UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS GRADUAÇÃO EM ENGENHARIA ELÉTRICA

WOLDSON LEONNE PEREIRA GOMES

SISTEMA DE AUTOMAÇÃO IOT PARA GESTÃO DE ATIVOS NO CENÁRIO DE INDÚSTRIA 4.0

DM: 21/23

BELÉM
2023

WOLDSON LEONNE PEREIRA GOMES

SISTEMA DE AUTOMAÇÃO IOT PARA GESTÃO DE ATIVOS NO CENÁRIO DA INDÚSTRIA 4.0

Dissertação de Mestrado apresentado ao Programa de Pós Graduação em Engenharia Elétrica da Universidade Federal do Pará, como requisito para a obtenção do Grau de Mestre em Engenharia Elétrica.

Orientador: Profº. Drº. Antonio da Silva Silveira.

BELÉM
2023

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

P436s Pereira Gomes, Woldson Leonne.
Sistema de automação iot para gestão de ativos no cenário de indústria 4.0 / Woldson Leonne Pereira Gomes. — 2023.
181 f. : il. color.

Orientador(a): Prof. Dr. Antonio da Silva Silveira
Coorientador(a): Prof. Dr. Marcos Cesar da Rocha Seruffo
Dissertação (Mestrado) - Universidade Federal do Pará, Instituto de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Belém, 2023.

1. Supervisório. 2. ESP-32. 3. InfluxDB. 4. Protocolos de Comunicação. 5. Grafana. I. Título.

CDD 629.89

WOLDSON LEONNE PEREIRA GOMES

SISTEMA DE AUTOMAÇÃO IOT PARA GESTÃO DE ATIVOS NO CENÁRIO DA INDÚSTRIA 4.0

Dissertação de Mestrado apresentado ao Programa de Pós Graduação em Engenharia Elétrica da Universidade Federal do Pará, como requisito para a obtenção do Grau de Mestre em Engenharia Elétrica.

Data da avaliação: _____/_____/_____

Conceito: _____

Banca Examinadora:

_____ - Orientador
Antonio da Silva Silveira
Doutor em Engenharia de Automação e Sistemas
Universidade Federal do Pará (UFPA)

_____ - Coorientador
Marcos Cesar Da Rocha Seruffo
Doutor em Engenharia Elétrica
Universidade Federal do Pará (UFPA)

_____ - Membro
Adriana Rosa Garcez Castro
Doutora em Engenharia Elétrica
Universidade Federal do Pará (UFPA)

_____ - Membro
Maria da Conceição Pereira Fonseca
Doutora em Engenharia Elétrica
Universidade Federal do Pará (UFPA)

_____ - Membro
Thiago Mota Soares
Doutor em Engenharia Elétrica
Universidade Federal do Pará (UFPA)

VISTO:

_____
Diego Lisboa Cardoso
Doutor em Automatique Et Productique
Universidade Federal do Pará (UFPA)
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica (PPGEE)

“O sol já se foi, agora só nos resta as memórias daquela luz.”

- Naruto Uzumaki

AGRADECIMENTO

Agradeço primeiramente a Deus. Obrigado pelo dom da vida, por toda sabedoria que me foi concedida e sempre estar me abençoando. Ele sabe das minhas dificuldades e o quanto lutei e luto para alcançar meus sonhos.

Agradeço imensamente aos meus pais, Walmick e Mírian, os quais sempre se esforçaram para me oferecer todo suporte necessário para estar aqui hoje, mesmo com incontáveis dificuldades encontradas, sempre trataram minha educação como prioridade. Além dos meus pais, na minha família agradeço a minha irmã, Kacau, por cuidar de mim no início da minha vida longe dos meus pais e por todo carinho.

RESUMO

A quarta revolução industrial possui diversos pilares, sendo *Internet of Things* e *Big Data* uns dos mais proeminentes. Essas tecnologias possibilitam a coleta e análise em tempo real de grandes conjuntos de dados, permitindo o desenvolvimento de modelos para as mais diversas situações, desde o comportamento do consumidor até a prevenção de falhas fabris. Nesse contexto, o presente trabalho aborda o desenvolvimento de uma arquitetura completa para a implementação de um sistema de automação para Indústria 4.0, a nível de hardware e software, baseado na coleta de dados de temperatura, horímetro, vibração e corrente em motores elétricos de uma indústria de alumínio primário. A partir das variáveis mensuradas, pode-se obter dados de vibração no domínio da frequência, desbalanceamento de fases e estimação com filtro de Kalman. Adotou-se um algoritmo de decisão multicritério para auxiliar na escolha para linguagem de programação. Após a elaboração dessa sistemática, obteve-se um conjunto de soluções que tornaram viável o desenvolvimento do sistema, o qual foi validado em um setup experimental controlado. O sistema de automação desenvolvido foi denominado IOTCORE, o qual realiza a coleta das variáveis em tempo real, com baixa latência, alto desempenho, e possibilita transmitir, armazenar e visualizar os dados em diversos supervisórios.

Palavras-chave: Supervisório, ESP-32, InfluxDB, Protocolos de Comunicação, Grafana.

ABSTRACT

The fourth industrial revolution has several pillars, with the Internet of Things and Big Data being one of the most prominent. These technologies make it possible to collect and analyze large data sets in real time, allowing the development of models for the most diverse situations, from consumer behavior to the prevention of manufacturing failures. In this context, the present work addresses the development of a complete architecture for the implementation of an automation system for Industry 4.0, at the hardware and software level, based on the collection of temperature, hour meter, vibration and current data in electric motors of a primary aluminum industry. From the measured variables, vibration data can be obtained in the frequency domain, phase imbalance and Kalman filter estimation. A multicriteria decision algorithm was adopted to assist in choosing the programming language. After elaborating this systematic, a set of solutions was obtained that made the development of the system feasible, which was validated in a controlled experimental setup. The automation system developed was called IOTCORE, which performs the collection of variables in real time, with low latency, high performance, and makes it possible to transmit, store and visualize the data in several supervisory systems.

Keywords: Supervisory, ESP-32, InfluxDB, Communication Protocols, Grafana.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

IoT	*Internet of Things*
AR/VR	*Augmented reality / Virtual Reality*
CPS	*Cyber-Physical System*
SCADA	*Supervisory Control And Data Acquisition*
MQTT	*Message Queuing Telemetry Transport*
SMS	*Short Message Service*
SCT	*Split-core Current Transformer*
OPC UA	*Open Platform Communications Unified Architecture*
SQL	*Structured Query Language*
API	*Application Programming Interface*
AHP	*Analytic Hierarchy Process*
GPIO	*General Purpose Input/Output*
ADC	*Analogic to Digital Converter,*
DAC	*Digital to Analog Converter*
SPI	*Serial Peripheral Interface*
I2C	*Inter-Integrated Circuit*
UART	*Universal Asynchronous Receiver / Transmitter*
PWM	*Pulse Width Modulation*
IDE	*Integrated Development Environment*
USB	*Universal Serial Bus*
HDMI	*High-Definition Multimedia Interface*
RAM	*Random Access Memory*
TI	Tecnologia da Informação
AWS	Amazon Web Services
VMM	*Virtual Machine Monitor*
CPU	*Central Processing Unit*
VM	*Virtual Machine*
SSMS	SQL Server Management Studio
NoSQL	*Not Only Structured Query Language*
HTTP	*Hypertext Transfer Protocol*
MIT	Massachusetts Institute of Technology

UDP	*User Datagram Protocol*
TCP	*Transmission Control Protocol*
CoAP	*Constrained Application Protocol*
IP	*Internet Protocol*
QoS	*Quality of Service*
SSL	*Secure Sockets Layer*
TLS	*Transport Layer Security*
HMI	*Human Machine Interface*
2D	2 dimensões
3D	3 dimensões
DNP3	*Distributed Network Protocol*
EMS	*Energy Management System*
BMS	*Buildings Management System*
AMS	*Asset Management System*
RMS	*Root Mean Square*
OPC	*OLE for Process Control*
MS	Microsoft
MSDE	Microsoft SQL Server Desktop Engine
ETL	*Extract, Transform, Load*
BI	*Business Intelligence*
DAX	*Data Analysis eXpression*
SDK	*Software Development Kit*
SVG	*Scalable Vector Graphics*
ELK	*Elasticsearch, Logstash, Kibana*
RDBM	*Relational Database Management System*
TSDB	*Time Series Database*
URL	*Uniform Resource Locator*
WSGI	*Web Server Gateway Interface*
MEMS	*Micro Electro Mechanical System*
FFT	*Fast Fourier Transform*
DSP	*Digital Signal Processor*
AC	*Alternating Current*
RPi	Raspberry Pi

SO	Sistema Operacional
VNC	*Virtual Network Computing*
SSD	*Solid State Drive*
JSON	*JavaScript Object Notation*
ODBC	*Open Database Connectivity*
TDS	*Tabular Data Stream*
UTC	*Universal Time Coordinated*
CSS	*Cascading Style Sheet*
SMTP	*Simple Mail Transfer Protocol*
DAQ-IOT	*Data Acquisition - Internet of Things*
PCB	*Printed Circuit Board*
OLED	*Organic Light-Emitting Diode*

SUMÁRIO

1 INTRODUÇÃO

1.1 Considerações iniciais

Indústria 4.0 é um termo que significa a quarta revolução industrial. Este período da industrialização é impulsionado por avanços tecnológicos como Internet das Coisas (IoT), inteligência artificial (IA), robótica, realidade aumentada (AR), computação em nuvem, entre outros avanços (NAYYAR; NAVED; RAMESHWAR, 2023). A Indústria 4.0 está mudando a forma como as empresas operam e a sociedade atua nos ambientes em que são forçadas a competir. A convergência entre a automação e as tarefas cotidianas com tecnologias da informação permite melhorar as operações, reduzir custos, melhorar a qualidade de vida e melhorar a qualidade de processos (PENA-CABRERA; LOMAS; LEFRANC, 2019).

Desse modo, a Indústria 4.0 pode ser considerada a revolução industrial do século XXI, caracterizada por uma fabricação com a utilização de inteligência artificial, *machine learning* e tecnologias automatizadas nos processos (DEY; TAMANE, 2019).

Para Sujatha, Prakash e Jhanjhi (2023), as tecnologias que facilitam a Indústria 4.0 incluem a Internet das Coisas (IoT), big data, realidade aumentada/realidade virtual (AR/VR), robôs autônomos, computação em nuvem (*cloud*), cibersegurança e simulação.

Simultaneamente, o rápido desenvolvimento da Internet das Coisas afeta quase todos os aspectos da sociedade, incluindo saúde, transporte e indústria de produção (ASHOURI; DAVIDSSON; SPALAZZESE, 2021). A IoT pode ser entendida como um conjunto de tecnologias que conectam sensores, atuadores, e dispositivos capazes de se comunicar entre si através de protocolos específicos em redes industriais de automação.

