e apresentado no XXV Encontro Nacional de Modelagem Computacional (XXV ENMC), XIII Encontro de Ciência e Tecnologia de Materiais (XIII ECTM), 9a Conferência Sul em Modelagem Computacional (9º MCSul) e IX Seminário e Workshop em Engenharia Oceânica (IX SEMENGO), realizado on-line no período de 19 a 21 de outubro de 2022. DOI: 10.29327/1142685.25-9. ISBN: 978-85-5722-474-2.
- **UM SISTEMA REMOTO DE MONITORAMENTO DE DADOS COM BASE EM MQTT PARA EFICIÊNCIA ENERGÉTICA EM AMBIENTES INDUSTRIAIS**
  Redigido em Inglês e publicado em 17 de dezembro de 2021 em VETOR - Revista De Ciências Exatas E Engenharias, 31(2), 25–35 como versão estendida do trabalho apresentado em XXIV ENMC e XII ECTM. DOI: 10.14295/vetor.v31i2.13726. ISSN 0102-7352, ISSN Eletrônico 2358-3452.

2 Referenciais Teóricos

Desde o início da industrialização, saltos tecnológicos levaram a mudanças de paradigma que hoje são expostas às chamadas "Revoluções Industriais": no campo da mecanização (a chamada 1ª Revolução Industrial); do uso intensivo de energia elétrica (a chamada 2ª Revolução Industrial); e da digitalização e automação generalizada (a chamada 3ª Revolução Industrial) (LASI et al., 2014).

Neste sentido, a 4ª Revolução Industrial (4IR), ou Indústria 4.0 como foi chamada na Alemanha em 2011, é um movimento universal que se consolida na profunda transformação de indústrias tradicionais em indústrias inteligentes por meio da incorporação de tecnologias e conceitos inovadores como Internet of Things (IoT), Cloud Computing, Sistemas Ciber-Físicos (do Inglês Cyber-physical Systems ou CPSs), Inteligência Artificial (IA), Big Data Analytics, Manufatura Aditiva e outras. Se caracteriza por otimizar os processos de produção, melhorar a qualidade de produtos e serviços, reduzir o consumo desnecessário de energia e contribuir para uma melhor gestão. Além disso, todas as informações necessárias, como alarmes, próximas etapas de trabalho e disposições de segurança, são fornecidas em tempo real (ZAGONEL, 2021; CARVALHO et al., 2018; PIVOTO et al., 2021).

Para que seja possível esta transformação da indústria, é necessário que haja uma interconexão entre o mundo físico e o mundo virtual através da "IoTização" como convencionado por Villarim, Souza e Baiocchi (2021) ou como dito por Aris (2020): uma convergência entre as plantas industriais e a área de Tecnologia da Informação (TI), o que resulta em uma junção de múltiplas tecnologias díspares e traz desafios de gerenciamento e segurança nunca vistos antes. Ainda segundo Aris (2020), é possível chamar essa nova realidade de TI Industrial, um ambiente onde três tipos de tecnologia convergem: Tecnologia da Informação, Tecnologia Operacional (TO) e Internet das Coisas Industrial (Industrial Internet of Things, IIoT).

2.1 Tecnologia Operacional e a IIoT

A TO se refere a redes industriais que são normalmente usadas nas estratégias de controle industrial. Estas redes industriais têm se consolidado a décadas e são compostas de diferentes tecnologias dependendo do tipo de aplicação e processo industrial. Destacam-se as redes Profibus DP, Devicenet e Profinet nas redes de controle de manufatura, e redes FF, HSE, Profibus PA no controle de processos contínuos. No contexto de redes de sistemas de energia destacam-se as redes Modbus, DNP3 e IEC61850 (REYNDERS; MACKAY; WRIGHT, 2004).

Porém, apesar de resolverem uma grande gama de aplicações, estas redes de campo (Fieldbus) são bastante limitadas, uma vez que não possuem suporte para várias aplicações específicas, como por exemplo: aplicações onde os dispositivos fiquem trocando de rede ou que tenham endereço desconhecido; aplicações onde seja necessária uma camada de segurança dos dados ou controle de acesso; e principalmente, aplicações que envolvam o envio de dados para serviços em nuvem (MAZUR; KAY; KREITER, 2013; SILVA, 2019).

Para a resolução de problemas como segurança dos dados e disponibilidade dos mesmos na nuvem, faz-se necessária a implementação de novas tecnologias responsáveis pela integração dessas redes de campo (TO) ao setor de TI. Para isso, existem os chamados IoT gateways (ou IIoT gateways), dispositivos de última geração dispostos a "intermediar" essas duas pontas, comunicando os dados extraídos da TO para servidores em data centers locais ou na nuvem, de uma forma estritamente segura e confiável.

Segundo Bhattacharjee (2019), em sua forma mais simples, um gateway pode ser apenas um hardware ou software desenvolvido para coletar e agregar dados de dispositivos de entrada e saída. No entanto, neste novo cenário de IIoT, o uso mais comum dos gateways é como dispositivo de borda. Ele pode se conectar diretamente a equipamentos de campo (sensores, atuadores, etc.) ou por meio de CLPs (Controladores Lógicos Programáveis), IEDs (Dispositivos Eletrônicos Inteligentes), sistemas SCADA (Sistema de Supervisão e Aquisição de Dados), HMIs (Interfaces Homem-Máquina), etc., que agregam dados de campo. Deve, portanto, suportar uma ampla variedade de interfaces de entrada e saída, incluindo conexões com fio, sem fio, serial e Ethernet.

Entre as principais tecnologias presentes nos IIoT gateways existentes que integram o chão de fábrica à web, é possível citar protocolos de comunicação como MQTT, OPC DA/UA e CoAP, sendo o protocolo MQTT um dos que mais se destaca pela sua facilidade de implementação e, principalmente, por ser uma tecnologia aberta (BHATTACHARJEE, 2019).

Com relação aos protocolos de redes de energia, eles ainda são pouco utilizados quando se fala de monitoramento do ambiente industrial, predominando a utilização principalmente em empresas de geração, transmissão e distribuição de energia. Porém, apesar de antigo, o protocolo Modbus ainda tem uma grande aceitação no mercado e está presente na maioria dos equipamentos.

Neste contexto, 3 importantes tecnologias utilizadas no desenvolvimento do trabalho proposto são apresentadas a seguir: sistemas SCADA (TO), o protocolo Modbus (TO) e o protocolo MQTT (IoT ou IIoT).

2.1.1 Sistemas SCADA

SCADA é a sigla em inglês para *Supervisory Control And Data Acquisition* que na tradução para o português significa Sistema de Supervisão e Aquisição de Dados. Basicamente, são um conjunto de softwares e funcionalidades que realizam o monitoramento e controle de uma determinada planta ou processo, aglomerando dados da TO (CORREA et al., 2019; METZGER, 2022). Estes desempenham um papel significativo no fornecimento de acesso remoto, monitoramento e controle de infraestruturas críticas (CIs), que incluem sistemas de energia, sistemas de distribuição de água, usinas de gás, sistemas de coleta de águas residuais, entre outros, como afirma Alimi et al. (2021).

Ainda segundo Alimi et al. (2021), historicamente, quando os sistemas SCADA foram implantados, o objetivo era melhorar a eficiência e eficácia dos CIs com pouca consideração dos potenciais desafios de segurança futuros. Na realidade, esses problemas de segurança do SCADA eram principalmente devidos a desafios ambientais e falhas de equipamentos por conta de desgaste. No entanto, o avanço da tecnologia nos últimos anos mudou o foco para vários desafios de segurança, que incluem invasões e ataques cibernéticos.

Hardwares como Unidades Terminais Remotas (RTUs) e CLPs servem como pontos de coleta locais para adquirir informações de sensores. Uma vez coletados, os dados dos sensores podem ser usados diretamente pelo uso do software SCADA ou salvos para revisão posterior. Os sistemas SCADA podem ajudar a monitorar e controlar processos de forma local ou remota. É possível além disso, influenciar e controlar um sistema SCADA sem ter que responder diretamente a cada evento. Por exemplo, usando regras baseadas em lógica, os operadores podem designar a conclusão de determinadas ações quando os sensores detectam anormalidades (METZGER, 2022).

Além disso, como afirma Metzger (2022), ao fornecer visibilidade em tempo real sobre o estado dos ativos e operações, sistemas SCADA ajudam os proprietários e operadores de negócios a tomar decisões mais inteligentes, melhorar a eficiência e minimizar o tempo de inatividade. Ainda segundo Metzger (2022), as fábricas modernas monitoram os dados dos sensores das máquinas para prever a manutenção, monitorar a velocidade de saída e aumentar a segurança do operador. Se um equipamento se tornar menos eficiente em um determinado ponto de seu ciclo de manutenção, pode ser mais lucrativo realizar ações de manutenção com mais frequência para evitar um gargalo na produção. Sem um sistema SCADA, seria difícil reconhecer esses padrões manualmente.

2.1.2 Protocolo Modbus

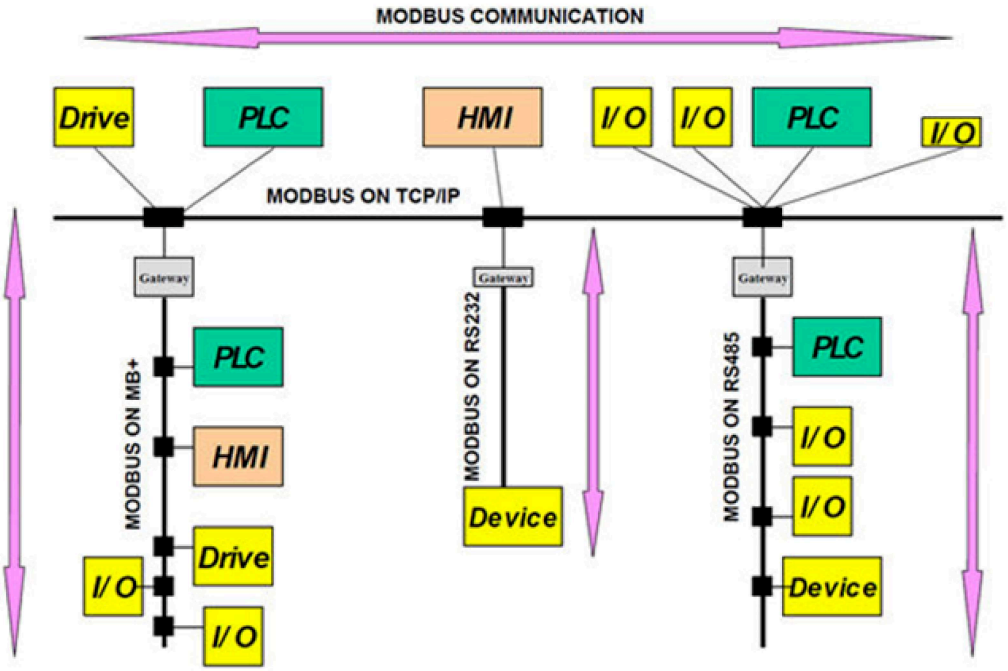
Desenvolvido pela Modicon (atual Schneider Electric) em 1979 para comunicação entre equipamentos industriais, o Modbus é um padrão verdadeiramente aberto e o protocolo

de rede mais utilizado no ambiente de fabricação industrial, tendo sido implementado por centenas de fornecedores em milhares de dispositivos, como afirma a própria Organização Modbus (THE-MODBUS-ORGANIZATION, 2022).

Este protocolo possui dois modos de operação, ASCII (*American Standard Code for Information Interchange*) e RTU (*Remote Terminal Unit*). Além disso, existem também derivações do Modbus como o Modbus TCP que utiliza o protocolo TCP/IP e Ethernet para transportar os dados, e o Modbus Plus, que utiliza a passagem de *token* para aumentar a velocidade e permitir multi-mestres (KELLER, 2017; MOORE-INDUSTRIES, 2021). Como o Modbus Plus é o único que ainda possui seus direitos reservados pela Schneider Electric, este não será abordado no presente trabalho (THE-MODBUS-ORGANIZATION, 2022).

A Figura 1 mostra um exemplo de uma rede Modbus, onde gateways fazem a conversão de um tipo de Modbus para outro, podendo assim se obter uma comunicação unificada de toda uma planta industrial.

Figura 1 – Exemplo arquitetura comunicação Modbus.



Fonte: The-Modbus-Organization (2022).

Todas as mensagens Modbus são enviadas no mesmo formato e se diferem somente na forma como são codificadas. No Modbus ASCII, as mensagens são codificadas em hexadecimal, usando caracteres ASCII de 4 bits. Para cada byte de informação são necessários dois bytes de comunicação, duas vezes mais do que com Modbus RTU ou Modbus TCP, o que torna o Modbus ASCII o mais lento dos três, no entanto, ele é bastante adequado quando usados links de modem telefônico ou rádio (RF) (MOORE-INDUSTRIES,

2021; THE-MODBUS-ORGANIZATION, 2022).

No Modbus RTU, os dados são codificados em binário e requerem apenas um byte de comunicação por byte de dados. Isso é ideal para uso em redes multi-drop (um barramento no qual todos os componentes estão conectados ao circuito elétrico), em velocidades de 1.200 a 115Kbps, sendo 9.600 e 19.200bps as velocidades mais comuns. Seu tipo de conexão é serial, baseado no modelo *Client-Server* (Cliente-Servidor), permitindo um cliente a se conectar com até 247 servidores (32 por segmento), cada um com um endereço preestabelecido de 1 a 247. Caso seja enviada uma mensagem ao endereço 0, todos os servidores a recebem (mensagem *broadcast*). É válido ressaltar que, por estes endereços, o cliente Modbus pode tanto ler, quanto escrever informações nos servidores Modbus (SILVEIRA, 2016; MOORE-INDUSTRIES, 2021).

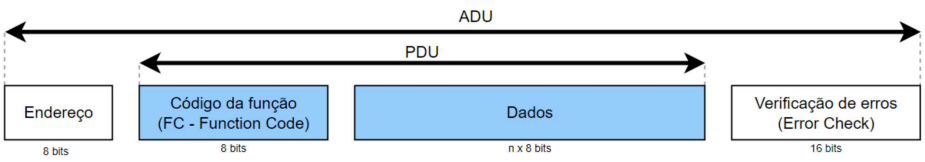
Basicamente existem 3 tipos de interface de transmissão de dados no Modbus RTU: RS-485, RS-422 e RS-232, onde "RS" é acrônimo para *Recommended Standard* (THE-MODBUS-ORGANIZATION, 2022). Sendo também possível encontrar estes padrões com a sigla "EIA" (*Electronic Industries Alliance*), como EIA-232, por exemplo.

O padrão RS-232 é utilizado apenas em comunicações do tipo ponto a ponto, ou seja, só admite dois dispositivos na rede, que no caso do protocolo Modbus representa o cliente e 1 servidor. A velocidade máxima desse padrão está em torno de 115Kbps, mas em alguns casos podem ser encontradas taxas um pouco maiores, com uma distância máxima entre os dispositivos da rede em torno de 30m (CORREA et al., 2019; SILVEIRA, 2016).

Já o padrão RS-485 é muito utilizado na indústria e sem dúvida é um dos padrões mais utilizados pelo protocolo Modbus. Esse padrão permite trabalhar com taxas de comunicação que podem chegar a 12Mbps e em alguns casos até 50Mbps. Vale lembrar que quanto maior o comprimento da rede, menor será a velocidade de comunicação, onde a distância máxima da rede está em torno de 1200m, e o número máximo de dispositivos no barramento da rede é de 32 (CORREA et al., 2019; SILVEIRA, 2016).

Segundo Keller (2017), para entender o protocolo Modbus é importante entender o funcionamento do *frame*, o qual é apresentado na Figura 2, onde ADU (*Application Data Unit*) representa todo o *frame* e o PDU (*Protocol Data Unit*) somente os dados.

Figura 2 – *Frame* Modbus RTU.



Fonte: Adaptado de Keller (2017).

As requisições sempre partem do mestre e são processadas pelos escravos. Um

escravo só pode escrever no barramento quando o mestre o solicitar, ou seja, o primeiro byte for seu endereço. Uma vez que o escravo confirme que o pedido foi para ele, este executa o comando solicitado e responde a mensagem com o código da função e os dados da requisição (KELLER, 2017; THE-MODBUS-ORGANIZATION, 2022).

O segundo byte da mensagem envida pelo Servidor é o código de função, cujo valor diz ao escravo qual comando deve ser executado. A Figura 3, adaptada de Keller (2017), apresenta uma tabela com as principais funções Modbus. Caso ocorra algum erro na requisição, o cliente responde indicando o código da função modificado e outras informações sobre os erros no campo de dados. Este código da função modificado possui o *bit* mais significativo em nível lógico 1, o que é reconhecido como indicação de erro (KELLER, 2017).

Figura 3 – Códigos de funções Modbus.

				Código de funções (FC)		
				Código	Sub código	Hex
Acesso de dados	Acesso ao bit	Entradas Discretas Físicas	Leitura de Entradas Discretas (Read Discrete Inputs)	02		01
		Bobinas físicas ou bits inte	Leitura de Bobinas (Read Coils)	01		01
			Escrita Bobina Única (Write Songle Coil)	05		05
			Escrita Multiplas Bobinas (Write Multiple Coils)	15		0F
	Acesso a 16 bits	Registros de entrada física	Leitura Registro de Entrada (Read Input Register)	04		04
		Registros de saída física	Leitura Registro de Retenção (Read Holding Register)	03		03
			Escrita Registro Único (Write Single Register)	06		06
			Escrita Multiplos Registros (Write Multiple Registers)	16		10
			Leitura/Escrita Multiplos Registros (Read/Write Multiple Registers)	23		17
			Registro de Gravação de Máscara (Mask Write Register)	22		16
			Leitura fila FIFO (Read FIFO queue)	24		18
Acesso a arquivos			Leitura Registro de Arquivo(Read File record)	20		14
			Escrita Registro de Arquivo (Write File record)	21		15
Diagnóstico			Leitura status de Exceção (Read Exception status)	07		07
			Diagnóstico (Diagnostic)	08	00-18,20	08
			Obter contador de eventos Com (Get Com event Couter)	11		0B
			Obter Log de Eventos Com (Get Com Event Log)	12		0C
			ID do servidor de relatórios (Report Server ID)	17		11
			Leitura da identificação do dispositivo (Read device identification)	43	14	2B
Outro			Transporte Interface encapsulada (Encapsulated interface Transport)	43	13, 14	2B

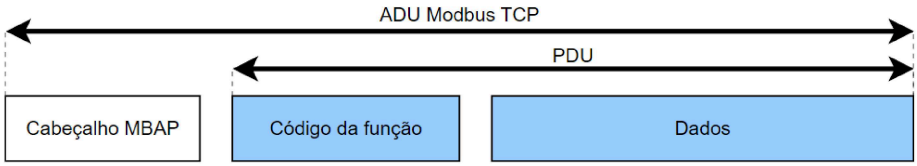
Fonte: Adaptado de Keller (2017).

Por conseguinte, o Modbus TCP, também referenciado como Modbus *over* Ethernet (ou Modbus via Ethernet), é simplesmente pacotes Modbus encapsulados em um padrão TCP/IP (*Transmission Control Protocol/Internet Protocol*), ou UDP/IP (*User Datagram Protocol*), o que possibilita a imediata e fácil conexão e comunicação de dispositivos com suporte Modbus TCP através de uma rede local Ethernet/WiFi ou Fibra.

É adicionado ao *frame* (Fig. 4) o cabeçalho MBAP (*Modbus Application Protocol*), o qual é utilizado para identificar o ADU do Modbus na rede TCP/IP. Além disso, a verificação de erros por CRC é removida, visto que esta é implementada pela própria

rede Ethernet. Este cabeçalho MBAP possui 7 *bytes* no total, divididos em 4 campos: o de identificação de transação (2 *bytes*); o identificador de protocolo (2 *bytes*); o de comprimento da mensagem (2 *bytes*); e o de identificação do servidor (1 *byte*) (THE-MODBUS-ORGANIZATION, 2022; KELLER, 2017).

Figura 4 – *Frame* Modbus TCP.



Fonte: Adaptado de Keller (2017).

Diferentemente do Modbus RTU, o Modbus TCP permite muito mais endereços e possui maior facilidade para que um Servidor se comunique com vários Clientes.

O padrão Ethernet no protocolo Modbus possui algumas variações, podendo chegar a 100Mbps ou até 10Gbps. A porta padrão do Modbus TCP é a 502. Sua distância máxima pode variar de 100m até próximo de 200m dependendo do tipo de cabo utilizado e das condições de instalação do mesmo (MOORE-INDUSTRIES, 2021; CORREA et al., 2019).

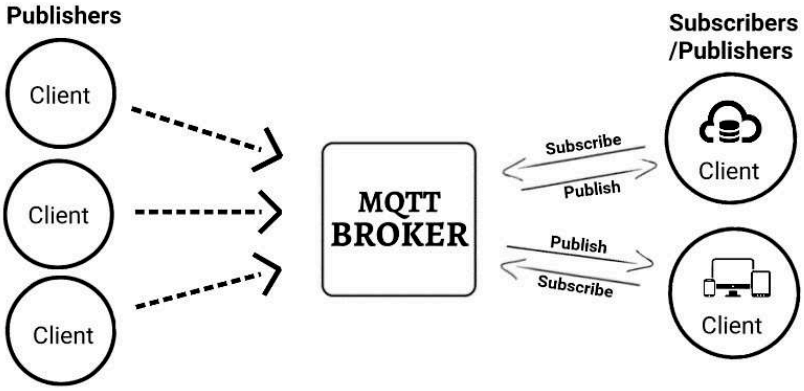
Uma das grandes vantagens do Modbus é que ele pode operar em praticamente todos os meios de comunicação, incluindo fios de par trançado, wireless, fibra ótica, Ethernet, modems telefônicos, telefones celulares e microondas. Isso significa que uma conexão Modbus pode ser estabelecida em uma planta nova ou existente com bastante facilidade. Quanto à segurança, o protocolo Modbus não possui nenhum mecanismo previsto: não possui senha, autorização, certificados ou qualquer outro sistema de segurança (MOORE-INDUSTRIES, 2021; THE-MODBUS-ORGANIZATION, 2022).

2.1.3 Protocolo MQTT

Uma das principais tecnologias IoT utilizadas na comunicação entre uma rede de dispositivos e a nuvem é o protocolo MQTT (Message Queuing Telemetry Transport). A arquitetura MQTT é simples de implementar, possui mecanismos de segurança de dados e uso moderado da largura de banda da rede. É um protocolo de rede do tipo *Publish/Subscribe* (Publica/Assina) destinado a sensores e pequenos dispositivos móveis tendo como principal uso fazer as máquinas trocarem informações, modo de comunicação conhecido como M2M (Máquina para Máquina) (ALMEIDA, 2015). Essa tecnologia foi desenvolvida pela IBM no final da década de 1990 e seu objetivo original era conectar sensores de satélites ou oleodutos. Apesar de ter sido criado há algum tempo, sua aplicabilidade ainda é excepcionalmente útil na atualidade, inclusive em diversos ramos empresariais (SILVA, 2019).

O padrão *Publish/Subscribe* de troca de mensagens usado no MQTT, funciona da seguinte forma: elementos que desejam publicar informações, fazem através do Broker, o intermediário que enfileira estas publicações recebidas e as envia aos elementos da rede que se subscreveram/assinaram para receber esta determinada informação (ALMEIDA, 2015). A Figura 5 mostra um exemplo da arquitetura do MQTT. Nela, os clientes desempenhando somente papel de *Publishers* poderiam ser sensores em uma rede de automação, IHMs, CLPs, IEDs, ou um sistema SCADA publicando suas mensagens a um tópico específico por exemplo, enquanto os clientes trabalhando como *Publishers* e *Subscribers* poderiam ser bancos de dados na nuvem, ferramentas de análises de dados, robôs autônomos ou usuários finais, que poderiam tanto receber quanto enviar mensagens entre si.

Figura 5 – Estrutura simplificada protocolo MQTT.



Fonte: Autoria própria (2020).

Apesar do Broker representar um elo de fragilidade na rede ao centralizar as comunicações, ele permite um desacoplamento entre as partes comunicantes, algo não possível em modelos de comunicação do tipo cliente/servidor (ALMEIDA, 2015).

Na arquitetura do protocolo MQTT a identificação das mensagens se dá por meio de tópicos. Este tópico, lembra o conceito de *Uniform Resource Locator* (URL), onde os níveis hierárquicos são separados por barras ("/") (ALMEIDA, 2015).

Por exemplo, pode-se ter o tópico "casa02/sala01/sensores/temperatura" e o tópico "casa02/sala01/atuadores/aquecedor", onde o primeiro poderia ler a temperatura medida pelo sensor da sala01 da casa02, já o segundo, poderia definir um *setpoint* para a temperatura da mesma sala.

Para se ter uma maior flexibilidade à estrutura de tópicos, podem ser utilizados caracteres curingas, chamados de *wildcards*. O caractere "+" tem a função de generalizar um nível dos tópicos e é utilizado em algum nível intermediário do tópico. Por exemplo, o tópico "casa02/+/sensores/umidade" poderia ser usado para assinar os sensores de umidade de todas as salas da casa02 (ALMEIDA, 2015; KELLER, 2017).

Já o caractere "#" é necessário quando deseja-se generalizar múltiplos níveis do

tópico, logo, é utilizado no final do tópico. Por exemplo, o tópico "casa01/sala05/sensores/#" poderia ser usado para assinar todos os sensores da sala05 da casa01 (ALMEIDA, 2015; KELLER, 2017).

Vale ressaltar que o sistema reserva o caractere "$" para uso interno do *Broker*, tornando o tópico disponível somente para leitura. Por exemplo, para o *Broker* Mosquitto, a quantidade de clientes ativos é dada pelo tópico "$SYS/broker/clients/total" (ALMEIDA, 2015; KELLER, 2017).

Segundo Almeida (2015), a conexão do cliente com o Broker, seja ele assinante ou publicador, é feita originalmente via TCP, com opções de login (nome de usuário e senha) e uso de criptografia (SSL/TLS). Almeida (2015) afirma ainda que é possível encontrar também outros meios físicos, com MQTT rodando em links seriais, por exemplo.

No protocolo MQTT, todo processo de conexão também estabelece um nível desejado de qualidade de serviço (QoS, de Quality of Service) indicando como deve ser a relação entre os elementos de comunicação. Almeida (2015) explica que são previstos 3 níveis de qualidade de serviço, sendo:

- **QoS 0 (*at most once*)** - Conhecido como "*best effort*" ou melhor esforço. Assemelha-se ao protocolo de transporte UDP, onde não se tem confirmações de entrega de mensagem. Quem envia também não tem a obrigação de manter a mensagem armazenada para futuras retransmissões;
- **QoS 1 (*at least once*)** - Neste nível existe a confirmação de entrega de uma mensagem. Atende situações onde quem envia acaba gerando várias mensagens iguais possivelmente por um atraso na chegada de confirmação de recebimento. Neste caso, é garantido que uma delas terá o reconhecimento realizado. Existe um armazenamento da mensagem por parte de quem envia até a posterior confirmação;
- **QoS 2 (*exactly once*)** - Já este nível garante que a mensagem seja entregue exatamente uma vez, com envio de confirmações de recebimento e confirmações de recebimento de confirmações de recebimento, ou seja, existem confirmações nos dois sentidos, para tudo que é trafegado. Enquanto uma mensagem não é confirmada, ela é mantida. Este é um caso mais próximo do protocolo de transporte TCP.

É importante ressaltar que não existe um QoS melhor ou pior, na verdade, isto irá depender de cada cenário de aplicação do MQTT, da qualidade do link de comunicação, dos recursos disponíveis no seu sensor, etc. Ainda, é importante ressaltar também, que cada nível de QoS é negociado entre o cliente e o broker, não entre o publicador e o subscritor. Assim, é possível ter uma publicação em QoS 0 e uma subscrição em QoS 2, para um mesmo tópico (ALMEIDA, 2015).

2.2 Tecnologia da Informação

Segundo Mourtzis, Vlachou e Milas (2016), a hiper-conectividade dos vários sistemas envolvidos no processo produtivo, sobre os quais se baseia a Indústria 4.0, vem inevitavelmente transformando os dados industriais em Industrial Big Data, visto que máquinas e sensores produzem muito mais dados que uma pessoa é capaz de produzir. Porém, apesar de uma enorme quantidade de dados exigir mais espaço para armazenamento e maior poder de processamento, duas coisas que são facilitadas pela Cloud Computing, é ela quem também permite determinar padrões e correlacionar dados para a melhoria de processos industriais de forma muito mais precisa e eficaz. Desta forma, não tem como falar de I4.0 sem falar de Cloud Computing (LI; CHEN; SHANG, 2021).

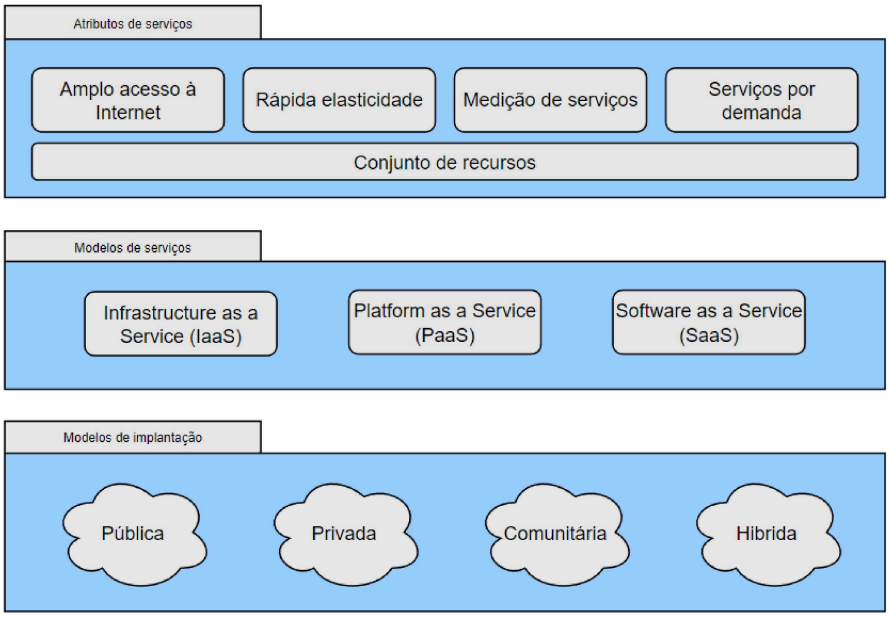
2.2.1 Cloud Computing

De forma sucinta, Cloud Computing se refere a um conceito tecnológico e a um modelo de negócio que permite o uso remoto de recursos computacionais através da conectividade com a internet. Permite o acesso a infraestrutura, software e informações através de qualquer dispositivo (computador, tablet, celular) que esteja conectado à internet e também permite o armazenamento de grandes quantidades de dados (Big Data). Essa capacidade é importante principalmente para armazenar os dados gerados durante todo o processo de produção. Da mesma forma, Cloud Computing reduz o investimento em recursos tecnológicos, permitindo que o espaço de armazenamento e capacidade de processamento sejam contratados sob demanda, o que proporciona flexibilidade, agilidade e adaptabilidade (THAMES; SCHAEFER, 2016; ALAM, 2021).

O modelo visual de definição de computação em nuvem definido por NIST (MELL; GRANCE et al., 2011) é apresentado na Figura 6. De acordo com o modelo estabelecido, a nuvem pode ser: pública, privada, comunitária ou híbrida. A nuvem pública é gerenciada por grandes empresas, onde o usuário contrata os serviços necessários para sua aplicação e os administradores da plataforma de Cloud Computing cuidam da manutenção, do escalonamento e da segurança das informações, de forma transparente para o usuário. Exemplos desse tipo de plataforma são AWS (Amazon Web Service), Google Cloud Platform, Microsoft Azure, Oracle Cloud, entre outras (MELL; GRANCE et al., 2011; CUNHA et al., 2016).

Na nuvem privada, o cliente é responsável pela administração e segurança dos dados e dos sistemas envolvidos. O acesso é feito por meio de uma rede privada virtual (VPN) e é utilizado por empresas que não desejam compartilhar espaço com outros usuários. A nuvem comunitária é semelhante a uma nuvem privada. Ela é fornecida a um grupo de organizações que possuem tipos semelhantes de requisitos com recursos adicionais. Pode existir dentro ou fora das instalações. Já a nuvem híbrida, envolve o uso tanto de nuvem

Figura 6 – Modelo visual de definição de computação em nuvem por NIST.



Fonte: Adaptado de Palanichamy et al. (2016), Mell, Grance et al. (2011).

pública quanto privada (MELL; GRANCE et al., 2011; CUNHA et al., 2016).

Além disso, existem 3 principais modelos de serviço em nuvem: Infraestrutura como Serviço (IaaS), Plataforma como Serviço (PaaS) e Software como Serviço (SaaS). No modelo IaaS, o consumidor não gerencia ou controla a infraestrutura de nuvem, mas tem controle sobre sistemas operacionais, armazenamento e aplicativos implantados; e controle possivelmente limitado de componentes de rede selecionados, como por exemplo, firewalls de hosts. O modelo PaaS fornece uma plataforma onde consumidores podem criar e executar seus aplicativos ou programas sem baixar e instalar outros softwares. Estes não precisam gerenciar ou controlar a infraestrutura, sistemas operacionais ou armazenamento, mas tem controle sobre os aplicativos implantados e possivelmente as definições de configuração (MELL; GRANCE et al., 2011; CUNHA et al., 2016).