A digitalização de dados de máquinas, métodos, processos, procedimentos e dispositivos inteligentes, integra e complementa a camada operacional de uma planta industrial, permitindo a comunicação e integração de sistemas e controles, permitindo respostas e tomadas de decisão em tempo real. Assim, a IoT torna-se um pré-requisito para a Indústria 4.0 (ANANDAN et al., 2022).

Através de IoT, é possível estabelecer o conceito de fusão entre mundo físico e virtual, conhecido como CPS (*Cyber Physical Systems*). CPS é a integração entre

processos físicos com a capacidade de controlar e monitorar, normalmente com laço de realimentação (PAULO; OLIVEIRA, 2017).

Esses sistemas cyber físicos tem uma estrutura básica que os caracterizam, composta de microcontroladores capazes de controlar atuadores e ler dados de sensores, os quais são de fundamental importância para realizar essa interface com os sistemas físicos, além de necessitarem de uma interface de comunicação para interligação de outros CPS ou para envio de dados para a nuvem (JAZDI, 2014).

No Brasil, o termo designado para descrever tal conjunto de tecnologias disruptivas é “transformação digital” e tem como proposta a integração de vários setores, tais como: infraestrutura, tecnologias, negócios, leis e ética com um forte impacto econômico e social (SOUSA; TAIRA; PARK, 2019).

A tecnologia IoT pode ser aplicada em vários setores, como produção, onde a maior parte da tecnologia está sendo implementada e empregada, derivados de máquinas que podem monitorar, analisar e prever problemas em potencial de forma autônoma, o que significa menos tempo de inatividade e mais eficiência em gerenciamento de instalações em geral (ANANDAN et al., 2022).

Ao automatizar processos industriais e padronizar um fluxo de comunicação, faz-se necessário controlar e supervisionar os atuadores, sensores ou variáveis indiretas calculadas, e esse processo ocorre de forma remota através de sistemas de supervisão, controle e aquisição de dados, conhecidos como sistemas SCADA (*Supervisory Control And Data Acquisition*) (GARCIA JR, 2019).

Os sistemas SCADA, baseados em computador, evoluíram nos últimos 40 anos, de operações autônomas e compartimentadas para arquiteturas de rede que se comunicam por grandes distâncias. Esses sistemas englobam a coleta de informações, em uma série de telas ou visores do operador.

Em ambientes industriais, a produção ocorre de forma regular, com isso as máquinas operam de forma contínua. Dentre os equipamentos da planta, os motores elétricos são de extrema importância nos processos industriais, são ativos gerenciados por equipes de manutenção. A falha em equipamentos dessa natureza representa uma parada na produção e, consequentemente, geração de perdas. Nesse sentido, monitorar esses ativos torna-se essencial em grandes indústrias.

Existem várias variáveis que estão relacionadas a falhas no equipamento caso estejam fora dos limites de operação normal do ativo. As variáveis mais importantes

são temperatura, vibração e corrente elétrica (ONÍLIO et al., 2021). Essas variáveis estão correlacionadas em alguns casos, por exemplo um desbalanceamento de fase pode ocasionar aumento de vibração e temperatura.

Muitas indústrias realizam esse tipo de coleta de variáveis, porém muitas vezes ocorre de forma pontual e planejada através de inspeções que ocorrem periodicamente. A falta dessa coleta em tempo real abre margem para maior índice de manutenções corretivas, devido muitos ativos, baixo número de inspetores e longos intervalos entre inspeções.

Quando as inspeções são realizadas com pouca frequência, torna-se difícil detectar problemas em seu estágio inicial, o que pode resultar em falhas inesperadas e custos elevados de manutenção corretiva.

Além disso, as inspeções em motores com baixa recorrência geralmente exigem o desligamento do equipamento, o que pode impactar significativamente a produtividade e os prazos de entrega. Por isso, a utilização de tecnologias de monitoramento remoto, como a análise de vibração em tempo real, pode ser uma solução viável para garantir a manutenção preventiva dos motores e evitar falhas inesperadas.

Baseado nos benefícios que a Indústria 4.0 proporciona, em especial a IoT, pode-se adotar soluções de monitoramento dessas variáveis aplicadas a manutenção preditiva. Ou seja, utilizar sensores de temperatura, vibração e corrente para prever falhas em motores, antes que as mesmas ocorram.

1.2 Motivação

Com base em trabalhos recentes, é possível afirmar que o monitoramento online de vibração, temperatura e corrente em motores de indução continua sendo uma área de grande interesse para a indústria. Alguns estudos recentes exploraram novas técnicas e tecnologias para melhorar a precisão e a eficiência desses sistemas.

Existem também algumas empresas que oferecem esse serviço, conforme um estudo publicado em Souza *et al.* (2022), realizou-se uma análise do sistema preditivo da empresa SEMEQ. De acordo com os autores, o sistema possui um sensor de

vibração e temperatura, que envia os dados via Wi-Fi para um *gateway*[1], sendo que esse último envia os dados coletados para a nuvem. O sensor funciona à bateria, não necessitando de cabos de alimentação, e o *gateway* necessita de uma alimentação externa. O sistema possui interface *web* e *mobile*, algoritmos de *deep learning* para predição de falhas, bem como algoritmos de transformada de Fourier para realizar análises no domínio da frequência. Não se tem conhecimentos acerca dos protocolos de comunicação adotados, nem mais informações construtivas no estudo, devido ser de natureza de tecnologia fechada.

Outro estudo, explorado por Onílio *et al*. (2021), aborda o desenvolvimento de um sistema IoT para coleta, processamento, transmissão e armazenamento de dados de vibração, temperatura e corrente em somente uma fase, aplicado em monitoramento online de motores de indução. O sistema utiliza um microcontrolador ESP-32, e o protocolo de comunicação MQTT (*Message Queuing Telemetry Transport*) para enviar os dados para uma plataforma chamada Ubidots. Essa plataforma possui um *broker* integrado em nuvem, o qual é responsável coordenar mensagens entre os diferentes clientes, bem como banco de dados de séries temporais, e permite a construção de *dashboards*, que é um painel visual que contém informações, métricas e indicadores para análise em tempo real, bem como configuração de alarmes e notificações via e-mail, Telegram, SMS (*Short Message Service*), entre outros. Os sensores utilizados nesse trabalho foram o SCT-013 com medição de corrente até 100A, e um MPU-6050 para medição de vibração e temperatura. O sistema possui alimentação através de baterias, o que dispensa alimentação externa, tornando-se uma vantagem, porém os autores não discutem sobre a vida útil da bateria, e a colocam como um desafio em trabalhos futuros.

Outros estudos abordam mais proeminentemente aspectos de computação em nuvem, no estudo de Rodríguez (2020), o autor desenvolveu um sistema para coleta, processamento e transmissão de dados de vibração, temperatura e corrente trifásica, utilizando computação em nuvem. O sistema de aquisição de dados possui um ESP-32, com alimentação externa, três sensores de corrente SCT-013 com medição de até 100A e um sensor de vibração e temperatura MPU-6050. O microcontrolador envia os dados diretamente para uma máquina virtual Ubuntu hospedada no Azure. Essa

[1] Gateway: em uma rede de comunicações, é um nó de rede equipado para interfacear com outra rede que usa protocolos diferentes.

máquina virtual possui um *broker* MQTT chamado mosquitto para estabelecer o protocolo de comunicação; Node-RED, o qual estabelece uma interface de envio de dados do *broker* MQTT para os softwares InfluxDB e MATLAB, além de conectar os dados do InfluxDB ao software Grafana, para criação de dashboards em tempo real. Além disso, nesse estudo adotou-se soluções de predição de falhas com algoritmos inteligentes baseado em tendência e modelo.

O presente estudo se alinha com outros estudos da literatura que exploram o monitoramento online de vibração, temperatura e corrente em motores industriais. No entanto, diferentemente dos estudos mencionados, este trabalho busca desenvolver um sistema completo, incluindo a escolha de linguagens de programação, configuração de servidores, desenvolvimento de protótipos, criação de aplicação web, criação de telas de supervisório e implementação de algoritmos de alerta. Ao abordar esses aspectos, o trabalho contribui para a implementação prática de soluções de monitoramento online de motores elétricos, com potencial para aumentar a eficiência operacional e prevenir falhas em processos industriais.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo geral deste trabalho trata-se do desenvolvimento de um sistema de automação, baseado em IoT, para coleta, transmissão, armazenamento e exibição de dados de temperatura, vibração e corrente trifásica em motores elétricos.

1.3.2 Objetivos específicos

- Aplicar algoritmos de decisão multicritério para escolha da linguagem de programação a ser utilizada para o desenvolvimento de um sistema de automação para Indústria 4.0, mais especificamente visando o desenvolvimento de um supervisório;
- Estabelecer as configurações de um servidor principal para gerenciamento do fluxo de dados pelo protocolo de comunicação, armazenamento em banco de dados e servir aplicação do supervisório;

- Virtualizar um servidor de *backup*;
- Desenvolver um protótipo capaz de coletar informações de temperatura, aceleração triaxial, velocidade de rotação triaxial, corrente elétrica trifásica, desbalanceamento de corrente entre fases, estimação de desbalanceamento de corrente entre fases com filtro de Kalman, horímetro, status de operação e enviar os dados através de protocolo de comunicação MQTT e OPC UA;
- Desenvolver aplicação web *open source* via programação em python, para supervisionar o sistema de automação;
- Desenvolver telas de supervisório no ICONICS Genesis64, Power BI (*Business Intelligence*) e Grafana;
- Desenvolver algoritmos de alertas e notificações via e-mail, para alertar sobre anomalias, através de configurações de máximo e mínimo para os valores das variáveis mensuradas.

1.4 Justificativa

A quarta revolução industrial está atualmente entre as principais prioridades de muitas organizações, centros de pesquisa e universidades, mas a maioria dos especialistas acadêmicos acredita que o termo “Indústria 4.0” em si não é claro, e é difícil para as empresas de manufatura entender esse fenômeno em tempo hábil e determinar as etapas necessárias para a transição para a Indústria 4.0 (NAYYAR; NAVED; RAMESHWAR, 2023).

Além dessa dificuldade em determinar as etapas necessárias, outra barreira identificada se trata da resistência de gestores para modernização dos parques tecnológicos industriais, por receio de detrimento na relação custo/benefício. Assim, muitas empresas preferem manter o sistema de produção legado, destinando recursos financeiros em outras áreas e, consequentemente, não acompanhando a tendência da transformação digital (GOMES; SERUFFO; SILVEIRA, 2024).

Vale ressaltar que, em 2022, o Brasil ocupou o 54º lugar no índice global de inovação, e 51º em produtos criativos. Essa posição reflete uma melhoria de posição em relação a anos anteriores, e a pesquisa também revela que as tecnologias em IoT

estão entre os principais impulsionadores desse crescimento (WORLD INTELLECTUAL PROPERTY ORGANIZATION, 2022).