Já no modelo SaaS, o consumidor usufrui do software sem gerenciar ou controlar a infraestrutura de nuvem subjacente, incluindo rede, servidores, sistemas operacionais, armazenamento ou mesmo recursos de aplicativos individuais, com exceção de possíveis configurações limitadas de aplicativos específicos do usuário (MELL; GRANCE et al., 2011; CUNHA et al., 2016).

Ainda de acordo com Cunha et al. (2016) e Mell, Grance et al. (2011), a computação em nuvem possui várias características que a tornam superior ao uso de hardware dedicado, como autoatendimento, rápida elasticidade, amplo acesso à rede, alocação de recursos sob demanda, pagamento por uso, entre outros. Além disso, o uso da nuvem descentraliza a informação para um armazenamento onde é possível acessar os dados de qualquer lugar,

possibilitando o acesso colaborativo e simultâneo às informações (ALAM, 2021).

Por conta das grandes vantagens que o uso da nuvem pode proporcionar, um dos principais objetivos do trabalho é a convergência dos dados para a mesma. Pensando nisso, a plataforma escolhida para o desenvolvimento da infraestrutura Cloud foi a AWS (Amazon Web Service), pois esta, além de possuir vários serviços de computação em nuvem que visam alta escalabilidade e disponibilidade, pelo fato de estar sendo a mais usada pelos profissionais de TI, possui uma maior quantidade de material para consulta se comparado a outras empresas de computação na nuvem, o que facilita bastante o desenvolvimento de novas aplicações.

2.2.1.1 Custo dos serviços utilizados na AWS

Aqui são descritos alguns custos dos serviços utilizados na AWS apesar do projeto ter sido desenvolvido durante o Free Tier e não ter gerado nenhum custo. O Free Tier da AWS está disponível por 12 meses a partir da data em que se cria uma conta na AWS. A seguir são apresentados os custos dos serviços organizados por categoria: Amazon EC2 e AWS Lambda - categoria Computação; AWS IoT - categoria Internet das Coisas; Amazon DynamoDB - categoria Banco de Dados; SNS - categoria Plataforma Móvel.

Amazon Elastic Compute Cloud - EC2

- ***Free Tier*** - Inclui 750 horas de instâncias Linux e Windows t2.micro (t3.micro para as regiões em que t2.micro não está disponível), todos os meses durante um ano.
- ***On-Demand*** - Com instâncias sob demanda, o usuário paga pela capacidade de computação por hora ou segundo, dependendo de quais instâncias estão sendo executadas. Não são necessários compromissos de longo prazo ou pagamentos adiantados. É possível aumentar ou diminuir sua capacidade de computação dependendo das demandas da aplicação e pagar apenas as taxas especificadas por hora para a instância usada.
- ***Spot Instances*** - As instâncias spot do Amazon EC2 permitem solicitar a capacidade de computação sobressalente do Amazon EC2 com até 90% de desconto no preço sob demanda.
- ***Saving Plans*** - Os Savings Plans são um modelo de preço flexível que oferece preços baixos no uso do EC2 e do Fargate, em troca de um compromisso com uma quantidade consistente de uso (medido em $/hora) por um período de um ou três anos.
- ***Dedicated Hosts*** - Um host dedicado é um servidor EC2 físico dedicado para seu uso. Hosts dedicados podem ajudar a reduzir custos, permitindo usar suas licenças

de software existentes vinculadas ao servidor, incluindo Windows Server, SQL Server e SUSE Linux Enterprise Server (sujeito aos termos de sua licença) e também podem ajudar a atender aos requisitos de conformidade.

- ***Per-Second Billing*** - Com o faturamento por segundo, o usuário paga apenas pelo que usar. O faturamento por segundo do EC2 remove o custo de minutos e segundos não utilizados de sua fatura. O usuário deve concentrar em melhorar seus aplicativos em vez de maximizar o uso por hora, especialmente para instâncias executadas em períodos de tempo irregulares, como desenvolvimento/teste, processamento de dados, análise, processamento em lote e aplicativos de jogos. O uso do EC2 é cobrado em incrementos de um segundo, com um mínimo de 60 segundos. Da mesma forma, o armazenamento provisionado para volumes do Amazon Elastic Block Store (EBS) será cobrado por incrementos de segundo, com um mínimo de 60 segundos.

AWS IoT Core

- ***Free Tier com AWS IoT Device Management*** - O nível gratuito do AWS IoT Device Management inclui 50 ações remotas por mês.
- ***Bulk Registration*** - O registro em massa permite registrar dispositivos conectados em massa. Os custos de registro em massa são baseados no número de dispositivos registrados.
- ***Things Registered*** - US$ 0,10 a cada 1.000 coisas registradas.
- ***Fleet Indexing and Search*** - US$ 2,25 a cada 1 milhão de atualizações de índice.

  US$ 0,05 a cada 10.000 consultas.
- ***Device Jobs*** - US$ 0,003 por ação para as primeiras 250.000 ações por mês.

  US$ 0,0015 por ação para mais de 250.000 ações por mês.

  Para mais de 12 milhões de ações por ano, é possível entrar em contato para negociar.
- ***Secure Tunneling*** - US$ 1,00 por túnel aberto.

AWS Lambda

O preço depende da quantidade de memória que você aloca para sua função. No modelo de recursos do AWS Lambda, você escolhe a quantidade de memória que deseja para sua função e recebe potência de CPU proporcional e outros recursos. Um aumento no tamanho da memória aciona um aumento equivalente na CPU disponível para sua função.

Por exemplo, 30 milhões de solicitações por mês (1 milhão de solicitações por dia) com o mínimo de memória alocada (0,125 GB) e com tempo de processamento de 1ms por função:

30 milhões de solicitações x 1ms = 30.000 segundos de processamento

0,125 GB x 30.000 segundos = 3.750 computação total (GB-s)

3.750 GB-s x 0,0000166667 USD = 0,06 USD

Custo total de computação = 0,062 USD mensais

30 milhões de solicitações x 0,0000002 USD = 6,00 USD

Custo total de solicitação = 6,00 USD mensais

0,062 USD + 6,00 USD = 6,062 USD

Custos do Lambda - sem nível gratuito (mensal): US$ 6,06

Amazon DynamoDB

O DynamoDB cobra pela leitura, gravação e armazenamento de dados em suas tabelas, juntamente com quaisquer recursos opcionais que podem ser habilitados. O DynamoDB tem dois modos de capacidade, que vêm com opções de cobrança específicas para processamento de leituras e gravações em suas tabelas: sob demanda e provisionado.

Custos DynamoDB para tabelas do tipo Standard e sob demanda, como foi utilizado no projeto:

- ***Free tier*** - 25 GB de armazenamento de dados para tabelas usando a classe de tabela Standard do DynamoDB;

  2,5 milhões de solicitações de leitura de fluxo do DynamoDB Streams;

  100 GB de transferência de dados para a Internet, agregados em todos os serviços e regiões da AWS.

- ***Tipo de taxa de transferência sob demanda*** - Unidades de solicitação de gravação (WRU): US$ 1,25 por milhão de unidades de solicitação de gravação;

  Unidades de solicitação de leitura (RRU): US$ 0,25 por milhão de unidades de solicitação de leitura.

- ***Data storage*** - Não é necessário provisionar armazenamento: o DynamoDB monitora o tamanho de suas tabelas continuamente para determinar suas cobranças de armazenamento. O preço do armazenamento de dados depende da sua classe de tabela.

  Os primeiros 25 GB armazenados por mês são gratuitos usando a classe de tabela padrão do DynamoDB, passado isso, é cobrado US$ 0,25/GB por mês.

- ***Backup on-demand*** - Warm Backup Storage (Backup quente, mais rápido para recuperar) US$0,10/GB por mês;

  Cold Backup Storage (Backup frio, mais lento para recuperar) US$0,03/GB por mês.

- ***Restoring a table*** - Restauração de Warm Backup Storage US$0,15 por GB;

  Restauração de Cold Backup Storage US$0,20 por GB.

Simple Notification Service - SNS

Com o Amazon Simple Notification Service (SNS), não há taxas iniciais, compromissos mínimos exigidos e contratos de longo prazo. O usuário paga apenas pelo que usa, com base no tipo de tópico usado. Como no projeto foram usados tópicos do tipo Standard, somente este será abordado.

A precificação de tópicos Standard é baseada no número de solicitações de API mensais feitas e no número de entregas para vários endpoints (o custo da entrega depende do tipo de endpoint).

- ***Mobile Push Notifications (SMS)*** - Free Tier: 1 milhão de notificações;

  Preço: US$0,50 por milhão de notificações.

- ***Email/Email-JSON*** - Free Tier: 1.000 notificações;

  Preço: US$2,00 a cada 100.000 notificações.

- ***HTTP/s*** - Free Tier: 100.000 notificações;

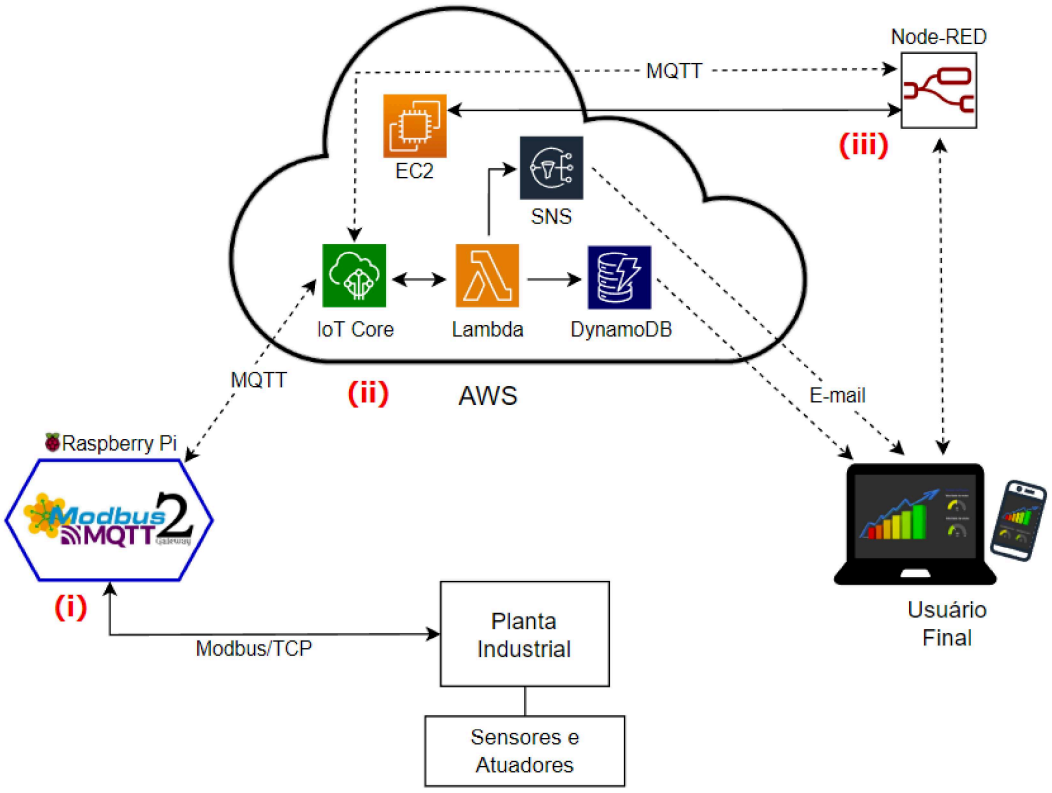
  Preço: US$0,60 por milhão de notificações.

3 Metodologia

Neste capítulo são descritos os materiais e métodos utilizados para o desenvolvimento do projeto proposto, bem como detalhes sobre as etapas do desenvolvimento.

Visando atingir os objetivos do trabalho, a arquitetura do sistema foi dividida em três módulos: (i) Modbus2MQTT Gateway, (ii) Infraestrutura Cloud e (iii) Interface de Controle e Monitoramento. A Figura 7 mostra a arquitetura do trabalho a ser desenvolvido.

Figura 7 – Arquitetura geral do sistema proposto.



Fonte: Autoria própria (2022).

3.1 Modbus2MQTT Gateway

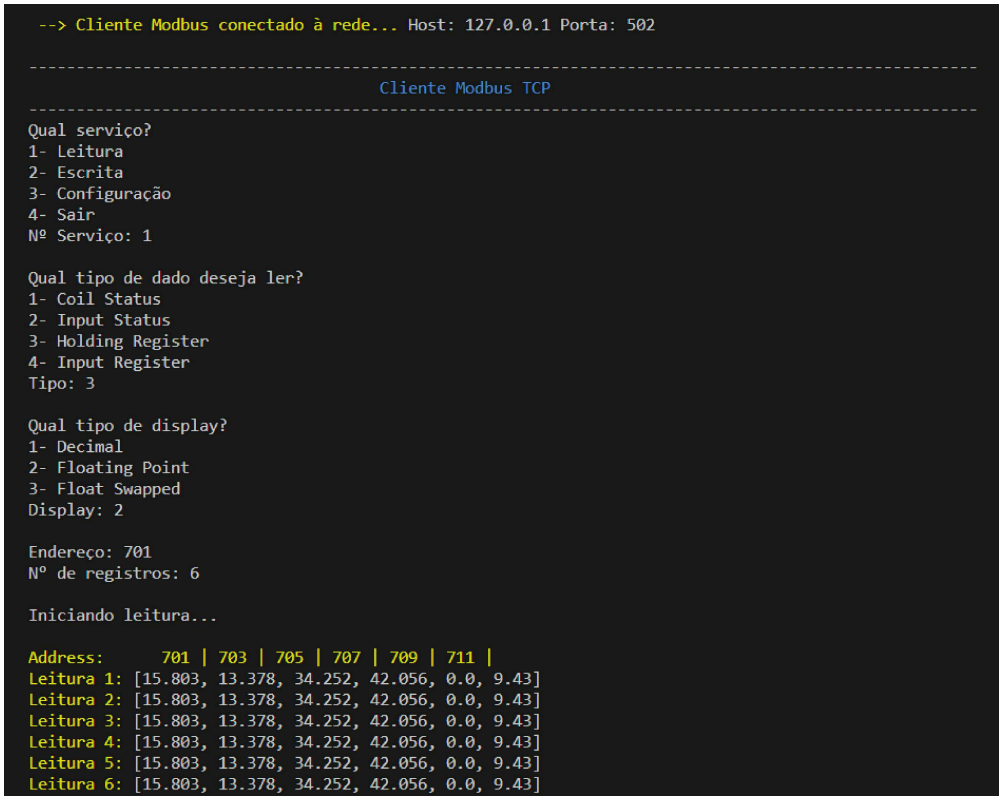
Este módulo, desenvolvido através da linguagem de programação Python, é responsável por permitir a troca de dados entre os equipamentos e a nuvem. Através deste gateway é possível que dispositivos da rede Modbus possam publicar e/ou subscrever em tópicos de interesse, transmitindo os dados de um lado para o outro via protocolo MQTT. É importante mencionar que todas as versões desta aplicação encontram-se disponíveis no GitHub em Lopes (2022). Este módulo foi dividido em 4 etapas: Desenvolvimento de

um cliente Modbus TCP; Desenvolvimento da comunicação entre os protocolos Modbus e MQTT; Desenvolvimento da interface gráfica do Modbus2MQTT; Testes.

3.1.1 Desenvolvimento de um Cliente Modbus TCP

Na primeira etapa foram realizados estudos sobre o protocolo de rede Modbus e também o desenvolvimento de uma aplicação com comunicação Modbus TCP. Para isto, foi utilizada uma biblioteca Python chamada "pyModbusTCP". Esta aplicação, chamada de "Cliente Modbus TCP", era capaz de fazer leituras e escritas em qualquer equipamento Modbus conectado à rede. É importante mencionar que o cliente Modbus consegue se conectar somente a um escravo (servidor) por vez. A Figura 8 mostra como era esta primeira versão do software desenvolvido.

Figura 8 – Cliente Modbus TCP - Primeira versão do software desenvolvido.



Fonte: Autoria própria (2022).

Ainda na Figura 8 é possível reparar que na opção de leitura Modbus (Serviço N° 1) existem as quatro primeiras *Function Codes* (FC 01 - Coil Status; FC 02 - Input Status; FC 03 - Holding Register e; FC 04 - Input Register) e 3 opções de display: Decimal (números inteiros); Floating Point (números racionais LSB - Least significant bit first); Float Swapped (números racionais MSB - Most significant bit first). Para isso, as leituras são tratadas por 3 diferentes funções no código, onde as 2 últimas leem 2 registros para calcular um valor flutuante, seguindo o padrão IEEE-754.