Apesar do avanço no índice global de inovação, nota-se que o crescimento da automação em ambientes industriais ainda é pequeno. Muitas empresas ainda adotam processos manuais, os quais oferecem margem para falhas humanas, acidentes, e mais tempo para produção (GOMES; SERUFFO; SILVEIRA, 2024).

Esses índices com pequenos avanços da automação na indústria brasileira, são decorrentes de vários fatores que englobam as mais variadas áreas de uma empresa, mas vale destacar dois pontos principais: os custos relacionados ao investimento inicial, e a falta de conhecimento necessário para implementação (GOMES; SERUFFO; SILVEIRA, 2024).

Aliado a isso, muitas empresas que já possuem sistemas automatizados nos parques industriais, não são envolvidas com a mudança para a Indústria 4.0 devido possuírem tecnologias legadas e fechadas, comprometendo assim a autonomia das empresas em seus processos industriais, representando um alto custo uma mudança tecnológica desse nível. Desse modo, as empresas preferem permanecer com os seus sistemas operantes, mesmo que eles não ofereçam a melhor solução, comparado com preceitos de transformação digital (GOMES; SERUFFO; SILVEIRA, 2024).

Apesar de todos os avanços em tecnologias para Indústria 4.0, percebe-se uma certa dificuldade na escolha de quais soluções adotar para estruturar um sistema baseado em IoT. Segundo Silva (2020), a escolha do sistema mais adequado para uma determinada tarefa atualmente é baseada no conhecimento das partes envolvidas, normalmente baseada na experiência ou na intuição, embora não signifique que seja a decisão mais adequada. Com isso, deixa margem para a aquisição de várias tecnologias com padrões diferentes em uma mesma planta industrial, o que ocasiona vários problemas, principalmente a nível de supervisórios.

A falta de padronização de tecnologias para a criação de softwares supervisórios é um grande desafio para a implementação da Indústria 4.0. Com a rápida evolução tecnológica, surgem novas soluções de software constantemente, o que pode levar a uma grande variedade de tecnologias diferentes sendo utilizadas em diferentes sistemas de produção. Isso dificulta a interoperabilidade entre sistemas,

tornando a integração e o compartilhamento de dados entre diferentes plataformas um desafio.

Além disso, a falta de padronização pode aumentar os custos de implementação e manutenção de sistemas, bem como dificultar a formação de pessoal especializado em diferentes tecnologias. Para garantir o sucesso da Indústria 4.0, é necessário desenvolver padrões e protocolos comuns que facilitem a interoperabilidade entre diferentes tecnologias e sistemas, permitindo a criação de soluções de software supervisórios eficientes e eficazes.

1.5 Estrutura do trabalho

Este trabalho está catalogado em seis capítulos, divididos em estudos teóricos e práticos. O presente capítulo tem por intuito apresentar o tema e proposição do estudo, ou seja, realizar uma introdução do assunto, definir os objetivos, justificativa e expor alguns dos trabalhos mais relevantes já existentes nessa conjuntura.

No segundo capítulo é realizado o referencial teórico, através de um levantamento bibliográfico acerca do tema, destacando os principais pontos necessários e utilizados para a elaboração das ideias fundadas no trabalho.

O terceiro capítulo é a composição da metodologia utilizada para buscar atingir os objetivos do trabalho, esclarecendo as formas necessárias para alcançar os resultados esperados.

No quarto capítulo, são expostos os resultados obtidos, com base na metodologia aplicada. Esse capítulo demonstra os resultados com números e ilustrações gráficas de forma a comparar os métodos aplicados.

Por fim, os capítulos cinco e seis são destinados, respectivamente, para uma sucinta discussão acerca dos resultados obtidos, e uma conclusão acerca do trabalho como um todo, bem como a sugestão de melhorias para futuros trabalhos.

2 REFERENCIAL TEÓRICO

Este capítulo tem como objetivo apresentar uma análise de diversas tecnologias e ferramentas utilizadas em projetos de Internet das Coisas (IoT) e Indústria 4.0. Serão abordados os seguintes temas:

- Microcontrolador ESP32: um microcontrolador de baixo custo e alta performance, amplamente utilizado em projetos de IoT;
- Microcomputador Raspberry Pi: um computador de placa única (*single-board computer*) que pode ser utilizado para diversos fins, desde projetos de IoT até servidores web;
- Conteinerização Docker: uma tecnologia que permite empacotar e distribuir aplicações de maneira eficiente e confiável, independentemente do ambiente de execução;
- Virtualização com VMware: uma tecnologia de virtualização que permite criar máquinas virtuais e executar diversos sistemas operacionais em um único servidor físico;
- Bancos de dados SQL Server e InfluxDB: dois bancos de dados amplamente utilizados em projetos de IoT e Indústria 4.0, cada um com suas próprias características e vantagens;
- Protocolos de comunicação MQTT e OPC UA: dois protocolos amplamente utilizados em projetos de IoT e Indústria 4.0 para comunicação entre dispositivos e sistemas;
- Software de supervisão e controle ICONICS Gênesis 64: uma ferramenta de supervisão e controle utilizada em projetos de automação industrial;
- Ferramentas de visualização de dados Power BI e Grafana: duas ferramentas de visualização de dados amplamente utilizadas em projetos de IoT e Indústria 4.0;
- *Framework* de desenvolvimento web Flask: um *framework* de desenvolvimento web em Python, amplamente utilizado para desenvolvimento de aplicações web e APIs;
- Algoritmos de decisão multicritério AHP: uma metodologia para tomada de decisão em situações com múltiplos critérios a serem considerados;

- Gestão de ativos da manutenção preditiva: uma abordagem para manutenção industrial que utiliza dados de sensores e análises preditivas para maximizar a eficiência e minimizar os custos de manutenção;
- Diagnóstico de falhas em motores elétricos: uma área de estudo que utiliza dados de sensores e análises preditivas para identificar e prevenir falhas em motores elétricos;
- Sensor de corrente SCT-013: um sensor de corrente não invasivo amplamente utilizado em projetos de IoT e Indústria 4.0;
- Sensor de movimento MPU6050: um sensor de movimento que mede aceleração e rotação em três eixos, amplamente utilizado em projetos de IoT e Indústria 4.0;
- Análise de frequência por Transformada de Fourier: uma técnica matemática utilizada para decompor um sinal em suas componentes de frequência;
- Filtro de Kalman: um algoritmo de filtragem utilizado para estimar estados de sistemas dinâmicos a partir de dados de sensores.

Cada um desses temas é relevante para projetos de IoT e Indústria 4.0, e são amplamente utilizados em diversas áreas, desde monitoramento de processos industriais até análise de dados em tempo real para tomada de decisão. A combinação dessas tecnologias e ferramentas pode ser extremamente poderosa, permitindo criar soluções completas e integradas para diversas aplicações.

Neste capítulo, serão apresentados conceitos básicos sobre cada um desses temas, ao final deste capítulo, espera-se que o leitor tenha uma compreensão ampla das tecnologias e ferramentas utilizadas em projetos de IoT e Indústria 4.0, assim como uma ideia de como elas podem ser utilizadas em conjunto para criar soluções completas e integradas.

2.1 Microcontrolador ESP32

O microcontrolador ESP32 é um dispositivo eletrônico amplamente utilizado no desenvolvimento de projetos de Internet das Coisas (IoT) e outros projetos eletrônicos devido a sua versatilidade e facilidade de uso (RODRÍGUEZ, 2020).

O ESP32 possui diversos pinos de entrada e saída, sendo capaz de controlar uma grande variedade de dispositivos e sensores. Os pinos do ESP32 são divididos em diferentes categorias, incluindo GPIOs (*General Purpose Input/Output*), ADCs (*Analog-to-Digital Converters*), DACs (*Digital-to-Analog Converters*), SPIs (*Serial Peripheral Interfaces*), I2Cs (*Inter-Integrated Circuit Bus*), UARTs (*Universal Asynchronous Receiver/Transmitter*), entre outros (CAMERON, 2021).

Ele é baseado no chip ESP32, desenvolvido pela empresa chinesa Espressif Systems, e possui dois núcleos de processamento, suporte a comunicação sem fio Wi-Fi e Bluetooth, além de uma ampla variedade de periféricos integrados, como ADCs, DACs, PWMs, UARTs, SPIs, I2Cs, entre outros (ESPRESSIF SYSTEMS, 2023).

Os ADCs e DACs são responsáveis por converter sinais analógicos em digitais e vice-versa, respectivamente. O ESP32 possui 18 canais de ADC, numerados de 0 a 17, com uma resolução de 12 bits. Já os DACs são apenas dois, mas com uma resolução de 8 bits. Os ADCs podem ser usados para ler sensores analógicos, como sensores de temperatura, umidade e pressão (NIRANJANA et al., 2022).

As interfaces SPI e I2C permitem que o ESP32 se comunique com outros dispositivos, como sensores, displays, módulos Wi-Fi e módulos Bluetooth, entre outros. A interface SPI é uma interface serial síncrona que suporta taxas de transferência de dados de até 80 Mbps. A interface I2C é uma interface serial assíncrona que suporta taxas de transferência de dados de até 3,4 Mbps. Ambas as interfaces são capazes de se comunicar com vários dispositivos ao mesmo tempo, usando apenas alguns pinos do microcontrolador (KARADUMAN; CHALLENGER, 2021).

O uso desses periféricos integrados permite que o ESP32 se comunique com o mundo externo de maneira eficiente e efetiva. Os ADCs permitem que o microcontrolador leia sinais analógicos, enquanto as interfaces SPI e I2C permitem que ele se comunique com outros dispositivos. Esses periféricos são amplamente utilizados em projetos de IoT devido à sua facilidade de uso e flexibilidade (CAMERON, 2021).