Para mais, no serviço de Escrita (N° 2) existem duas opções de escrita: FC 05 - Write Single Coil; FC 06 - Write Single Register. Em 'Configuração' (N° 3) é possível alterar as configurações da rede como endereço IP e porta.

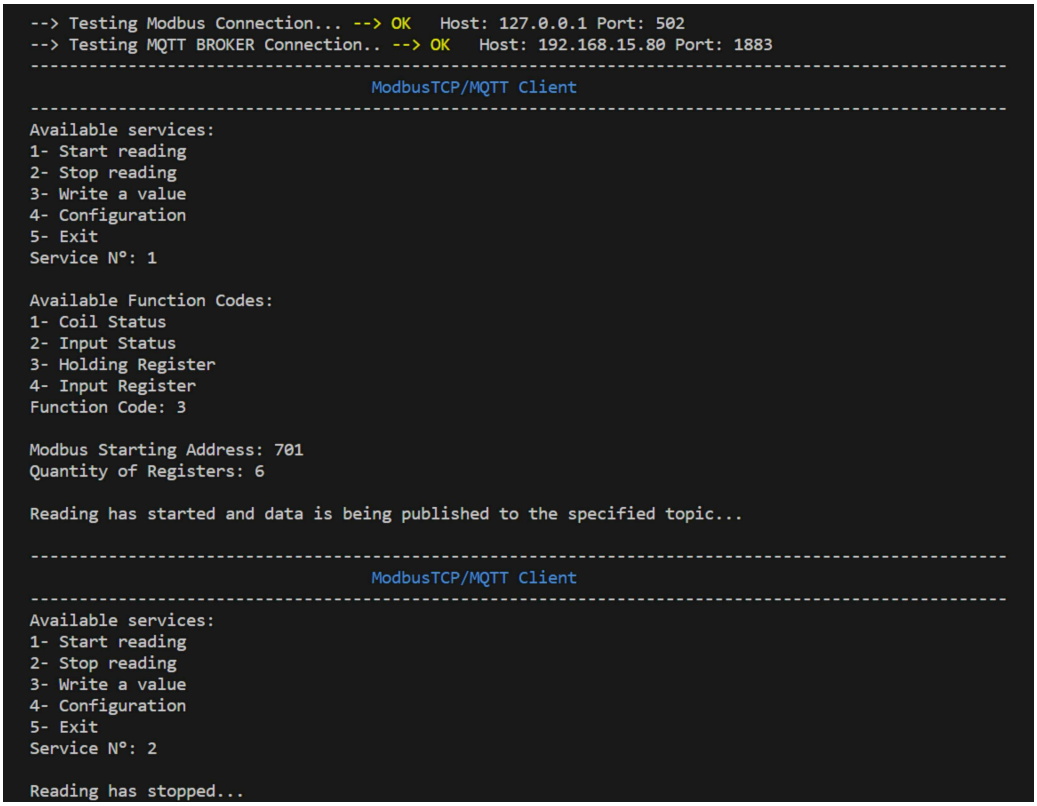
Vale ressaltar que nesta primeira versão, ainda não havia sido implementada a programação multi-tarefas, sendo assim, a leitura deveria ser interrompida pelo usuário para que este pudesse voltar ao menu principal.

3.1.2 Desenvolvimento da comunicação Modbus/MQTT

Já nesta etapa, foram realizados estudos sobre o protocolo MQTT, seguido da implementação de uma lógica de programação para a publicação dos dados obtidos através das leituras do cliente Modbus TCP para um MQTT Broker.

Primeiramente foi utilizada somente a biblioteca "paho-mqtt", posteriormente, foi implementado o SDK (Software Development Kit) para o AWS IoT Core, desta forma, a aplicação desenvolvida é capaz de fazer publicações MQTT em qualquer Broker com ou sem credenciais (usuário e senha), ou na nuvem da AWS, contando com certificações TLS (Transport Layer Security) para maior segurança. O AWS IoT Core é melhor detalhado na próxima seção (3.2 Infraestrutura Cloud). A Figura 9 mostra como era a cara da aplicação nesta segunda etapa de desenvolvimento.

Figura 9 – ModbusTCP/MQTT Client - Segunda versão do software desenvolvido.



Fonte: Autoria própria (2022).

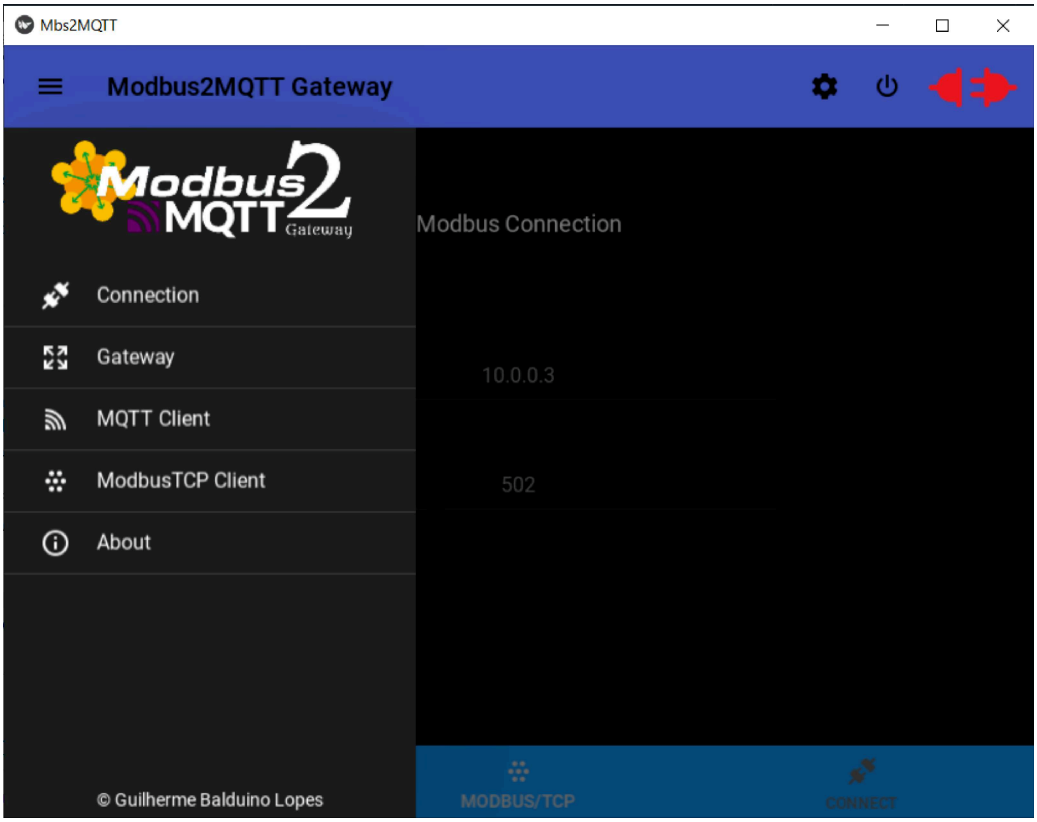
Nesta versão do software, também foi implementada a programação multi-tarefas (biblioteca *threading* do Python), ou seja, a aplicação já era capaz de fazer as leituras e publicações MQTT em background sem que o usuário ficasse com o terminal travado. Vale ressaltar também que, para maior alcance do projeto, a partir desta versão as aplicações começaram a ser desenvolvidas na língua inglesa.

3.1.3 Desenvolvimento da interface gráfica

Para o desenvolvimento da interface gráfica foi utilizada a biblioteca Kivy (e KivyMD - Kivy Material Design) na criação das janelas, botões e campos de entrada do usuário, além da utilização de bibliotecas mais comuns como "json" para manipulação de arquivos JSON (JavaScript Object Notation).

A Figura 10 mostra a tela inicial (tela de conexão Modbus e MQTT) da aplicação com o menu lateral aberto, onde é possível navegar pelas telas: Connection; Gateway; MQTT Client; ModbusTCP Client e; About (tela com breve descrição da aplicação).

Figura 10 – Modbus2MQTT Gateway com menu lateral.

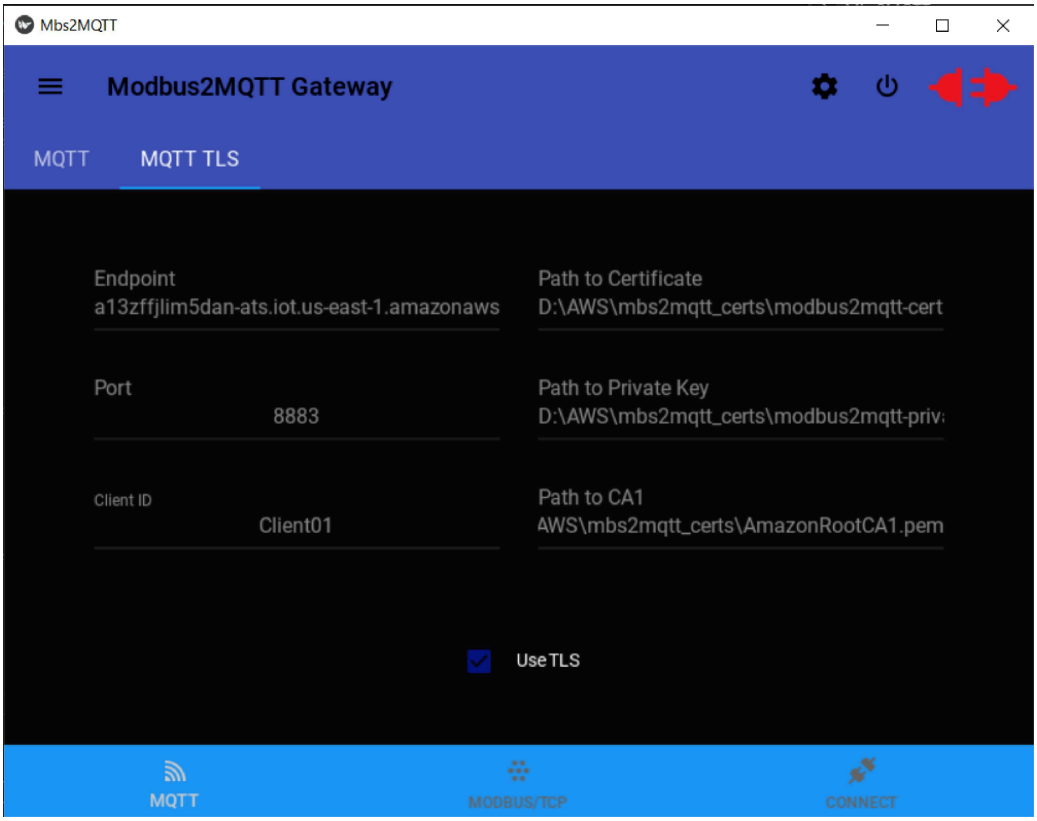


Fonte: Autoria própria (2022).

A primeira opção do menu é a própria tela inicial (Connection) onde é possível definir os parâmetros de conexão tanto Modbus quanto MQTT. A Figura 11 mostra esta tela de conexão na opção de configuração MQTT na aba que permite conexão com encriptação padrão TLS, onde o usuário deve prover o Endpoint (ou endereço/host do

Broker), a porta, a identificação do cliente e os caminhos para os certificados digitais que permitem a criptografia da comunicação. Para que seja possível utilizar este tipo de conexão a opção "Use TLS" deve estar selecionada, caso o contrário, o gateway tentará fazer uma conexão MQTT sem TLS.

Figura 11 – Tela de Conexão na aba de configuração MQTT TLS.



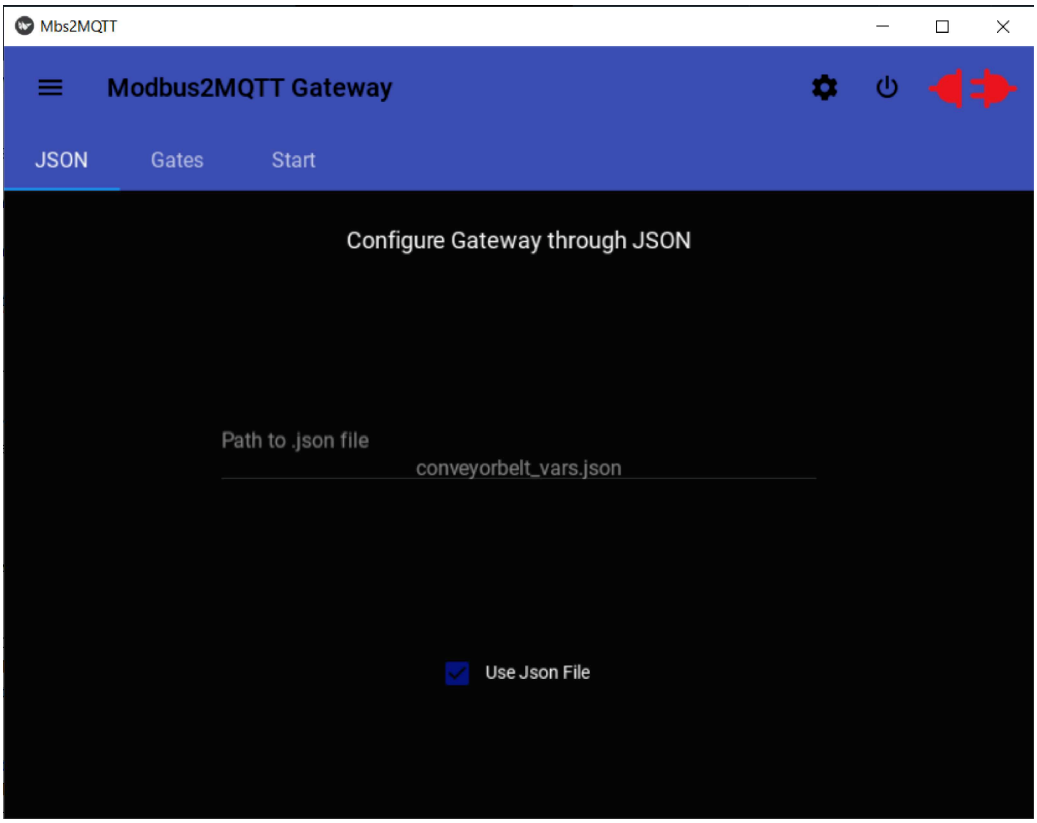
Fonte: Autoria própria (2022).

A Figura 12 apresenta a tela de configuração do Gateway na aba para parametrização através de arquivo JSON, enquanto que a Figura 13 mostra a estrutura do arquivo JSON elaborado para os testes iniciais da aplicação. É importante ressaltar que, para que o sistema funcione perfeitamente, os arquivos JSON a serem lidos pelo Gateway devem manter este mesmo padrão estrutural, onde:

- ***"System Description"*** - Uma breve descrição sobre o sistema a ser conectado.
- ***"Modbus2MQTT"*** - Contem a parametrização das variáveis a serem lidas através do protocolo Modbus e publicadas pelo MQTT. Cada variável deve ter o registro a ser lido (Address), a quantidade de registros (Length), as Permissões de leitura ou escrita (Rights), o tipo da variável (UINT16 para números decimais inteiros, F32 para Floating Points, Boolean para 1 ou 0, etc.), a unidade de medida do valor a ser lido e o tópico a ser publicado.

- ***"MQTT2Modbus"*** - Contem as variáveis que irão receber mensagens publicadas pelo MQTT e transformar estas em escritas na rede Modbus. Cada variável deve ter o registro a se escrever (Address), as Permissões de leitura ou escrita (Rights), o tipo da variável, a unidade de medida do valor e o tópico a qual o cliente fará uma subscrição.

Figura 12 – Tela de configuração de Gateway via arquivo JSON.

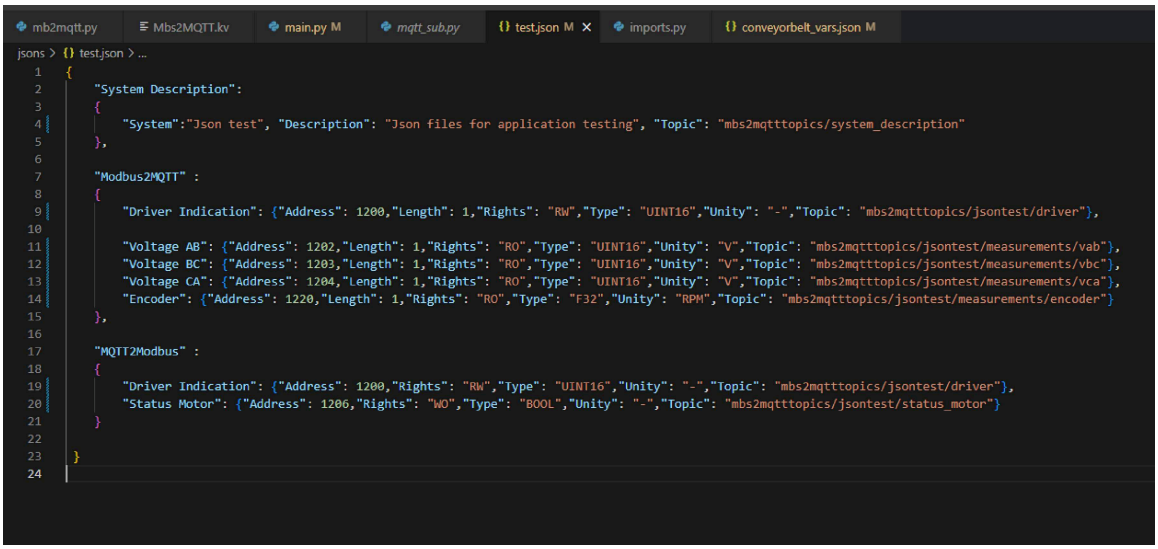


Fonte: Autoria própria (2022).

Já a Figura 14 mostra a aba "Gates" com opção de 20 configurações manuais (Gates de 1 a 21). Estes Gates manuais podem ser utilizados para atrelar um endereço Modbus a um tópico MQTT de forma independente às parametrizações via arquivo JSON. Para que estes sejam usados, a opção "Use this gate" deve estar selecionada em cada Gate a ser utilizado antes de iniciar o gateway na aba Start.

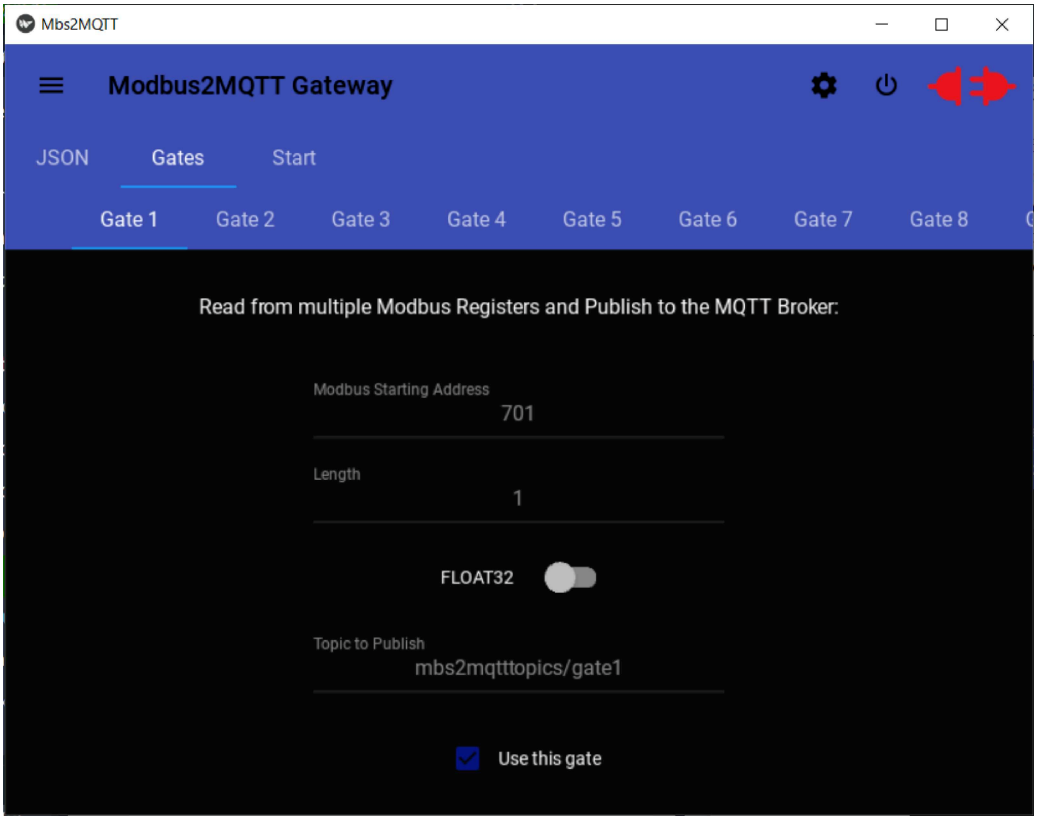
Além disso, é válido mencionar que ao clicar na engrenagem na barra superior o usuário tem a opção de iniciar o gateway (iniciar a troca de mensagens) sempre que o sistema for conectado (Fig. 15). Além disso, ao clicar no '**X** ' para fechar a aplicação, um pop-up é acionado para confirmar o encerramento (Fig. 16).

Figura 13 – Estrutura arquivo JSON.



Fonte: Autoria própria (2022).

Figura 14 – Tela de configuração de Gateway manual com 4 sub-abas.

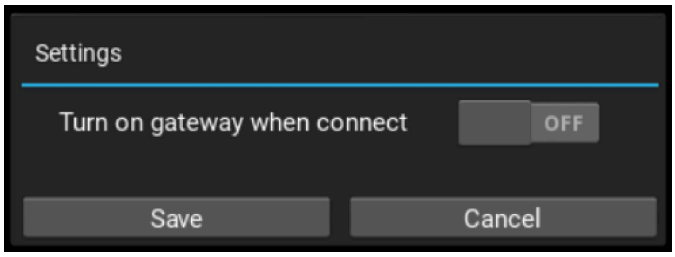


Fonte: Autoria própria (2022).

3.1.4 Testes de comunicação em ambiente simulado

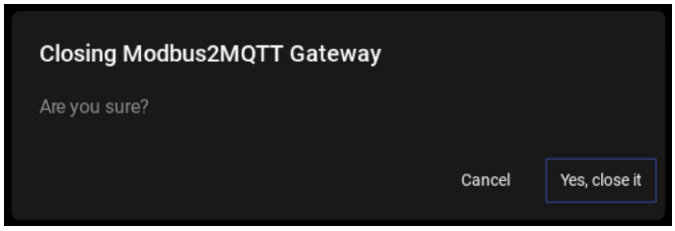
Durante o desenvolvimento da aplicação foram feitos vários testes de comunicação entre a rede Modbus e o MQTT Broker. Para estes testes preliminares foram utilizados servidores Modbus simulados através do Modsim32 (Fig. 17), um software simples e leve,

Figura 15 – Pop-up de configuração.



Fonte: Autoria própria (2022).

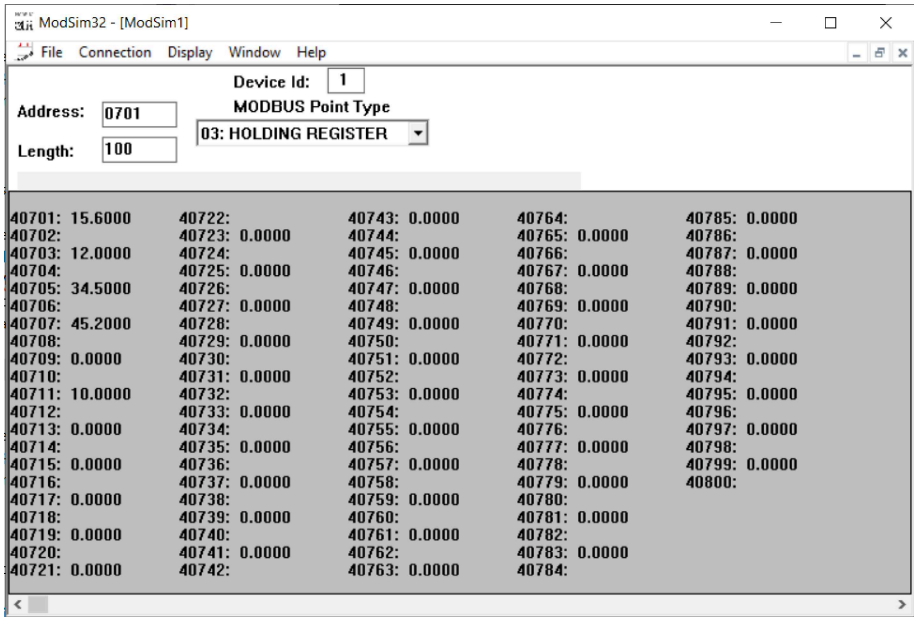
Figura 16 – Pop-up de confirmação de encerramento da aplicação.



Fonte: Autoria própria (2022).

com suporte à comunicação Serial e Ethernet, bastante utilizado para testes de novas aplicações.

Figura 17 – ModSim32 - Software para simulação de servidores/escravos Modbus.



Fonte: Autoria própria (2022).

Estes testes preliminares deram início a uma grande jornada de resolução de um bug no código. Notou-se que a aplicação Modbus2MQTT Gateway era capaz de realizar tanto publicações quanto subscrições em tópicos MQTT. Porém, a aplicação parava de funcionar quando as duas tarefas eram realizadas ao mesmo tempo. Foi então descoberto que isto acontecia pois o cliente MQTT iniciado na conexão, ficava ocupado durante uma

subscrição/publicação, o que gerava um *deadlock*, uma falha onde o código ficava travado esperando uma condição que nunca aconteceria.

Para resolver este problema, o código passou por uma nova modularização, onde agora é criado um objeto "Conexão" e uma classe "Cliente" cria objetos (capazes de utilizar funções responsáveis por publicações ou subscrições) que concorrem pelo mesmo objeto "Conexão". Desta forma, pode haver duas ou mais tarefas que se revesam para utilizar a conexão ativa.

3.2 Infraestrutura Cloud

Já este módulo, é responsável pelo desenvolvimento de toda a infraestrutura de computação em nuvem na plataforma da AWS, contando com o AWS IoT Core para recebimento dos dados publicados pelo Gateway, o banco de dados Amazon DynamoDB para armazenamento dos mesmos, o serviço sem servidor (*serveless*) Lambda responsável pelo processamento de mensagens que chegam ao IoT Core Broker, SNS (Simple Notification Service) para envio de e-mails, e a instância EC2 (Elastic Compute Cloud) onde está instalado o servidor Node-RED. Este foi dividido então em 5 etapas:

3.2.1 AWS IoT Core e Mosquitto Broker em EC2

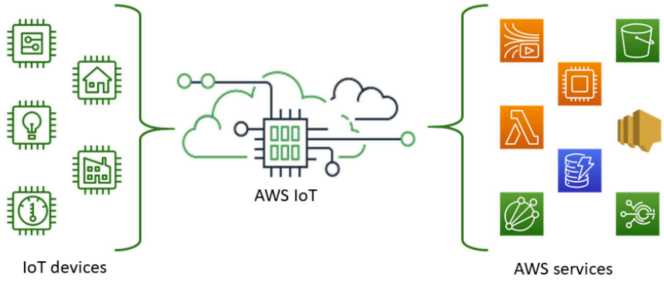
Primeiramente, foi criada uma instância EC2 com o intuito de servir como um MQTT Broker remoto, que estaria sempre ativo e de forma gratuita (1 ano grátis - Free Tier). Para isto, foi criada uma instancia do tipo "t2.micro", com um sistema operacional Ubuntu, em uma subnet pública. Em seguida, nesta instância foi instalado o Eclipse Mosquitto, um Broker MQTT de código aberto, muito simples de ser manejado e que possui autenticações como senha e usuário. Desta forma, foi possível ter um MQTT Broker sempre disponível para poder realizar os primeiros testes.

Posteriormente, para um melhor gerenciamento das publicações e maior facilidade de integração com outros serviços da AWS, foi utilizado o serviço IoT Core.

O AWS IoT Core pode oferecer suporte a bilhões de dispositivos e trilhões de mensagens. É capaz de processar e encaminhar essas mensagens para endpoints da AWS e para outros dispositivos de forma confiável e segura. Além disso, fornece softwares, bibliotecas e tutoriais que ajudam a integrar os dispositivos IoT na nuvem. Para mais, o Broker de mensagens do AWS IoT Core oferece suporte a dispositivos e clientes que usam protocolos MQTT, MQTT sobre WSS, HTTPS e LoRaWAN. A Figura 18 é uma representação ilustrativa do AWS IoT Core da própria AWS.

No AWS IoT Core foi então criado uma *Thing* (ou Coisa em português) chamada de "modbus2mqttThing", que nada mais é que uma representação digital do cliente dentro

Figura 18 – AWS IoT Core.



Fonte: Amazon Web Service - AWS (2022).

da AWS, contando com toda certificação para a máxima segurança das mensagens.

Além disso, no IoT Core foram criadas 3 regras *rules* destinadas ao processamento de mensagens que chegam ao Broker. Estas regras são:

- ***MbsMQTTConn_rule*** - Regra responsável pelo processamento das mensagens que chegam ao tópico de status de conexão do Modbus2MQTT Gateway;

  SQL: SELECT * FROM "mbs2mqtttopics/status/connection".

- ***MbsMQTTConveyor_rule*** - Regra responsável pelo processamento das mensagens que chegam ao tópico específico da planta a qual o sistema será implementado no Capítulo 4 e seus subtópicos;

  SQL: SELECT * FROM "mbs2mqtttopics/conveyorbelt_pub/#".

Cada regra foi configurada para enviar as mensagens recebidas diretamente ao serviço Lambda de forma assíncrona através de uma ação Lambda (*InvokeFunction*) do próprio IoT Core utilizando o método HTTPS POST.

3.2.2 AWS Lambda

O AWS Lambda é um serviço de computação que permite executar código sem provisionar ou gerenciar servidores (*serveless*). Ele executa código de diferentes linguagens em uma infraestrutura de computação de alta disponibilidade e realiza toda a administração dos recursos de computação, incluindo manutenção do servidor e do sistema operacional, provisionamento de capacidade, dimensionamento automático e registro em log.

O Lambda executa a função somente quando necessário e escala automaticamente, de algumas solicitações por dia a milhares por segundo. Além disso, o mesmo é cobrado apenas pelo tempo de computação que consome — não há cobrança quando o código não está em execução.

Foram criadas então 3 funções Lambdas, também disponíveis no GitHub Lopes (2022) utilizando a linguagem Python (versão 3.9):

- ***ConnStatusSNS_function*** - Responsável por enviar um e-mail através do serviço SNS (Simple Notification Service) sempre que o status de conexão do Gateway for alterado. O argumento *Subject* na linha 11 define qual será o assunto do e-mail: Modbus2MQTT Gateway Status Update. Já o argumento *Message* define qual será a mensagem contida no e-mail, no caso, "Modbus2MQTT Gateway status: (received_message)", onde "received_message" é a mensagem MQTT enviada pelo Modbus2MQTT Gateway ao tópico "mbs2mqtttopics/status/connection" toda vez que se conecta ou desconecta de uma planta.

- ***Save2DynamoDB_function*** - Responsável por escrever as mensagens recebidas em uma tabela do Amazon DynamoDB. Como Triggers (eventos responsáveis por chamar e executar a função), foram atrelados todos os tópicos da planta a ser implementada no Capítulo 4, desta forma, sempre que chegar uma mensagem MQTT aos mesmos, a função Lambda será executada e esta mensagem será salva no banco de dados.

- ***AlarmSNS_function*** - Responsável por enviar um e-mail através do serviço SNS sempre que algum valor de um tópico específico atingir um valor limite. Nesta função, por exemplo, sempre que o valor da temperatura da carcaça do motor superar os 70 °C, um alarme é enviado para o e-mail inscrito.

3.2.3 Amazon DynamoDB

A terceira etapa de desenvolvimento da infraestrutura cloud se tratou da criação de uma tabela no Amazon DynamoDB para armazenar as mensagens através da Função Lambda.

Segundo a própria Amazon Web Service (AWS), o DynamoDB é um serviço de banco de dados NoSQL totalmente gerenciado que fornece desempenho rápido e previsível com escalabilidade contínua sem que o usuário precise se preocupar com provisionamento de hardware, instalação, configuração, replicação, atualização de software ou dimensionamento de *cluster*. Ele também oferece criptografia em repouso, o que elimina a carga operacional e a complexidade envolvidas na proteção de dados confidenciais.

Além disso, o DynamoDB distribui automaticamente os dados e o tráfego das tabelas por um número suficiente de servidores para lidar com seus requisitos de taxa de transferência e armazenamento, mantendo um desempenho consistente e rápido. Todos os dados são armazenados em SSDs (Solid State Drive ou unidade de estado sólido) e são replicados automaticamente em várias zonas de disponibilidade em uma região da AWS,

fornecendo alta disponibilidade integrada e durabilidade de dados. Também é possível usar tabelas globais para manter as tabelas do DynamoDB sincronizadas entre diferentes regiões da AWS.

À vista disso, foi criada uma tabela do tipo DynamoDB Standard, com capacidades de escrita e leitura provisionadas por Auto Scaling (ou Escalonamento Automático, em português). Vale ressaltar que uma das vantagens de se trabalhar com bancos de dados NoSQL é que estes não precisam de colunas pré estabelecidas em um esquema de inter-relação de tabelas, suas chaves são geradas de forma flexível conforme as mensagens vão sendo salvas.