Segundo Oner (2021), o ESP32 se tornou uma escolha popular no desenvolvimento de projetos de IoT devido à sua capacidade de se conectar a redes sem fio, como Wi-Fi e Bluetooth, permitindo que dispositivos conectados à internet se

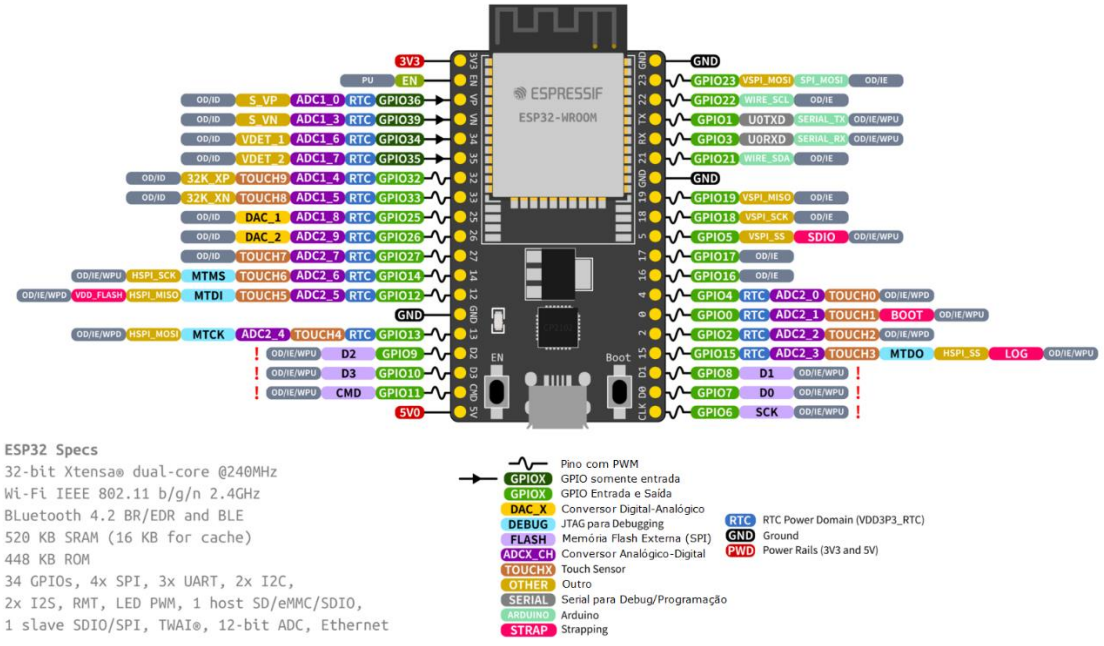
comuniquem com o microcontrolador. Além disso, a ampla disponibilidade de recursos online e a grande comunidade de desenvolvedores que utilizam o ESP32 em seus projetos contribuem para sua popularidade e fácil aprendizado.

De acordo com Selvam (2022), o ESP32 é capaz de executar programas em linguagem Python, o que o torna uma opção popular para o desenvolvimento rápido de protótipos de projetos eletrônicos. Além disso, o ESP32 é compatível com uma variedade de *Integrated Development Environments* (IDEs), como o Arduino IDE e o PlatformIO, facilitando a programação do microcontrolador por desenvolvedores de diferentes níveis de habilidade.

Um dos aspectos mais relevantes do ESP32 é sua capacidade de oferecer recursos de segurança avançados, como criptografia de dados e autenticação de dispositivos, tornando-o uma opção segura para o desenvolvimento de soluções conectadas, como destaca Cameron (2021).

Os GPIOs são os pinos mais versáteis do ESP32, podendo ser configurados tanto como entrada quanto como saída. Eles são numerados de 0 a 39, e cada um possui funcionalidades específicas, como interrupção externa, PWM (*Pulse-Width Modulation*), entre outras (KARADUMAN; CHALLENGER, 2021). A Figura 1 ilustra o pinout do ESP32.

Figura 1 - Pinout ESP-32



Fonte: adaptado de Espressif Systems (2023)

Nessa figura, é possível ver a localização dos pinos do ESP32, bem como as suas funções e características específicas. É importante destacar que o *pinout* pode variar de acordo com a placa ou módulo específico do ESP32 utilizado.

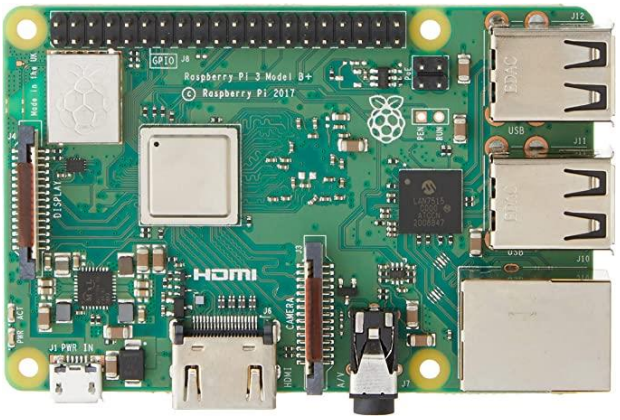
Em resumo, o microcontrolador ESP32 é uma ferramenta poderosa para o desenvolvimento de projetos eletrônicos, especialmente na área de IoT. Com sua facilidade de uso, suporte a diferentes linguagens de programação e interfaces de comunicação flexíveis, o ESP32 tem sido amplamente utilizado em uma variedade de projetos. Além disso, a ampla disponibilidade de recursos online e a grande comunidade de desenvolvedores que utilizam o ESP32 em seus projetos contribuem para sua popularidade e fácil aprendizado.

2.2 Raspberry Pi

O Raspberry Pi é um microcomputador de baixo custo que tem se popularizado cada vez mais, principalmente no âmbito educacional e de projetos de automação residencial. Ele foi lançado em 2012 pela Raspberry Pi Foundation, uma organização sem fins lucrativos sediada no Reino Unido, e desde então tem passado por diversas atualizações e melhorias em suas especificações (MONK, 2023).

O Raspberry Pi, ilustrado na Figura 2, é construído em torno de um processador ARM, que pode variar de acordo com o modelo, e possui diversos componentes integrados, como portas USB, Ethernet, Wi-Fi, Bluetooth, saída de vídeo HDMI, entre outros. Além disso, ele possui um conector de 40 pinos que pode ser utilizado para conectar diversos tipos de periféricos, como sensores, displays e atuadores (STAPLE, 2023).

Figura 2 - Raspberry Pi 3 Modelo B+



Fonte: Autor (2023)

Existem diversos modelos de Raspberry Pi, cada um com suas especificações e características específicas. O modelo mais recente é o Raspberry Pi 4, lançado em 2019, que possui um processador de quatro núcleos com *clock* de até 1.5 GHz, 2 GB ou 4 GB de memória RAM, portas USB 3.0, saída de vídeo dual HDMI, Wi-Fi e Bluetooth integrados, entre outras especificações (HATTERSLEY, 2021).

Uma das grandes vantagens do Raspberry Pi é a sua flexibilidade e versatilidade. Ele pode ser utilizado para uma ampla variedade de projetos, desde servidores web até robótica e automação residencial. Além disso, ele é compatível com diversos sistemas operacionais, incluindo o Raspbian, uma distribuição Linux especialmente desenvolvida para o Raspberry Pi (RADOVICI; CULIC, 2022).

Outra grande vantagem do Raspberry Pi é o seu baixo custo. Ele pode ser adquirido por valores a partir de US$35, tornando-se uma opção acessível para projetos de baixo orçamento. Além disso, existem diversas opções de acessórios e periféricos específicos para o Raspberry Pi, como cases, câmeras, displays e sensores (THE RASPBERRY PI FOUNDATION, 2023).

O Raspberry Pi tem se tornado cada vez mais popular na indústria devido ao seu baixo custo em comparação com outras soluções. Isso permite que empresas de diferentes tamanhos e orçamentos possam implementar soluções de automação e controle, monitoramento e coleta de dados, entre outras aplicações, de forma mais acessível. Além disso, o Raspberry Pi é uma plataforma versátil, permitindo que seja utilizado em diferentes áreas e setores, desde a agricultura até a indústria aeroespacial. Essas características fazem do Raspberry Pi uma opção atraente para empresas que buscam implementar soluções tecnológicas com um orçamento mais limitado (WU; MASLOV, 2022).

O Raspberry Pi é uma opção popular para a criação de servidores de IoT (Internet das Coisas). Com suas características de baixo custo e versatilidade, o Raspberry Pi pode ser utilizado para coletar, processar e enviar dados de sensores em aplicações de IoT. Além disso, sua capacidade de executar diferentes sistemas operacionais, como o Raspbian, o Ubuntu e o Windows 10 IoT Core, permite que os desenvolvedores escolham o sistema mais adequado para sua aplicação (SHOVIC, 2021).

O Raspberry Pi também pode ser integrado com diferentes protocolos de comunicação, como o MQTT, facilitando a troca de dados com outros dispositivos IoT.

Com suas capacidades de processamento e conectividade, o Raspberry Pi é uma opção atraente para a criação de servidores de IoT em diversas aplicações, desde a automação residencial até o monitoramento de ativos industriais (PARIKH, 2022).

O Raspbian é um sistema operacional livre e de código aberto baseado no Debian, projetado para rodar em Raspberry Pi. Desenvolvido pela equipe da Fundação Raspberry Pi, o Raspbian é um sistema operacional otimizado para aproveitar ao máximo as capacidades do Raspberry Pi, como a interface GPIO (*General Purpose Input/Output*) e a conexão de periféricos. Ele também inclui uma série de ferramentas e softwares pré-instalados, como o navegador Chromium, o editor de texto Geany, o processador de texto LibreOffice e o ambiente de desenvolvimento Python. Além disso, o Raspbian é compatível com uma ampla gama de software desenvolvido para Debian, facilitando a instalação de novos aplicativos e serviços. O Raspbian é um sistema operacional popular para projetos em Raspberry Pi, desde aplicações de automação até projetos educacionais (JAPON, 2022).

Em resumo, o Raspberry Pi é um microcomputador versátil, acessível e bastante utilizado em projetos de automação, robótica e educação. Ele possui diversas especificações e modelos, cada um com suas características específicas, e é compatível com diversos sistemas operacionais e linguagens de programação.

2.3 Conteinerização

Com a popularização da virtualização e o crescimento da demanda por aplicações escaláveis e portáteis, a conteinerização tem se tornado uma alternativa cada vez mais utilizada na indústria de tecnologia. A tecnologia de conteinerização permite que aplicações e serviços sejam empacotados em um formato único e portátil, isolando-os do ambiente de execução e permitindo que sejam executados de maneira consistente em diferentes plataformas e ambientes. Dentre as diversas ferramentas de conteinerização existentes, o Docker é uma das mais populares e amplamente utilizadas (KANE; MATTHIAS, 2023).

O Docker é uma plataforma de conteinerização de software que permite que desenvolvedores e equipes de TI criem, implantem e executem aplicativos em conteineres. Com o Docker, é possível empacotar uma aplicação junto com suas dependências e bibliotecas em um conteiner isolado, garantindo que ela seja

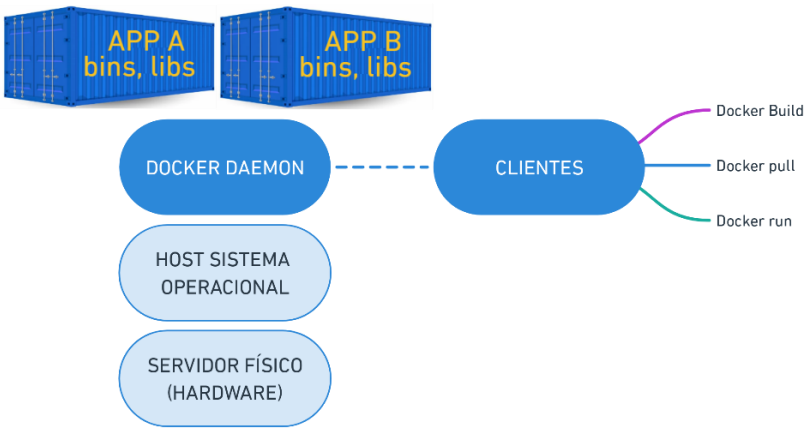
executada de maneira consistente e previsível em diferentes ambientes. Além disso, o Docker oferece uma série de recursos, como gerenciamento de recursos, balanceamento de carga e networking, que simplificam o processo de implantação e escalabilidade de aplicações (LESZKO, 2022).