3.2.4 Simple Notification Service

O Amazon Simple Notification Service (Amazon SNS) é um serviço de mensagens da AWS totalmente gerenciado para a comunicação A2A (de aplicação para aplicação) e A2P (de aplicação para pessoa). A funcionalidade *pub/sub* de A2A fornece tópicos para sistemas de mensagens de alta taxa de transferência baseados em *push* e de muitos para muitos entre sistemas distribuídos, microsserviços e aplicações sem servidor orientadas por eventos, como Lambda. Usando tópicos do Amazon SNS, os sistemas editores podem repassar mensagens para um grande número de sistemas de assinantes, incluindo filas do Amazon SQS (Simple Queue Service), funções do AWS Lambda, *endpoints* HTTPS e o Amazon Kinesis Data Firehose para processamento paralelo. A funcionalidade A2P permite enviar mensagens para usuários em grande escala por SMS, *push* de dispositivos móveis e e-mail.

Foi criado então um tópico do tipo Standard no Amazon SNS chamado "modbus2mqtt_snstopic", com um e-mail (podendo também ser uma lista de e-mails) cadastrado como assinante. Ou seja, toda mensagem que chega através deste tópico é enviada para o e-mail assinante e, por ser do tipo Standard, a mensagem é enviada pelo menos uma vez e sem garantia de ordem de chegada.

3.2.5 Instância EC2 - Node-RED

Para se obter uma melhor experiência no painel de controle e monitoramento dos dados, foi criada uma instância do tipo t2.micro com o intuito de ser um servidor Node-RED. Desta forma, a aplicação e o ambiente de desenvolvimento dos Dashboards ficam sempre ativas para o usuário final através de um IP elástico público, caso o mesmo possua acesso ao serviço por meio de usuário e senha.

3.3 Interface de Controle e Monitoramento

Finalmente, o último módulo do desenvolvimento do sistema proposto cuida da parte da organização e apresentação dos dados em forma de Dashboards interativos, além das opções de controle da planta.

Foram criados então, através do ambiente de desenvolvimento do Node-RED, vários *nodes* (ou nós) de "mqtt-in", responsáveis por se inscreverem nos tópicos aos quais o Modbus2MQTT Gateway publica os dados. Após criados, os mesmos foram conectados à *nodes* de funções com o intuito de tratar as mensagens recebidas e depois à diferentes *nodes* de Dashboard para a apresentação das informações.

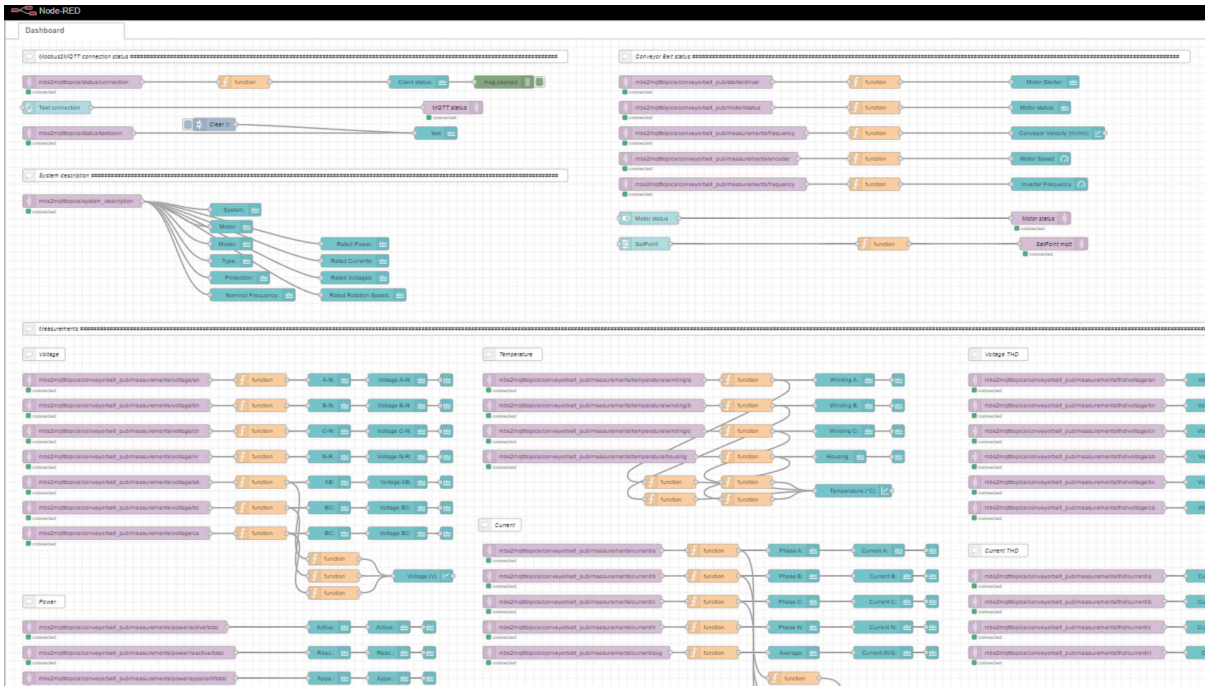
Posteriormente, foi integrada a parte de controle. Para isso, foram criados *nodes* de botões/*sliders* ligados a *nodes* de "mqtt-out". Desta forma, ao pressionar um botão por exemplo, uma mensagem é publicada a um tópico em que o Modbus2MQTT Gateway esteja inscrito e o mesmo transfere esta mensagem para a rede Modbus.

A Figura 19 apresenta uma visão geral do *flow* (fluxo) final no Node-RED, onde estes foram organizados por:

- **Modbus2MQTT connection status** - Onde através do tópico "mbs2mqtttopics/status/connection" o painel informa se o Modbus2MQTT Gateway está conectado ao Broker MQTT. Além disso, possui um botão para teste de conectividade que, ao pressionado, envia uma mensagem MQTT ao Gateway que o responde da mesma forma instantaneamente.
- **System description** - Apresenta uma breve descrição do sistema através do tópico "mbs2mqtttopics/system_description";
- **System status** - Já personalizado para a planta a ser conectada, apresenta algumas informações de status e opções de controle da mesma através de alguns sub-tópicos de "mbs2mqtttopics/conveyorbelt_pub/#" e tópico "mbs2mqtttopics/conveyorbelt_sub/" respectivamente;
- **Measurements** - Apresenta todas as informações e indicadores acerca da planta através dos sub-tópicos de "mbs2mqtttopics/conveyorbelt_pub/#".

Já a Figura 20 mostra com mais nitidez o fluxo de status de conexão com o gateway. Onde o primeiro fluxo printa a mensagem enviada pelo Modbus2MQTT através do tópico "mbs2mqtttopics/status/connection". O segundo envia uma mensagem para o tópico "mbs2mqtttopics/status" através de um clique no botão "Test connection". Já o terceiro fluxo recebe uma mensagem que o Modbus2MQTT envia para o tópico "mbs2mqtttopics/status/testconn" quando recebe a mensagem do tópico "mbs2mqtttopics/

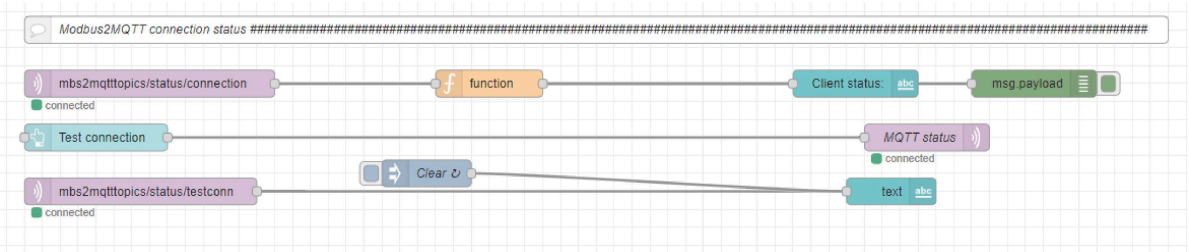
Figura 19 – Visão macro da estrutura do Flow desenvolvido no Node-RED.



Fonte: Autoria própria (2022).

status" enviada pelo fluxo anterior. O nó "Clear" limpa a mensagem recebida a cada 5 segundos para garantir que não seja antiga. Desta forma, é possível confirmar se as aplicações estão realmente se comunicando em tempo real. O arquivo Json do fluxo completo se encontra também na página do GitHub Lopes (2022).

Figura 20 – Flow para status de conexão do Node-RED com o Modbus2MQTT Gateway.



Fonte: Autoria própria (2022).

3.4 Recursos necessários

Como recursos necessários para o funcionamento do sistema pode ser citado a conta na plataforma de computação na nuvem da AWS, e algum dispositivo (PC, Smartphone ou Tablet) com acesso à internet para poder acessar a interface de controle e monitoramento através de um navegador Web. Ainda durante o desenvolvimento, o projeto pôde contar com vários equipamentos já dispostos pelo Laboratório de Automação, Sistemas Eletrônicos e Controle (LASEC), como CLPs e IEDs que se comunicam através do protocolo Modbus

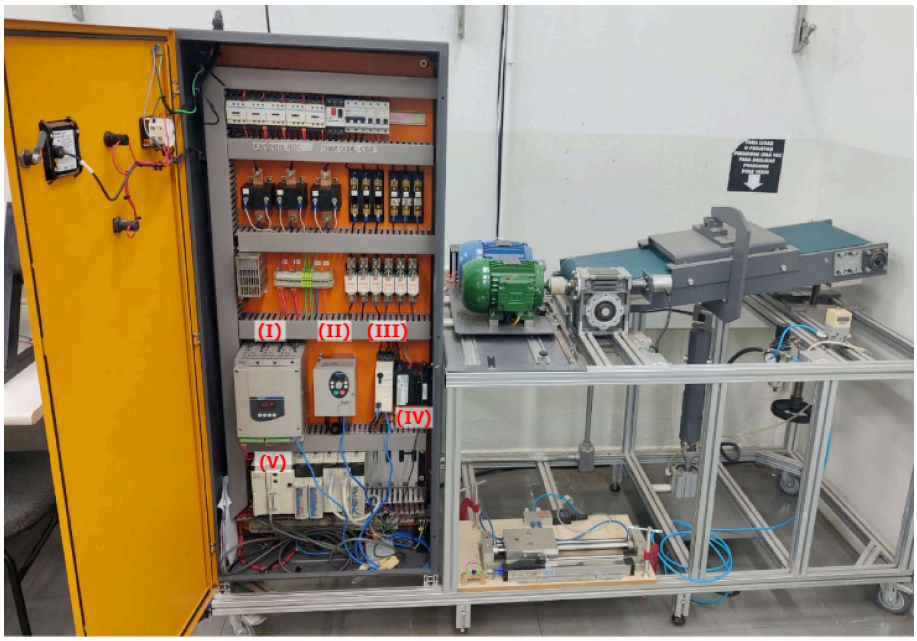
TCP (entre outros protocolos), e também uma planta didática de medição de consumo de uma esteira industrial, a qual é apresentada no Cap. 4.

4 Implementação em Sistema Real

Neste capítulo é apresentado a implementação do sistema em uma planta real, bem como os resultados obtidos. A Figura 21 mostra a planta didática de medição de consumo de uma esteira industrial provida pelo LASEC para a implementação e estudo de caso. Nesta Figura também é possível ver, ligados à esteira, 2 motores (um comum e um de alto rendimento), um simulador de carga e o painel de comandos, o qual conta com 3 tipos de partida, um multimedidor de qualidade de energia e um CLP, os quais são indicados na Figura 21 por:

- I - Telemecanique Altistart 48 Soft-start;
- II - Telemecanique Altivar 31 Inverter (Inversor de frequência);
- III - Telemecanique Tesys IUCB05BI (Partida direta);
- IV - Power Logic PM850UMG (Multimedidor de qualidade de energia);
- V - CLP Schneider TSX Premium.

Figura 21 – Planta didática de medição de consumo de uma esteira industrial.

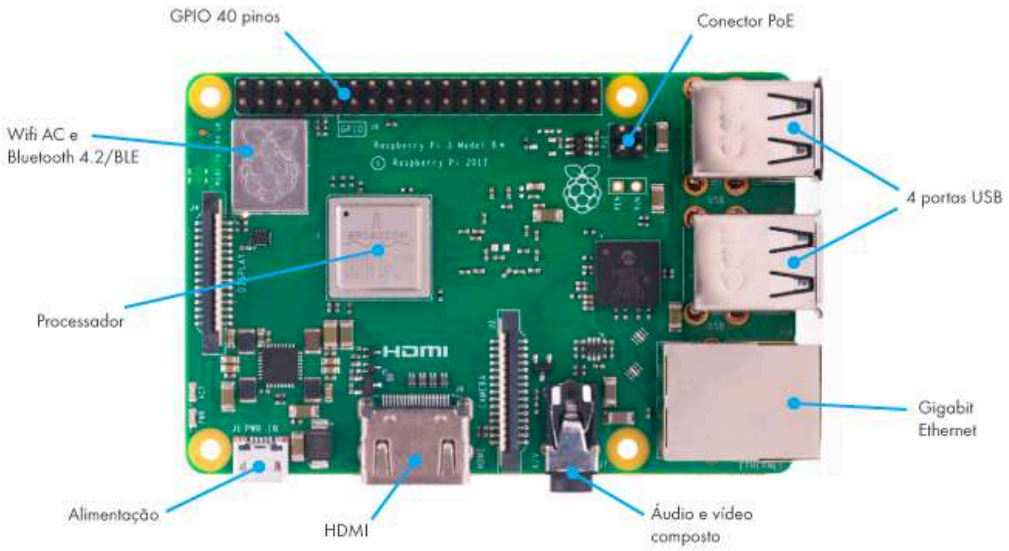


Fonte: Autoria própria (2022).

Os endereços dos registradores Modbus da planta se encontram disponíveis em uma tabela no Anexo A.

Além disso, para a implementação do projeto se faz necessário o uso de uma placa de desenvolvimento que seja capaz de se conectar tanto à Internet quanto à uma rede local, com o objetivo de hospedar o Modbus2MQTT Gateway. Por conseguinte, a tecnologia embarcada escolhida foi uma Raspberry Pi 3 B+ (Fig. 22), pois, além de entregar uma maior facilidade para execução de aplicações desenvolvidas em Python, também possui várias saídas para periféricos como tela, teclado e mouse, o que facilita a configuração/parametrização do Gateway.

Figura 22 – Raspberry Pi 3 Model B+.



Fonte: Blog Fazedores (2022).

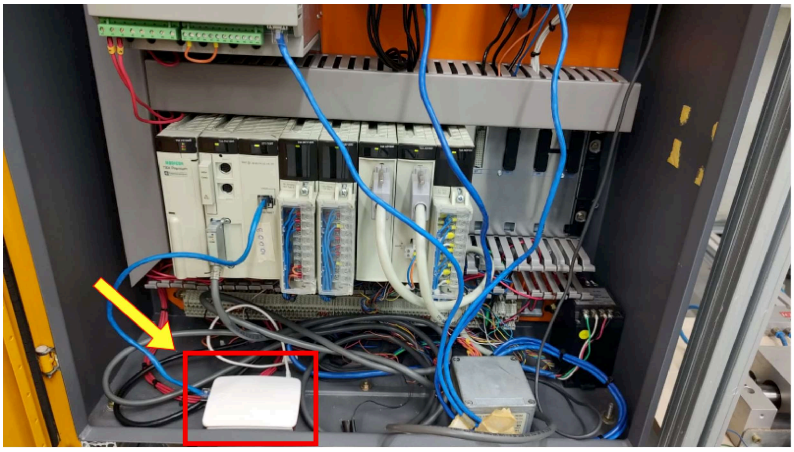
Algumas das características da Raspberry são:

- Processador Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz;
- 1GB LPDDR2 SDRAM;
- 2.4GHz e 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE;
- Conector de interface HDMI;
- 4 portas USB 2.0;
- Entrada de fonte DC micro USB 5V/2.5A.

A Figura 23 mostra uma foto da Raspberry Pi conectada ao CLP da Schneider através de um cabo Ethernet para que seja possível a comunicação Modbus TCP. Como o CLP possui endereço de rede 10.0.0.3/24, a Raspberry foi configurada no endereço 10.0.0.50/24. Desta forma, ambos estão alocados na mesma sub-rede, algo fundamental para o funcionamento da comunicação Modbus. Além disso, a Raspberry também foi

conectada ao Wi-Fi do laboratório para poder ter acesso à Internet, e assim, conseguir publicar as mensagens através do protocolo MQTT.