O uso do Docker tem crescido rapidamente nos últimos anos, especialmente em ambientes de desenvolvimento e produção. A facilidade de uso e a portabilidade são algumas das principais vantagens que fazem do Docker uma ferramenta atraente para desenvolvedores e equipes de TI. Além disso, o Docker é amplamente suportado por plataformas de nuvem, como a AWS e Azure, o que torna mais fácil implantar e gerenciar aplicativos em diferentes ambientes de infraestrutura (FOUDA, 2022).

Uma das principais vantagens do Docker é a capacidade de construir imagens de contêiner personalizadas que contêm todas as dependências necessárias para executar um aplicativo específico. Essas imagens podem ser compartilhadas com outros desenvolvedores ou implantadas em qualquer ambiente que suporte o Docker (SAMETRIYA; VASAVADA, 2022).

Para exemplificar, suponha que um desenvolvedor esteja trabalhando em um aplicativo em um laptop com sistema operacional Windows. Ao usar o Docker, ele pode empacotar o aplicativo em um contêiner e distribuí-lo para outros desenvolvedores que usam diferentes sistemas operacionais, como macOS ou Linux. Esses desenvolvedores podem, então, executar o contêiner em seu próprio ambiente sem precisar instalar dependências adicionais ou alterar seu ambiente de desenvolvimento (CANDEL, 2022). A Figura 3 ilustra a arquitetura básica do Docker.

Figura 3 - Arquitetura Básica Docker



Fonte: Autor (2023)

A arquitetura do Docker é composta por várias camadas, começando pelo host do Docker, onde o software Docker é instalado e onde os contêineres são executados. Em seguida, há o Docker daemon, que gerencia a criação, execução e distribuição de contêineres. A camada de API fornece uma interface para o desenvolvedor interagir com o Docker e a camada de cliente fornece uma interface para que os desenvolvedores possam interagir com a API do Docker. Por fim, há a camada de contêiner, que é onde o aplicativo é empacotado e executado (GKATZIOURAS, 2022).

Para exemplificar uma aplicação real, pode-se citar o SQL Server, um dos bancos de dados mais utilizados no mundo corporativo, devido a sua alta confiabilidade e escalabilidade. Com a utilização do Docker, é possível ter um ambiente de desenvolvimento, teste ou produção com SQL Server instalado em poucos minutos, sem a necessidade de configurações complexas e demoradas (SAMETRIYA; VASAVADA, 2022).

A utilização do SQL Server no Docker é uma das maneiras mais eficientes de se criar e gerenciar um ambiente de Banco de Dados, além de ser uma opção de baixo custo e de fácil manutenção (GKATZIOURAS, 2022).

Em resumo, a containerização com o Docker é uma técnica poderosa para empacotar e distribuir aplicativos com facilidade e eficiência. Com sua interface simples e recursos avançados, o Docker é uma ferramenta popular para a implantação de aplicativos em ambientes de produção, bem como para ambientes de desenvolvimento e teste.

2.4 Máquinas Virtuais

Com o avanço da tecnologia, cada vez mais empresas buscam alternativas para reduzir os custos em infraestrutura de TI, mantendo o alto desempenho e a segurança dos sistemas. Uma das alternativas mais utilizadas é a virtualização, que permite a criação de ambientes virtuais em uma única máquina física (VMWARE INC., 2023).

A virtualização de máquinas é uma tecnologia que permite criar e executar uma ou mais máquinas virtuais dentro de um único servidor físico. Isso significa que é possível compartilhar o mesmo hardware entre várias máquinas virtuais, reduzindo o

número de servidores físicos necessários. Além disso, a virtualização oferece maior flexibilidade e escalabilidade, permitindo que as máquinas virtuais sejam facilmente movidas de um servidor para outro (ROBINSON, 2017).

O VMWare é um dos softwares mais utilizados para virtualização de máquinas. Ele permite a criação e execução de múltiplas máquinas virtuais em um único servidor físico, possibilitando a instalação de diversos sistemas operacionais em uma mesma máquina. Além disso, o VMWare oferece recursos de gerenciamento e monitoramento de máquinas virtuais, como a criação de snapshots (cópias de segurança) e a migração de máquinas virtuais entre servidores (LEVCHENKO; CARDOSO, 2018).

O VMWare é uma opção de virtualização muito utilizada em empresas de todos os portes, por oferecer uma grande variedade de recursos e uma interface amigável para gerenciamento das máquinas virtuais. Além disso, o VMWare oferece opções de virtualização de desktops, permitindo que os usuários acessem seus desktops de qualquer lugar e dispositivo. Alguns dos benefícios da virtualização com VMWare, de acordo com VMWare (2023) são:

- Redução de custos: ao utilizar a virtualização, é possível reduzir o número de servidores físicos necessários, reduzindo custos com hardware, energia elétrica e espaço físico.
- Flexibilidade e escalabilidade: com a virtualização, é possível facilmente adicionar ou remover recursos de uma máquina virtual, além de mover as máquinas virtuais entre servidores.
- Maior disponibilidade e segurança: com o VMWare, é possível criar ambientes de alta disponibilidade e redundância, garantindo a continuidade do negócio em caso de falhas. Além disso, as máquinas virtuais são isoladas umas das outras, oferecendo maior segurança.
- Facilidade de gerenciamento: o VMWare oferece uma interface gráfica de fácil utilização para gerenciamento das máquinas virtuais, além de recursos como *snapshots* e migração de máquinas virtuais.

A arquitetura do VMware é baseada em um *hypervisor*, também conhecido como VMM (*Virtual Machine Monitor*), que fica entre o hardware físico do computador e as máquinas virtuais que são executadas nele (ROBINSON, 2017).

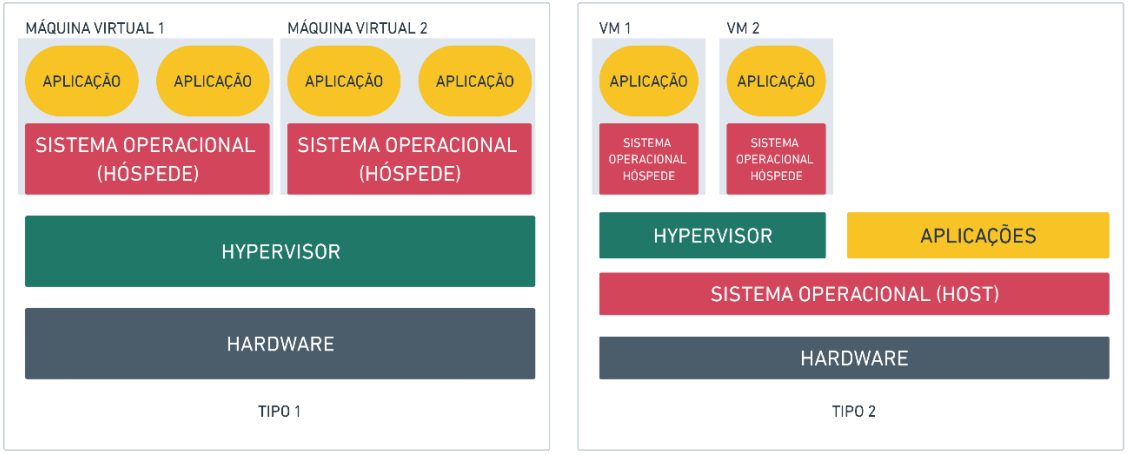
O *hypervisor* é responsável por gerenciar e controlar todos os recursos do hardware físico, como memória, CPU, armazenamento e dispositivos de

entrada/saída. Ele divide o hardware em vários ambientes virtuais, cada um com seus próprios recursos, para que múltiplas máquinas virtuais possam ser executadas ao mesmo tempo, sem interferir umas nas outras (LEVCHENKO; CARDOSO, 2018).

Existem dois tipos de *hypervisors*: o *Type* 1 e o *Type* 2. O *Type* 1, também conhecido como "*bare-metal*", é executado diretamente no hardware físico do computador e é capaz de controlar todos os recursos do sistema. Já o *Type* 2 é executado sobre um sistema operacional já instalado, como um aplicativo, e depende dele para controlar o hardware do sistema (IQBAL; PATTINSON; KOR, 2015).

O VMware utiliza o *hypervisor Type* 1, o que permite uma maior eficiência e desempenho na execução de máquinas virtuais. Além disso, ele também oferece recursos avançados de gerenciamento, como migração de máquinas virtuais entre servidores físicos sem interrupção de serviço e recursos de alta disponibilidade para evitar falhas de hardware (DORDEVIC et al., 2022). A Figura 4 ilustra a diferença entre ambos os tipos.

Figura 4 - Tipos de Hypervisor



Fonte: Autor (2023)

As máquinas virtuais (VMs) desempenham um papel importante na IoT e na Indústria 4.0, permitindo a consolidação de recursos e a execução de aplicativos em ambientes isolados. Na IoT, a virtualização permite que dispositivos de hardware limitados executem vários sistemas operacionais e aplicativos em um único dispositivo, permitindo a flexibilidade e escalabilidade necessárias para lidar com a grande quantidade de dispositivos conectados.

2.5 Banco de Dados

Banco de dados é uma ferramenta fundamental na Indústria 4.0 e Internet das Coisas (IoT). Eles permitem que as empresas coletem, armazenem e gerenciem grandes quantidades de dados de forma eficiente e escalável. Dentre as diversas opções de bancos de dados disponíveis no mercado, o SQL Server e o InfluxDB são duas soluções populares que atendem às necessidades específicas dessas indústrias.

Os fluxos de dados provenientes de cada dispositivo IOT conectado podem ser vistos como coleções de séries temporais, que precisam de técnicas adequadas para garantir sua persistência. Em particular, essas soluções devem ser capazes de fornecer uma ingestão e recuperação de dados eficazes, que são tarefas desafiadoras (DI MARTINO et al., 2019).

Nos próximos tópicos, será abordado detalhes do SQL Server e do InfluxDB. Discutindo as características, recursos e vantagens, bem como as aplicações em diferentes setores da Indústria 4.0.

2.5.1 SQL Server

O SQL Server é um sistema de gerenciamento de banco de dados relacional desenvolvido pela Microsoft. Ele foi projetado para ser um sistema de banco de dados seguro, escalável e confiável, e é amplamente utilizado em empresas de todos os tamanhos e setores (OSAMA, 2019).

O SQL Server é conhecido por sua interface fácil de usar e seus recursos avançados de análise de dados. Ele suporta a linguagem de consulta SQL (*Structured Query Language*), permitindo que os usuários gerenciem e manipulem facilmente seus dados. Além disso, o SQL Server oferece recursos avançados de segurança, como autenticação de usuários, criptografia de dados e auditoria de segurança (PETKOVIĆ, 2020).