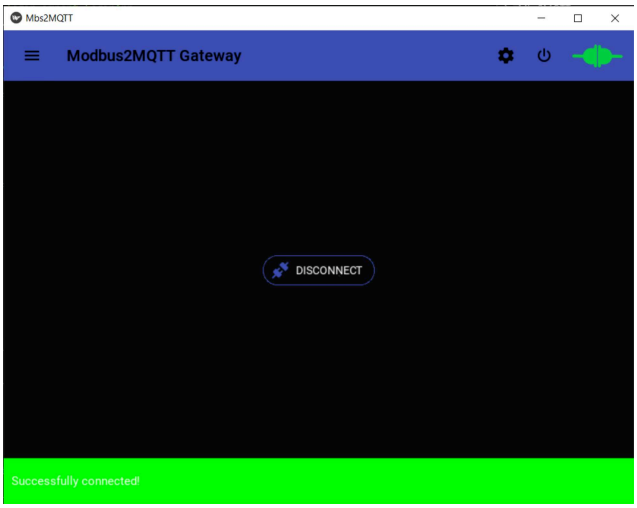
Figura 23 – Raspberry Pi conectada ao CLP da planta.



Fonte: Autoria própria (2022).

Após a Raspberry Pi estar devidamente conectada à planta e já com o App Modbus2MQTT Gateway instalado, é possível ativar a conexão do Gateway (Fig. 24) e iniciar sua comunicação através do arquivo JSON (Fig. 25).

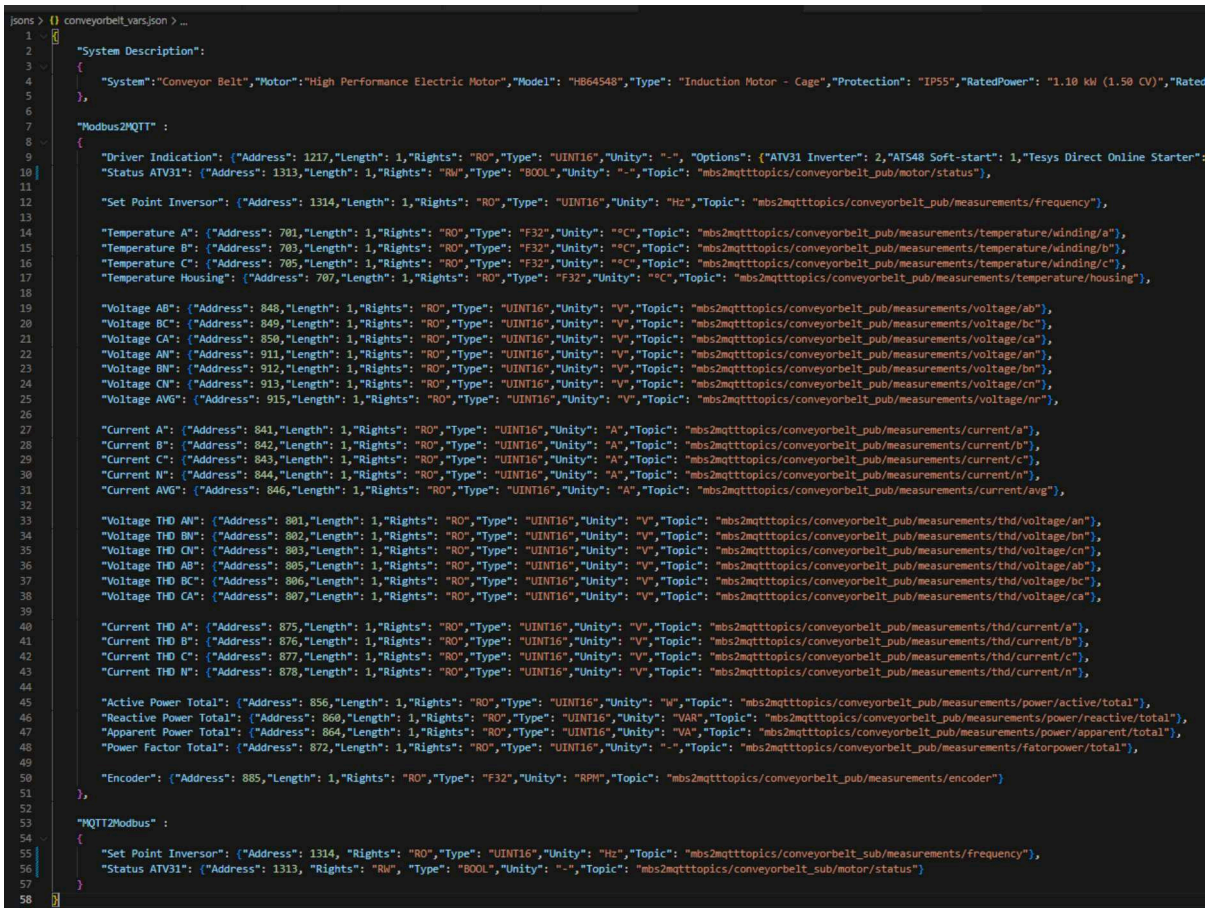
Figura 24 – Modbus2MQTT Gateway conectado.



Fonte: Autoria própria (2022).

Ao fazer isso, tanto o Node-RED quanto o banco de dados Amazon DynamoDB começam a receber os dados referentes à planta em tempo real.

Figura 25 – Print do arquivo JSON com parametrizações da planta.



Fonte: Autoria própria (2022).

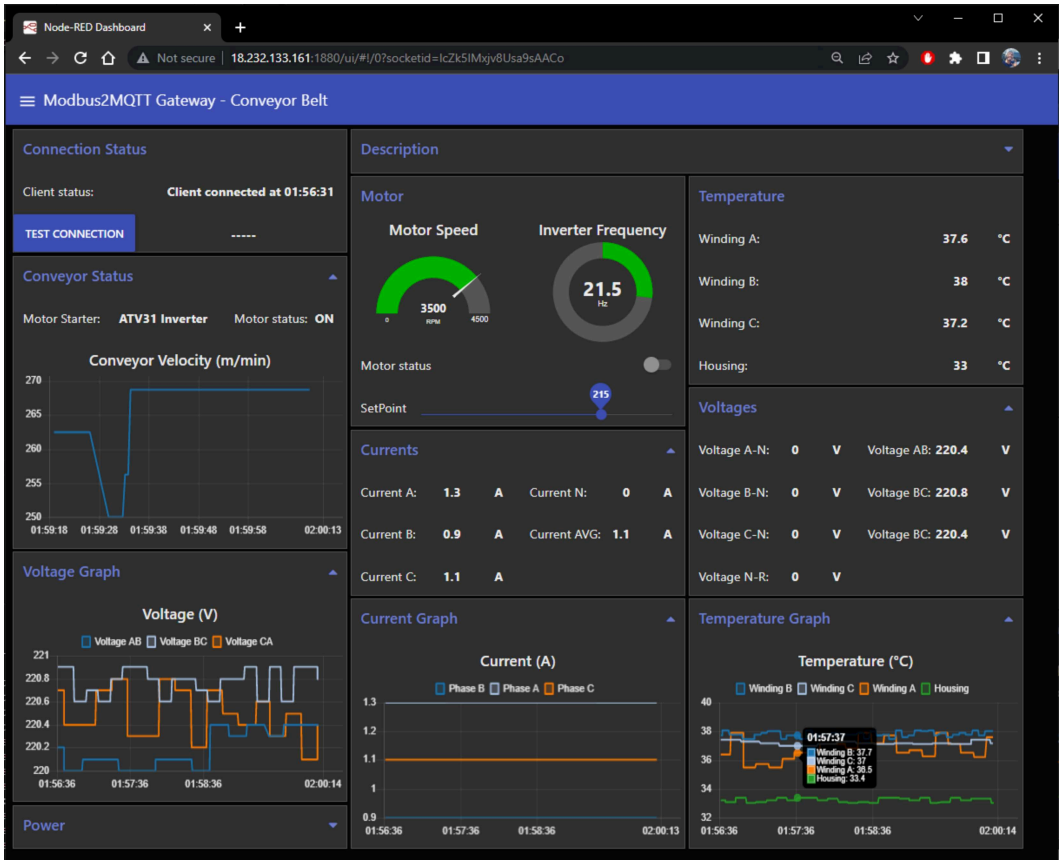
4.1 Resultados e discussões finais

4.1.1 Interface de Controle e Monitoramento

A Figura 26 mostra a interface de controle e monitoramento Node-RED apresentando os dados recebidos pelo sistema, contando com gráficos temporais para as grandezas elétricas e marcadores (Gauges) para velocidade e frequência do motor. Além disso, também são apresentados os controles de status do motor (para ligar/desligar o motor) e Setpoint (medida em Hz) da velocidade do mesmo.

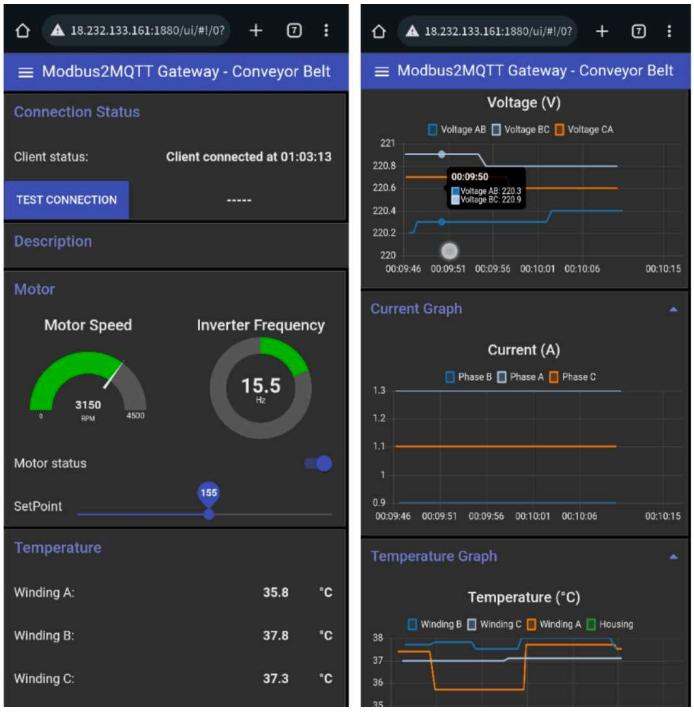
Já na Figura 27, com a interface aberta em um Smartphone, é possível notar que a aplicação é totalmente responsiva, ou seja, pode se adaptar a qualquer dispositivo, com qualquer formato de tela sem apresentar falhas.

Figura 26 – Interface Node-RED.



Fonte: Autor (2022).

Figura 27 – Dashboard Node-RED acessado através de um Smartphone.

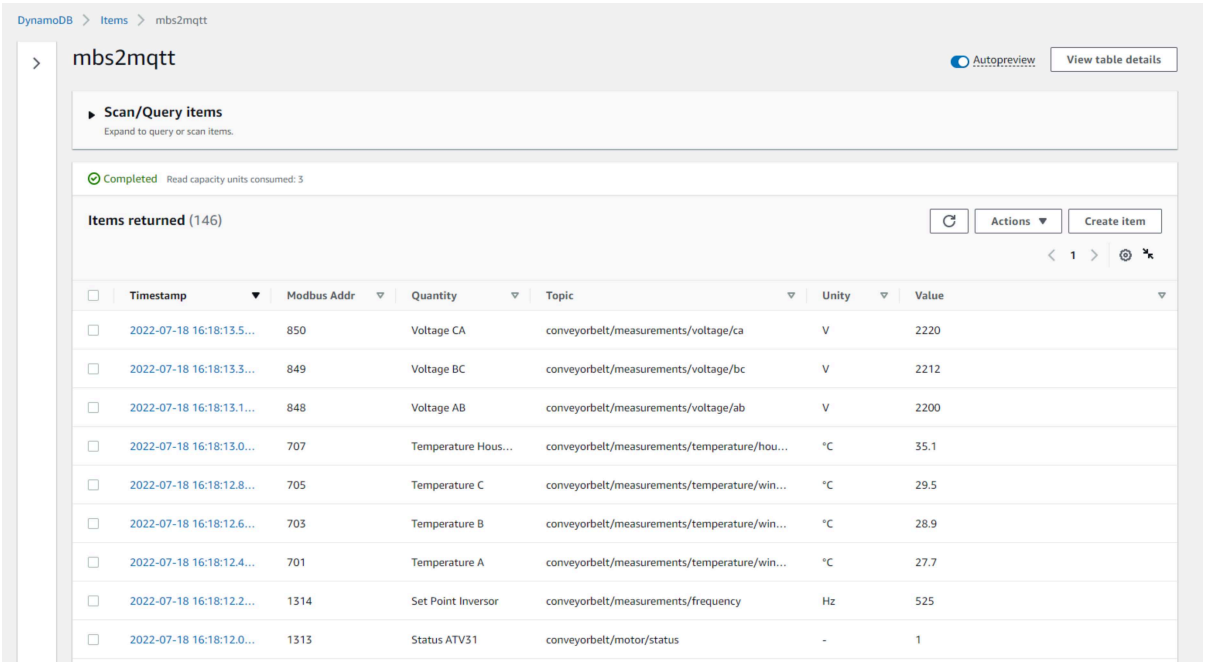


Fonte: Autor (2022).

4.1.2 Banco de Dados Amazon DynamoDB

Na Figura 28 é possível ver a tabela criada no banco de dados Amazon DynamoDB também recebendo os dados do sistema. Nela é possível observar os Timestamps (estampa de tempo) das leituras, os respectivos endereços Modbus, os nomes das variáveis medidas, os tópicos ao qual elas foram publicadas, as unidades de medida e os seus valores, assim como foi definido na função Lambda "Save2DynamoDB_function".

Figura 28 – Tabela no Amazon DynamoDB com os dados do sistema.



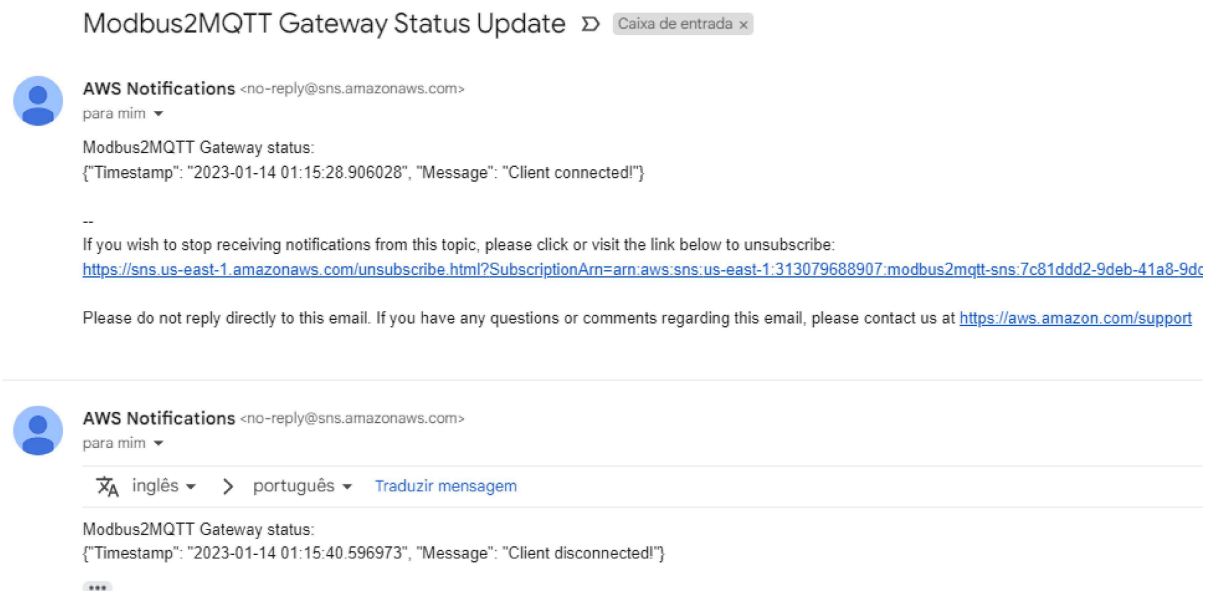
Fonte: Autor (2022).

4.1.3 Alarmes e Notificações

A Figura 29 mostra um print de 2 e-mails enviados através do tópico "modbus2mqtt_snstopic"do Amazon SNS pela função Lambda "ConnStatusSNS_function". Pode-se notar que o Assunto do e-mail e a mensagem são exatamente como foram definidas na função. Onde o conteúdo entre chaves ({}) se trata do parâmetro "received_message"da função.

Já a Figura 30, exibe o e-mail de alarme recebido através da função Lambda "AlarmSNS_function", a qual é executada quando o valor da temperatura supera os 70 °C.

Figura 29 – Email de notificação de Modbus2MQTT conectado e desconectado.



Fonte: Autor (2022).

Figura 30 – Email de notificação de alarme do Modbus2MQTT Gateway.

Fonte: Autor (2022).