O SQL Server também é altamente escalável, permitindo que as empresas gerenciem grandes volumes de dados com facilidade. Ele suporta clusters de servidor, permitindo que os usuários adicionem ou removam servidores do cluster conforme necessário. Além disso, o SQL Server suporta processamento em paralelo, permitindo

que ele aproveite o poder de processamento de vários núcleos de CPU (LAROCK; LAAR, 2023).

Entre os recursos avançados do SQL Server estão a análise de dados em tempo real, a integração com ferramentas de *business intelligence*, a replicação de dados e a integração com outras plataformas Microsoft, como o Excel e o SharePoint. Isso torna o SQL Server uma escolha popular em empresas que precisam de uma solução de banco de dados completa e integrada (POLLACK; STRATE, 2023).

No que diz respeito à sua aplicação na Indústria 4.0 e IoT, o SQL Server é amplamente utilizado para gerenciar grandes volumes de dados gerados por sensores e dispositivos de IoT. Ele pode ser usado para armazenar e analisar dados de sensores em tempo real, permitindo que as empresas monitorem e controlem seus processos de produção e operações em tempo real (CERBULESCU et al., 2022).

Em relação aos custos, a Microsoft oferece várias opções de licenciamento, incluindo licenças por servidor, por núcleo ou por usuário. Para empresas menores, o SQL Server Express é uma opção gratuita que oferece muitos dos recursos básicos do SQL Server, mas com algumas limitações de capacidade de processamento e armazenamento de dados (GORMAN, 2020).

Para empresas maiores, o SQL Server Enterprise é a opção mais completa, oferecendo recursos avançados de segurança, escalabilidade e análise de dados. O licenciamento é baseado no número de núcleos do servidor e pode ser bastante caro para empresas que precisam de grande capacidade de processamento e armazenamento de dados (GORMAN, 2020).

A criação de tabelas e a execução de queries são duas operações fundamentais no SQL Server. As tabelas são estruturas que armazenam os dados do banco de dados, enquanto as queries são comandos que permitem que os usuários acessem e manipulem esses dados (KOROTKEVITCH, 2022).

No contexto da automação, a criação de tabelas e a execução de queries são usadas para gravar alarmes e *setpoints* de dados de processo e controle. Por exemplo, um sistema de automação pode gerar dados de processo e controle, como temperaturas, pressões e velocidades de rotação. Esses dados podem ser armazenados em tabelas do SQL Server para análise posterior (CERBULESCU et al., 2022).

O SQL Server Management Studio (SSMS) é a ferramenta padrão da Microsoft para gerenciamento de banco de dados SQL Server. O SSMS é uma ferramenta poderosa que fornece uma interface gráfica do usuário para gerenciar servidores, bancos de dados, esquemas, tabelas, índices, procedimentos armazenados e muito mais. Ele também oferece uma ampla gama de recursos de administração, como *backups* e restaurações, gerenciamento de segurança, planejamento de manutenção e criação de consultas (LAROCK; LAAR, 2023).

O Azure Data Studio é outra ferramenta de gerenciamento de banco de dados da Microsoft que foi lançada em 2018. O Azure Data Studio é uma ferramenta multiplataforma que permite gerenciar e desenvolver bancos de dados SQL Server em Windows, macOS e Linux. Ele oferece muitos recursos semelhantes ao SSMS, como criação de consultas, gerenciamento de segurança e integração com o controle de origem. No entanto, o Azure Data Studio também oferece suporte para o gerenciamento de bancos de dados em nuvem no Microsoft Azure (D'ANTONI, 2020).

Ambas as ferramentas são muito úteis para gerenciar o SQL Server e cada uma tem suas próprias vantagens. O SSMS é a ferramenta padrão para gerenciar o SQL Server em ambientes locais, enquanto o Azure Data Studio é mais adequado para gerenciar bancos de dados em nuvem no Azure. Além disso, o Azure Data Studio é uma ferramenta mais leve que pode ser executada em várias plataformas, o que pode ser uma vantagem para equipes que trabalham em ambientes heterogêneos (D'ANTONI, 2020).

Em resumo, o SQL Server é um sistema de gerenciamento de banco de dados seguro, escalável e confiável, com recursos avançados de análise de dados e integração com outras plataformas Microsoft (DAVIDSON, 2021). Ele é amplamente utilizado em empresas de todos os tamanhos e setores e é particularmente útil na Indústria 4.0 e IoT para gerenciar grandes volumes de dados gerados por sensores e dispositivos de IoT.

2.5.2 InfluxDB

O InfluxDB é um banco de dados NoSQL (*Not Only Structured Query Language*) de série temporal, projetado para armazenar, consultar e processar grandes volumes de dados de série temporal em tempo real. Ele foi criado para

atender às necessidades de coleta, armazenamento e análise de dados de sensores, infraestrutura e aplicativos de Internet das Coisas (IoT) (DI MARTINO et al., 2019).

O InfluxDB é otimizado para armazenar e consultar dados de série temporal, o que o torna uma escolha popular para aplicações de monitoramento e análise de sistemas, bem como para aplicações de IoT e *edge computing*. Segundo a documentação técnica, InfluxData (2023), algumas das características mais importantes do InfluxDB incluem:

- Armazenamento de dados de série temporal em alta velocidade: o InfluxDB é projetado para lidar com altas taxas de gravação de dados, permitindo que as aplicações armazenem milhões de pontos de dados por segundo.
- Consulta eficiente de dados de série temporal: o InfluxDB oferece uma linguagem de consulta própria, chamada InfluxQL, que é usada para recuperar e manipular dados de série temporal de forma eficiente.
- Escalabilidade horizontal: o InfluxDB pode ser escalado horizontalmente, que é caracterizado pela capacidade de um sistema ou aplicação lidar com um aumento na carga de trabalho ao adicionar recursos adicionais em paralelo, permitindo que as aplicações se expandam conforme necessário.
- Integração com outras ferramentas populares: o InfluxDB oferece integrações com outras ferramentas populares, como Grafana, Telegraf e Kapacitor, o que o torna uma escolha popular para aplicações de monitoramento de sistemas e análise de dados.

Para começar a usar o InfluxDB, é necessário primeiro instalá-lo em um servidor ou em um contêiner Docker. Em seguida, é possível usar o cliente do InfluxDB ou a API HTTP para criar bancos de dados, gravar dados de série temporal e consultar dados armazenados. O InfluxDB também oferece uma interface, que permite visualizar e analisar dados de série temporal em tempo real (INFLUXDATA, 2023).

O InfluxDB é distribuído sob a licença MIT (*Massachusetts Institute of Technology*), que é uma licença de software livre e de código aberto. Isso significa que os usuários têm liberdade para executar, copiar, distribuir, modificar e distribuir modificações do software sem restrições, desde que a licença seja mantida e os termos sejam respeitados (BLOKDYK, 2018).

Em termos de retenção de dados, o InfluxDB oferece uma variedade de políticas de retenção, permitindo que os usuários configurem automaticamente o período de tempo em que os dados são mantidos no banco de dados antes de serem excluídos. Isso permite que os usuários personalizem a quantidade de tempo que desejam armazenar seus dados com base em suas necessidades específicas (INFLUXDATA, 2023).

Quanto à arquitetura, o InfluxDB é projetado para ser escalável e resiliente, permitindo que grandes quantidades de dados de séries temporais sejam armazenadas e consultadas de maneira eficiente. Ele usa um mecanismo de armazenamento de dados de séries temporais otimizado, que é altamente eficiente em termos de espaço e permite que os dados sejam compactados e armazenados em memória e em disco para desempenho ideal (BLOKDYK, 2018).

O InfluxDB também inclui um mecanismo de consulta *SQL-like* poderoso que permite que os usuários pesquisem e analisem facilmente seus dados de séries temporais usando uma variedade de funções e operadores. Além disso, o InfluxDB é compatível com vários protocolos de entrada e saída de dados, incluindo HTTP (*Hypertext Transfer Protocol*), MQTT, UDP (*User Datagram Protocol*) e TCP (*Transmission Control Protocol*), tornando-o altamente flexível e interoperável com outras tecnologias (INFLUXDATA, 2023).

Além da utilização local, o InfluxDB pode ser implantado em uma variedade de plataformas de nuvem, incluindo o Azure, AWS e Google Cloud. O Azure é a plataforma de nuvem pública da Microsoft, que oferece vários serviços de computação em nuvem, incluindo hospedagem de aplicativos, armazenamento de dados e serviços de análise de dados, enquanto que a Amazon Web Services (AWS) é uma plataforma de nuvem pública da Amazon e o Google Cloud é a plataforma de nuvem pública do Google (BLOKDYK, 2018).

2.6 Protocolos de Comunicação

Desde o surgimento da Revolução Industrial, a comunicação tem sido uma parte essencial da indústria. Desde as primeiras redes de telégrafo até os sistemas de automação industrial modernos, a comunicação é uma parte crucial da operação de fábricas e plantas industriais (ZURAWSKI, 2015).

No início, a comunicação na indústria era realizada principalmente por meio de linhas telefônicas e cabos de sinalização. No entanto, com o passar do tempo, novas tecnologias de comunicação foram desenvolvidas para atender às necessidades em constante evolução da indústria. Nos anos 1980 e 1990, surgiram protocolos como o Modbus, que permitiam que dispositivos e sistemas de automação industrial se comunicassem uns com os outros (CUNHA, 2004).

Com o advento da Internet das Coisas (IoT), a comunicação na indústria tornou-se ainda mais sofisticada. A IoT permitiu a criação de redes de sensores e dispositivos que coletam e compartilham dados em tempo real, permitindo que as empresas monitorem e controlem seus processos de maneira mais eficiente e eficaz. Como resultado, surgiram novos protocolos de comunicação projetados especificamente para atender às necessidades da IoT, como MQTT e CoAP (*Constrained Application Protocol*) (HAMDANI; SBEYTI, 2019).

Atualmente, a comunicação na indústria é uma combinação de tecnologias antigas e novas. As redes de comunicação legadas, como Modbus, ainda são usadas em muitas fábricas e plantas industriais, enquanto as tecnologias mais recentes, como MQTT e OPC UA, estão se tornando cada vez mais populares. Esses protocolos de comunicação modernos permitem que as empresas se comuniquem com seus dispositivos e sistemas de automação industrial de maneira mais eficiente e segura (SHI; NIU; SUN, 2020).

Este tópico tem como objetivo apresentar e discutir dois importantes protocolos de comunicação: MQTT e OPC UA. Inicialmente, serão apresentados os conceitos básicos de comunicação de dados, protocolos de comunicação e suas características. Em seguida, o protocolo MQTT será detalhado, abordando suas principais características, aplicações e vantagens em relação a outros protocolos de comunicação. Em seguida, será apresentado o protocolo OPC UA, suas características, funcionalidades e como ele é utilizado em sistemas de automação industrial.

2.6.1 *Message Queuing Telemetry Transport*

O protocolo MQTT (*Message Queuing Telemetry Transport*) é um protocolo de comunicação de dados projetado para uso em dispositivos IoT (*Internet of Things*).

Ele foi desenvolvido pela IBM em 1999 e é amplamente utilizado em aplicações que exigem uma comunicação de dados leve e eficiente (MISHRA; KERTESZ, 2020).

Uma das principais vantagens do protocolo MQTT é sua simplicidade e eficiência. Ele é projetado para minimizar o tráfego de dados na rede, tornando-o ideal para dispositivos com recursos limitados, como sensores e atuadores IoT. Além disso, o protocolo MQTT suporta um modelo de publicação e assinatura, no qual os dispositivos publicam informações em tópicos específicos e outros dispositivos se inscrevem nesses tópicos para receber as informações (PARIKH, 2022).

A arquitetura do protocolo MQTT é baseada no modelo *publish/subscribe* (ou publicação/assinatura, em português). Nessa arquitetura, os dispositivos que enviam informações (*publishers*) não precisam conhecer diretamente os dispositivos que as recebem (*subscribers*). Em vez disso, eles simplesmente publicam as informações em um tópico, e os dispositivos interessados em receber essas informações se inscrevem nesse tópico (HILLAR, 2018).

O broker MQTT é responsável por receber as informações dos *publishers* e encaminhá-las para os subscribers correspondentes. Ele é o intermediário entre os dispositivos que enviam informações e os que as recebem (HILLAR, 2017). A Figura 5 ilustra a arquitetura básica do MQTT:

Figura 5 - Exemplo arquitetura básica MQTT

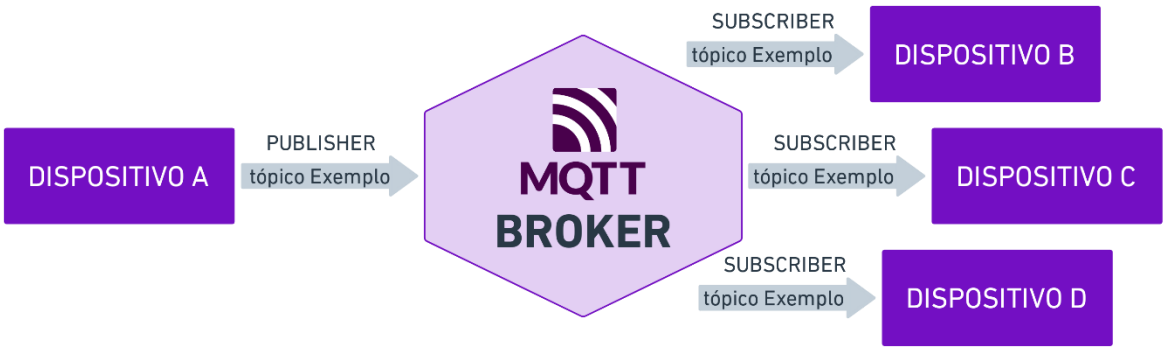
Fonte: autor (2023)

Na Figura 5, pode-se observar que o dispositivo A é um *publisher*, e o dispositivo B é um *subscriber*. O broker MQTT é responsável por receber as informações do dispositivo A e encaminhá-las para o dispositivo B, através do tópico chamado “Exemplo”.

Além disso, é possível que vários dispositivos sejam inscritos no mesmo tópico e recebam as informações publicadas por um *Publisher*, a Figura ilustra essa situação,

na qual vários dispositivos inscrevem o tópico “Exemplo”, o qual é publicado pelo dispositivo A.

Figura 6 - Generalização de exemplo da arquitetura MQTT



Fonte: autor (2023)

Na Figura 6, pode-se observar que os dispositivos B, C e D estão inscritos no mesmo tópico. Quando o dispositivo A publica uma informação nesse tópico, o *broker* MQTT encaminha essa informação para todos os dispositivos inscritos nesse tópico.

Um dos *brokers* MQTT mais utilizados é o Eclipse Mosquitto, que é uma implementação *open source* do protocolo. Ele oferece recursos avançados de segurança, suporte a autenticação de usuários e encriptação de dados, além de ser altamente escalável e flexível (SMART, 2020).

Outro broker MQTT popular é o HiveMQ, que oferece recursos avançados de *clustering*, que permitem que vários brokers trabalhem juntos para gerenciar grandes volumes de tráfego de dados. Ele também oferece recursos avançados de segurança, como autenticação de usuários e controle de acesso baseado em funções (MISHRA; KERTESZ, 2020).

Além desses brokers, existem diversas outras opções disponíveis, como o VerneMQ, o RabbitMQ, o AWS IoT Core e o Google Cloud IoT Core, entre outros. Cada um deles oferece recursos e funcionalidades específicos, permitindo que os usuários escolham a opção mais adequada para suas necessidades.

O protocolo MQTT é baseado em TCP/IP e usa um formato de mensagens binárias para enviar e receber dados. Ele também suporta diferentes níveis de qualidade de serviço (QoS), que determinam o grau de garantia de entrega de uma mensagem. O nível de QoS 0 garante que a mensagem seja entregue pelo menos uma vez, mas sem garantir que ela seja entregue com sucesso. Já o nível de QoS 1

garante que a mensagem seja entregue pelo menos uma vez e com sucesso. Por fim, o nível de QoS 2 garante que a mensagem seja entregue exatamente uma vez e com sucesso (DHAVAL; ASHWIN, 2022).

Em relação a outros protocolos de comunicação, como HTTP, o protocolo MQTT apresenta vantagens significativas. Por exemplo, o HTTP é mais adequado para transferência de arquivos grandes e interações de cliente/servidor. Já o protocolo MQTT é ideal para aplicações IoT devido à sua simplicidade e eficiência. Além disso, o modelo de publicação/assinatura do MQTT é mais flexível do que o modelo cliente/servidor do HTTP (WUKKADADA et al., 2018).

Para exemplificar o funcionamento do protocolo MQTT, pode-se considerar uma aplicação de monitoramento de temperatura em um edifício inteligente. Nessa aplicação, sensores de temperatura são distribuídos pelo edifício e enviam as informações para um servidor central usando o protocolo MQTT. Os dados são publicados em tópicos específicos, como "temperatura/sala1" e "temperatura/sala2", e outros dispositivos, como um sistema de controle de ar condicionado, se inscrevem nesses tópicos para receber as informações. Dessa forma, o sistema pode monitorar a temperatura em tempo real e ajustar o ar condicionado para manter a temperatura ideal em cada sala (PRATAMA; YULIANI, 2021).

A segurança é uma preocupação importante em qualquer sistema de comunicação de dados, e o protocolo MQTT não é exceção. Ele oferece diversas medidas de segurança para garantir que as informações sejam protegidas contra acesso não autorizado e outras ameaças (ATILGAN; OZCELIK; YOLACAN, 2021).

O protocolo MQTT suporta autenticação de cliente e servidor, usando nome de usuário e senha. Além disso, ele oferece suporte a criptografia de ponta a ponta, usando protocolos SSL/TLS (*Secure Sockets Layer*/*Transport Layer Security*). Isso significa que as informações são criptografadas antes de serem enviadas e só podem ser descriptografadas pelo destinatário correto (ALKHAFAJEE et al., 2021).

Outra medida de segurança importante no protocolo MQTT é a verificação de tópicos. Os tópicos são usados para identificar as informações que estão sendo enviadas ou recebidas, e é possível configurar o broker MQTT para permitir apenas determinados tópicos e rejeitar outros. Isso ajuda a evitar que informações sensíveis sejam divulgadas inadvertidamente ou interceptadas por um terceiro mal-intencionado (ATILGAN; OZCELIK; YOLACAN, 2021).

O MQTT é um protocolo de comunicação moderno e altamente eficiente, o qual oferece diversas vantagens em relação a protocolos de comunicação legados. Basicamente, o MQTT em relação a outros protocolos legados apresenta alta eficiência e velocidade de transferência de dados, escalabilidade, suporte a redes distribuídas, recursos avançados de segurança, fácil configuração e manutenção, além de baixo custo de implementação (MISHRA; KERTESZ, 2020).

2.6.2 *Open Platform Communications Unified Architecture*

O Protocolo OPC UA (*Open Platform Communications Unified Architecture*) é um padrão de comunicação de dados para sistemas de automação industrial e de controle de processos. Ele foi desenvolvido para permitir a troca de informações entre diferentes sistemas, equipamentos e dispositivos em uma rede, independentemente do fabricante ou plataforma utilizada (VEICHTLBAUER; ORTMAYER; HEISTRACHER, 2017).

O OPC UA é uma evolução do antigo padrão OPC (*OLE for Process Control*) e é projetado para atender às necessidades de comunicação de dados em ambientes industriais modernos, incluindo a Internet das Coisas. Ele foi criado para fornecer uma arquitetura segura, confiável e escalável para a troca de dados entre sistemas (MORATO et al., 2021).

O OPC UA é baseado em um modelo de informação que descreve a estrutura e os atributos dos dados que são compartilhados entre os sistemas. Este modelo permite que os sistemas compreendam os dados que estão sendo trocados, mesmo que eles tenham sido criados por diferentes fabricantes ou plataformas. Além disso, o OPC UA é projetado para garantir a segurança e a integridade dos dados, utilizando criptografia e autenticação em todas as etapas da comunicação (MUHLBAUER et al., 2020).

O protocolo OPC UA é um protocolo independente de plataforma, o que significa que pode ser executado em diferentes sistemas operacionais. Ele também é independente de linguagem de programação, permitindo que os desenvolvedores implementem soluções usando a linguagem de sua preferência (MAHNKE; LEITNER; DAMM, 2009).

A arquitetura do protocolo OPC UA é projetada para fornecer uma estrutura flexível e escalável para a troca de dados entre sistemas e dispositivos em uma rede industrial. A arquitetura é baseada em um modelo cliente-servidor, onde os clientes solicitam dados dos servidores e os servidores respondem com os dados solicitados (LEE; KIM; HONG, 2021).

A arquitetura do OPC UA é composta por vários componentes, incluindo o Modelo de Informação, os Serviços de Mensagem, a Camada de Transporte e a Camada de Segurança. Cada um desses componentes desempenha um papel fundamental na comunicação de dados por meio do protocolo OPC UA (MAHNKE; LEITNER; DAMM, 2009).

O protocolo OPC UA é baseado em um modelo de informação que descreve a estrutura e os atributos dos dados que são compartilhados entre os sistemas. Este modelo é organizado em uma hierarquia de nós, que representam os diferentes elementos e componentes dos sistemas e dispositivos em uma rede industrial (MAHNKE; LEITNER; DAMM, 2009).

Os nós no modelo de informação do OPC UA são organizados em uma estrutura de árvore, onde cada nó é identificado por um caminho exclusivo chamado de Identificador de Objeto (*Object Identifier* ou *Node ID*). Os nós podem ser organizados em diferentes tipos de hierarquias, dependendo do tipo de informação que eles representam (KATTI et al., 2018).