5 Conclusão e trabalhos futuros

Este trabalho teve como meta principal o estudo e a integração de diferentes ferramentas voltadas à essa nova realidade conduzida pela quarta revolução industrial, que têm como foco auxiliar em tomadas de decisões que cooperem com uma gestão mais assertiva do consumo de energia nas indústrias e, consequentemente com o resultado final da produtividade e da lucratividade na produção industrial, além de auxiliar na sustentabilidade ambiental.

Foi desenvolvido um sistema de controle e monitoramento industrial de baixo custo, porém robusto, que através dos protocolos MQTT e Modbus, é responsável por controlar e monitorar remotamente equipamentos de uma planta industrial, enviando seus dados para a nuvem. Onde são armazenados de forma segura, escalável e redundante em um banco de dados NoSQL (Amazon DynamoDB) e também são tratados e transformados em informações visuais organizadas em Dashboards no Node-RED.

Com relação ao gateway Modbus2MQTT, ele foi desenvolvido em linguagem Python e pode ser utilizado tanto em um PC quanto em um sistema embarcado (Raspberry Pi, BeagleBone, etc.). Este foi desenvolvido no inicio do trabalho, onde passou por varias versões e melhorias durante o desenvolvimento. As quais são todas disponíveis na página do GitHub (LOPES, 2022). Ele foi validado em vários equipamentos presentes no LASEC, porém, é limitado a somente 1 escravo (servidor) Modbus por vez. Entretanto, é comum encontrar escravos Modbus que se comunicam com vários outros dispositivos, agregando seus dados internamente, como o CLP da planta utilizada na implementação no Cap. 4 - Implementação em Sistema Real. Durante o desenvolvimento deste, várias técnicas de programação tiveram que ser utilizadas, o que contribuiu para uma grande evolução profissional na área de desenvolvimento de softwares.

Com relação aos MQTT Brokers, responsáveis por receber as mensagens do sistema e fazer o intercâmbio das informações de um lado para o outro, foram utilizados dois: o AWS IoT Core e o Mosquitto Broker. Dois Brokers bastante robustos, relativamente simples de serem utilizados e com bastante material de apoio na internet.

Na implementação do sistema, os resultados apresentados evidenciam que o mesmo possui uma interface de controle e monitoramento que é disponível em qualquer lugar do mundo, de forma responsiva para qualquer dispositivo que possua um navegador com acesso à internet, e que a comunicação através do protocolo MQTT é bastante confiável e robusta para aplicações com um grande número de informações a serem transferidas de um ponto a outro.

Foi apresentado também que a plataforma da AWS garante a segurança dos dados

da aplicação e o gerenciamento dos recursos computacionais necessários por um preço acessível para soluções grandes em produção e sem custo para o primeiro ano de sistemas em desenvolvimento.

As etapas realizadas durante o processo de desenvolvimento do sistema foram todas apresentadas em detalhes com o intuito de servir como inspiração e referência para grupos de PeD que buscam desenvolver soluções para indústrias que desejam entrar para a era da Indústria 4.0. Além disso, alguns artigos científicos foram publicados desde o início do desenvolvimento do projeto.

Como proposta de trabalhos futuros serão estudadas formas de fazer com que o Modbus2MQTT Gateway consiga se conectar com vários dispositivos Modbus ao mesmo tempo, para que não seja necessário um dispositivo capaz de agregar as informações de todos os outros dispositivos.

Referências

ALAM, T. Cloud computing and its role in the information technology. *IAIC Transactions on Sustainable Digital Innovation (ITSDI)*, v. 1, p. 108–115, 2021. Citado 2 vezes nas páginas 27 e 29.

ALIMI, O. A. et al. A review of research works on supervised learning algorithms for scada intrusion detection and classification. *Sustainability*, MDPI, v. 13, n. 17, p. 9597, 2021. Citado na página 20.

ALMEIDA, M. B. Mqtt - protocolos para iot. 2015. Disponível em: <https://www.embarcados.com.br/mqtt-protocolos-para-iot/>. Acesso em: Janeiro de 2022. Citado 3 vezes nas páginas 24, 25 e 26.

ARIS, L. Indústria 4.0 depende da convergência entre ot e ti para avançar no brasil. 2020. Disponível em: <https://www.industria40.ind.br/artigo/20090-industria-40-depende-da-convergencia-entre-ot-e-ti-para-avancar-no-brasil>. Acesso em: Dezembro de 2021. Citado na página 18.

BHATTACHARJEE, S. The critical role of gateways in iot. 2019. Disponível em: <https://br.mouser.com/blog/blog/blog/critical-role-of-gateways-iot>. Acesso em: Fevereiro de 2022. Citado na página 19.

CARVALHO, N. et al. Manufacturing in the fourth industrial revolution: A positive prospect in sustainable manufacturing. *Procedia Manufacturing*, Elsevier, v. 21, p. 671–678, 2018. Citado 2 vezes nas páginas 15 e 18.

CORREA, R. P. d. S. et al. Solução de monitoramento de usinas fotovoltaicas utilizando gateway modbus/mqtt. Universidade Federal de Uberlândia, 2019. Citado 3 vezes nas páginas 20, 22 e 24.

CUNHA, M. J. da et al. Proposal for an iot architecture in industrial processes. *2016 12th IEEE International Conference on Industry Applications (INDUSCON)*, p. 1–7, 2016. Citado 3 vezes nas páginas 14, 27 e 28.

ENTENDA por que e como alinhar KPI's na gestão comercial. (Sem autor). 2017. Disponível em: <https://farolbi.com.br/importancia-de-alinhar-kpis/>. Acesso em: Fevereiro de 2022. Citado na página 14.

KELLER, A. L. Internet das coisas aplicada à indústria: dispositivo para interoperabilidade de redes industriais. Universidade do Vale do Rio dos Sinos, 2017. Citado 6 vezes nas páginas 21, 22, 23, 24, 25 e 26.

LASI, H. et al. Industry 4.0. *Business & information systems engineering*, Springer, v. 6, n. 4, p. 239–242, 2014. Citado na página 18.

LI, C.; CHEN, Y.; SHANG, Y. A review of industrial big data for decision making in intelligent manufacturing. *Engineering Science and Technology, an International Journal*, Elsevier, 2021. Citado na página 27.

LOPES, G. B. Repositórios github - @guilhermemikin. 2022. Disponível em: <https://github.com/GuilhermeMikin>. Citado 4 vezes nas páginas 33, 43, 46 e 55.

MAZUR, D. C.; KAY, J. A.; KREITER, J. H. Benefits of iec 61850 standard for power monitoring and management systems in forest products industries. In: IEEE. *Conference Record of 2013 Annual IEEE Pulp and Paper Industry Technical Conference (PPIC)*. [S.l.], 2013. p. 69–75. Citado na página 19.

MELL, P.; GRANCE, T. et al. The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory, National . . . , 2011. Citado 2 vezes nas páginas 27 e 28.

METZGER, P. What is a scada system and how does it work? 2022. Disponível em: <https://www.onlogic.com/company/io-hub/what-is-a-scada-system-and-how-does-it-work/>. Acesso em: Dezembro de 2021. Citado na página 20.

MOORE-INDUSTRIES. Using modbus for process control and automation. 2021. Disponível em: <https://www.miinet.com/white-paper.html>. Acesso em: Janeiro de 2022. Citado 3 vezes nas páginas 21, 22 e 24.

MOURTZIS, D.; VLACHOU, E.; MILAS, N. Industrial big data as a result of iot adoption in manufacturing. *Procedia CIRP*, v. 55, p. 290–295, 2016. ISSN 2212-8271. 5th CIRP Global Web Conference - Research and Innovation for Future Production (CIRPe 2016). Disponível em: <https://www.sciencedirect.com/science/article/pii/S2212827116307880>. Citado na página 27.

PALANICHAMY, N. et al. Cloud computing for energy management in smart grid - an application survey. *IOP Conference Series: Materials Science and Engineering*, v. 121, p. 012010, 03 2016. Citado na página 28.

PIVOTO, D. G. et al. Cyber-physical systems architectures for industrial internet of things applications in industry 4.0: A literature review. *Journal of Manufacturing Systems*, Elsevier, v. 58, p. 176–192, 2021. Citado 2 vezes nas páginas 14 e 18.

QUAIS são os tipos de redes industriais? (Sem autor). 2018. Disponível em: <http://blog.murrelektronik.com.br/quais-sao-os-tipos-de-redes-industriais/>. Acesso em: Abril de 2022. Citado na página 15.

REYNDERS, D.; MACKAY, S.; WRIGHT, E. *Practical industrial data communications: best practice techniques*. [S.l.]: Elsevier, 2004. Citado na página 18.

SEBRAE; CNI. 7º congresso brasileiro de inovação da indústria. In: *Mobilização Empresarial pela Inovação (MEI)*. [S.l.]: Portal da Indústria, 2018. (Estudos e prospectivas, v. 2018). Citado na página 15.

SILVA, F. C. S. Modelos cliente-servidor do modbus tcp e publish-subscribe do mqtt: quando utilizar cada um deles e quais suas vantagens e desvantagens? 2019. Disponível em: <https://www.novus.com.br/site/default.asp?Idioma=55&TroncoID=053663&SecaoID=0&SubsecaoID=0&Template=../artigosnoticias/user_exibir.asp&ID=618088>. Acesso em: Janeiro de 2022. Citado 2 vezes nas páginas 19 e 24.

SILVEIRA, C. B. Saiba tudo sobre o protocolo modbus. 2016. Disponível em: <https://www.citisystems.com.br/modbus/>. Acesso em: Novembro de 2021. Citado na página 22.

THAMES, L.; SCHAEFER, D. Software-defined cloud manufacturing for industry 4.0. *Procedia cirp*, Elsevier, v. 52, p. 12–17, 2016. Citado na página 27.

THE-MODBUS-ORGANIZATION. Modbus organization. 2022. Disponível em: <http://www.modbus.org>. Acesso em: Julho de 2022. Citado 4 vezes nas páginas 21, 22, 23 e 24.

VENTURELLI, M. Eficiência energética 4.0. 2020. Disponível em: <https://marcioventurelli.com/2020/06/26/eficiencia-energetica-4-0/>. Acesso em: Novembro de 2021. Citado na página 15.

VILLARIM, A. W. R.; SOUZA, C. P. de; BAIOCCHI, O. Desenvolvimento de um sistema iot de baixo custo para monitoramento de equipamentos de plantas industriais. *ENCONTRO INSTITUCIONAL DA PÓS-GRADUAÇÃO UFPB: Internacionalização da UFPB: diversidade*, v. 2, p. 297, 2021. Citado 3 vezes nas páginas 15, 16 e 18.

ZAGONEL. Qual é a relação entre a 'indústria 4.0' e a 'eficiência energética na indústria'? 2021. Disponível em: <https://www.zagonel.com.br/iluminacao/blog/show/qual-e-a-relacao-entre-a-industria-40-e-a-eficiencia-energetica-na-industria-14/>. Acesso em: Dezembro de 2021. Citado na página 18.

ANEXO A – Tabela Modbus da Planta didática para implementação

Planta didática de medição de qualidade de energia de uma esteira transportadora de cargas			
ID	**Tag Name**	**Modbus Address**	**Comment**
216	es.indica_driver	1217	1, 2 ou 3
324	es.sel_driver	1325	1, 2 ou 3
319	es.tesys	1320	
332	es.sel_pid	1333	
314	es.atv31_acc	1315	
315	es.atv31_dcc	1316	
317	es.ats48_acc	1318	
318	es.ats48_dcc	1319	
313	es.atv31_velocidade	1314	Set-point 0 a 720 (ou 72Hz)
312	es.atv31	1313	0 ou 1
316	es.ats48	1317	0 ou 1
319	es.tesys	1320	0 ou 1
0	es.temp_r	701	Floatpoint
2	es.temp_s	703	Floatpoint
4	es.temp_t	705	Floatpoint
6	es.temp_carc	707	Floatpoint
10	es.le_carga	711	
24	es.esteira	725	
114	es.mv_le	815	
8	es.tipo_motor	709	
22	es.status_pid	723	
47	es.tensao_rs	848	UINT16
48	es.tensao_st	849	UINT16
49	es.tensao_tr	850	UINT16
40	es.corrente_r	841	UINT16
41	es.corrente_s	842	UINT16
42	es.corrente_t	843	UINT16
43	es.corrente_n	844	UINT16
45	es.corrente_media	846	UINT16
52	es.ativa_r	853	UINT16
53	es.ativa_s	854	UINT16
54	es.ativa_t	855	UINT16
55	es.ativa_total	856	UINT16
68	es.fp_r	869	UINT16
69	es.fp_s	870	UINT16
70	es.fp_t	871	UINT16
71	es.fp_total	872	UINT16

60	es.aparente_r	861	UINT16
61	es.aparente_s	862	UINT16
62	es.aparente_t	863	UINT16
63	es.aparente_total	864	UINT16
56	es.reativa_r	857	UINT16
57	es.reativa_s	858	UINT16
58	es.reativa_t	859	UINT16
59	es.reativa_total	860	UINT16
86	es.status_ats48	887	UINT16
88	es.status_atv31	889	UINT16
90	es.status_tesys	891	UINT16
4	es.thd_tensao_rs	805	UINT16
5	es.thd_tensao_st	806	UINT16
6	es.thd_tensao_tr	807	UINT16
0	es.thd_tensao_rn	801	UINT16
1	es.thd_tensao_sn	802	UINT16
2	es.thd_tensao_tn	803	UINT16
74	es.thd_corrente_r	875	UINT16
75	es.thd_corrente_s	876	UINT16
76	es.thd_corrente_t	877	UINT16
77	es.thd_corrente_n	878	UINT16
92	es.valor1	893	UINT16
93	es.angulo1	894	UINT16
94	es.valor2	895	UINT16
95	es.angulo2	896	UINT16
96	es.valor3	897	UINT16
97	es.angulo3	898	UINT16
98	es.valor4	899	UINT16
99	es.angulo4	900	UINT16
100	es.valor5	901	UINT16
101	es.angulo5	902	UINT16
30	es.frequencia	831	UINT16
4	es.encoder	885	UINT16
2	es.carga	1303	UINT16
4	es.p	1305	UINT16
6	es.i	1307	UINT16
8	es.d	1309	UINT16
10	es.mv_escreve	1311	UINT16
4	es.demanda_anterior	1205	UINT16
5	es.demanda_atual	1206	UINT16
6	es.demanda_media	1207	UINT16
7	es.demanda_pico	1208	UINT16
8	es.demanda_prevista	1209	UINT16
10	es.energia_ativa	1211	UINT16
12	es.energia_reativa	1213	UINT16
14	es.energia_aparente	1215	UINT16

10	es.tensao_an	911	UINT16
11	es.tensao_bn	912	UINT16
12	es.tensao_cn	913	UINT16
13	es.tensao_nr	914	UINT16
14	es.tensao_media_fases	915	UINT16
26	es.rms_fundamental_fase_a	927	UINT16
27	es.angulo_fundamental_fase_a	928	UINT16
28	es.rms_fundamental_fase_b	929	UINT16
29	es.angulo_fundamental_fase_b	930	UINT16
30	es.rms_fundamental_fase_c	931	UINT16
31	es.angulo_fundamental_fase_c	932	UINT16
32	es.rms_fundamental_neutro	933	UINT16
33	es.angulo_fundamental_neutro	934	UINT16
40	es.rms_fundamental_tensao_a_n	941	UINT16
41	es.angulo_fundamental_tensao_a_n	942	UINT16
42	es.rms_fundamental_tensao_b_n	943	UINT16
43	es.angulo_fundamental_tensao_b_n	944	UINT16
44	es.rms_fundamental_tensao_c_n	945	UINT16
45	es.angulo_fundamental_tensao_c_n	946	UINT16
50	es.corrente_seq_pos_magnitude	951	UINT16
51	es.corrente_seq_pos_angulo	952	UINT16
52	es.corrente_seq_neg_magnitude	953	UINT16
53	es.corrente_seq_neg_angulo	954	UINT16
54	es.corrente_seq_zero_magnitude	955	UINT16
55	es.corrente_seq_zero_angulo	956	UINT16
56	es.tensao_seq_pos_magnitude	957	UINT16
57	es.tensao_seq_pos_angulo	958	UINT16
58	es.tensao_seq_neg_magnitude	959	UINT16
59	es.tensao_seq_neg_angulo	960	UINT16
60	es.tensao_seq_zero_magnitude	961	UINT16
61	es.tensao_seq_zero_angulo	962	UINT16
64	es.corrente_fator_desequilibrio	965	Floatpoint
65	es.tensao_fator_desequillibrio	966	Floatpoint
67	es.fp_fundamental	968	Floatpoint

Os drivers ligados ao motor e definidos pelos Ids 216 e 324 são:

Driver	Modbus Value
ats48 soft-start	1
atv31 inversor	2
tesys p. direta	3