Existem vários tipos de nós no modelo de informação do OPC UA, incluindo nós de objeto, nós de variável, nós de método, nós de tipo e nós de *view*. Cada um desses tipos de nós representa um tipo diferente de informação que pode ser compartilhada entre os sistemas (CAVALIERI; CUTULI, 2010).

Os nós de objeto representam componentes físicos ou lógicos dos sistemas, como equipamentos, sistemas de controle ou outras entidades. Eles podem incluir informações sobre o estado, a configuração ou outras propriedades do objeto (CAVALIERI; CUTULI, 2010).

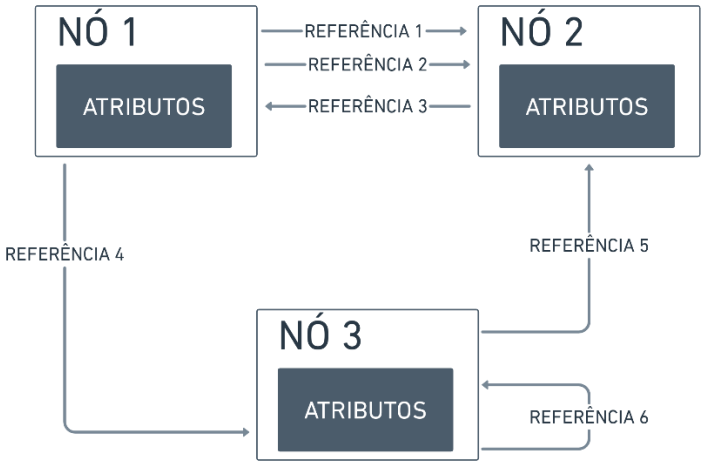
Os nós de variável representam os valores dos dados que estão sendo compartilhados entre os sistemas. Eles podem incluir informações como valores de sensores, status de alarmes ou outras informações que descrevem o estado de um sistema (MAHNKE; LEITNER; DAMM, 2009).

Os nós de método representam operações que podem ser executadas nos sistemas, como operações de controle ou diagnóstico. Eles podem ser usados para iniciar ou parar processos, executar diagnósticos ou outras ações que afetam o funcionamento do sistema (MAHNKE; LEITNER; DAMM, 2009).

Os nós de tipo representam os tipos de objetos, variáveis e métodos que estão sendo compartilhados entre os sistemas. Eles podem ser usados para definir a estrutura e os atributos dos dados que estão sendo trocados (MAHNKE; LEITNER; DAMM, 2009).

Os nós de *view* representam subconjuntos dos nós em uma hierarquia de nós. Eles podem ser usados para criar visualizações personalizadas dos dados para diferentes usuários ou aplicativos (MAHNKE; LEITNER; DAMM, 2009). A Figura representa um exemplo da configuração de nós.

Figura 7 - Nós e referência OPC UA



Fonte: Adaptado de Mahnke, Leitner e Damm (2009)

Na Figura 7, o Nó 1, Nó 2 e Nó 3, todos contendo atributos, estão conectados com várias referências (referência 1-6). O servidor retorna *NodeIds* ao navegar ou consultar o endereço. O espaço e os clientes usam o *NodeId* para endereçar os nós nas chamadas de serviço. Um Nó pode ter vários *NodeIds* alternativos que podem ser usados para endereçar o *Node* (MAHNKE; LEITNER; DAMM, 2009).

O *NodeId* canônico pode ser obtido lendo o atributo *NodeId*, mesmo que o *Node* tenha sido acessado por um *NodeId* alternativo. Os *NodeIds* contêm um *Namespace*, permitindo que diferentes autoridades de nomenclatura definam os *NodeIds* de maneira exclusiva. Autoridades de nomeação podem sejam organizações,

fornecedores ou sistemas (MAHNKE; LEITNER; DAMM, 2009). Detalhes sobre o *NodeId* e seu tipo de dados são descritos na Tabela 1.

Tabela 1 - Atributos comuns de nós OPC UA

Atributo	Tipo de dado	Descrição
NodeId	*NodeId*	Identifica exclusivamente um nó em um servidor OPC UA e é usado para endereçar o nó nos serviços OPC UA
NodeClass	*NodeClass*	Uma enumeração que identifica o *NodeClass* de um nó, como objeto ou método
BrowseName	*QualifiedName*	Identifica o Nó ao navegar pelo Servidor OPC UA
DisplayName	*LocalizedText*	Contém o Nome do Nó que deve ser usado para exibir o nome em uma interface de usuário
Description	*LocalizedText*	Este atributo opcional contém uma descrição textual do nó
WriteMask	*UInt32*	É opcional e especifica quais Atributos do *Node* são graváveis, ou seja, podem ser modificados por um cliente OPC UA
UserWriteMask	*UInt32*	É opcional e especifica quais Atributos do *Node* pode ser modificado pelo usuário atualmente conectado ao servidor

Fonte: Adaptado de Mahnke, Leitner e Damm (2009)

Desse modo, através do protocolo de comunicação OPC UA, todos os dispositivos num sistema de produção deverão ser capazes de se comunicarem entre si, utilizando um padrão de comunicação compatível com as definições da Indústria 4.0 (SILVA, 2021).

2.7 ICONICS GENESIS64

O pacote ICONICS GENESIS 64 fornece meios para desenvolvimento de supervisórios e integração de todas as plantas, disponibilizando dados em uma plataforma distribuída em tempo real. É uma completa solução de software para todas as aplicações HMI (*Human Machine Interface*) e SCADA. Isso é usado para uma variedade de aplicações como HMI/SCADA, Automação Predial, Historiador de Dados da Planta e muito mais. Ele fornece GraphWorX^{TM}64, um pacote de software HMI, para projetar gráficos altamente interativos e animados para sistemas de controle de

processos. É uma tela onde se pode exibir em tempo real e processar dados através de qualquer fonte de dados compatível com OPC UA (QASIM et al., 2020).

O sistema SCADA Genesis 64 oferece uma ampla variedade de recursos e funcionalidades para o monitoramento e controle de processos industriais. Alguns dos principais recursos, de acordo com ICONICS (2023), incluem:

- Interface gráfica intuitiva e fácil de usar, com suporte a gráficos em duas dimensões (2D) e três dimensões (3D).
- Suporte para uma ampla variedade de dispositivos e protocolos de comunicação, incluindo OPC, OPC UA, MQTT, Modbus, DNP3 (*Distributed Network Protocol*), entre outros.
- Alarmes e eventos personalizáveis, com suporte a notificações via e-mail, SMS ou outros meios.
- Recursos avançados de análise de dados, incluindo tendências históricas, gráficos de desempenho e relatórios personalizados.
- Recursos de segurança avançados, incluindo autenticação de usuário, criptografia e controle de acesso baseado em funções.
- Capacidade de integração com outros sistemas, incluindo sistemas de gerenciamento de energia (EMS), sistemas de gerenciamento de edifícios (BMS) e sistemas de gerenciamento de ativos (AMS).

GraphWorX^{TM}64 permite que você crie seus próprios símbolos (HMI objetos) ou use da Biblioteca de Símbolos disponível no GENESIS64. Ele fornece objetos estáticos e dinâmicos para criar exibições animadas. A biblioteca de símbolos GraphWorXTM64 fornece formas básicas como linha, círculo, elipse, quadrado, retângulo, arco e polígono. Além dos símbolos básicos, permite você adicionar controles deslizantes, mostradores, medidores, indicadores, botões, luzes, relógio e outros objetos baseados em dados em tempo real (QASIM et al., 2020).

Organizar a recolha e tratamento dos dados, bem como o seu arquivo no GENESIS64 existem várias maneiras: os dados atuais podem ser coletados e visualizados por saída direta para a tela do operador em forma numérica ou em forma de gráfico. O arquivamento dos dados, bem como o seu tratamento matemático (filtragem, determinação média, valor mínimo/máximo, RMS (*Root Mean Square*), expectativa matemática e variação) pode ser executado usando o componente

TrendWorX64 Logger ou o aplicativo de coleta de dados de alta velocidade Hyper Historian (MIKHAYLOVNA; ALEKSANDROVNA, 2019).

Além do GraphWorX64, já descrito anteriormente, outros recursos do pacote GENESIS64 são importantes para o desenvolvimento de uma solução completa. De acordo com Faritovich (2016), outras ferramentas disponíveis no software supervisório GENESIS64 são:

- AlarmWorX64: servidor de alarmes e eventos que permite que se responda a problemas de forma rápida e eficiente com gerenciamento avançado alarmes e alertas.
- EarthWorX: Sistema Integrado de Informação Geográfica para visualização em tempo real de sistemas de controle de processos automatizados com referência a coordenadas geográficas de sistemas distribuídos.
- O Hyper Historian: um historiador altamente confiável e poderoso, permite armazenar dados históricos, altamente escaláveis com suporte para conectividade de dados OPC padrão da indústria.
- TrendWorX64: componente integrado para coletar e analisar históricos dados em tempo real com suporte de agregação.
- Workbench64: um sistema centralizado para iniciar, configurar e implantação de aplicações de grande escala.
- ReportWorX: um sistema de documentação (criação, execução, relatórios de redirecionamento), interface Web. Integração de dados do MS SQL Server, MSDE (*Microsoft SQL Server Desktop Engine*), Access, Oracle, SAP, etc.
- AssetWorX: uma tecnologia inteligente de ativos que permite usuários para organizar todos os dados de instalações e equipamentos em classes reutilizáveis (ativos) que podem convenientemente ser usado para todos esses ativos ao construir o sistema. Em outras palavras, o componente permite simplificar o desenvolvimento e melhorar navegação.
- GridWorX: um componente que permite integrar facilmente dados de qualquer rede ou banco de dados conectado e fornece visualização de malha em tempo real.

- FDDWorX: um componente de detecção e diagnóstico de falhas previsíveis com a possibilidade de operação contínua de qualquer equipamento.
- Web HMI: software de automação baseado na web em tempo real

É importante mencionar que o software Genesis64 SCADA não é um software *open source*, o que significa que o código-fonte não é aberto ao público e não está disponível para modificações ou distribuição (ICONICS, 2023).

Além disso, é importante notar que o Genesis64 SCADA é um software proprietário que só roda em sistemas operacionais Windows, o que limita a sua compatibilidade com outros sistemas operacionais como MacOS, Linux e outros (ICONICS, 2023).

2.8 Power BI

O Power BI oferece a capacidade de analisar e apresentar dados, moldar e entregar os resultados. Tudo isso pode ser alcançado em uma fração do tempo que seria necessário para especificar, desenvolver e testar uma solução corporativa (ASPIN, 2016).

O Power BI Desktop é um software que permite conectar, transformar e visualizar dados. Vamos nos aprofundar em alguns detalhes. O Power BI Desktop compreende seus próprios componentes. Os dois que são mais importantes para um iniciante em Power BI são a tela do Power BI e o Power Query (ARNOLD, 2023).

O Power Query permite que você execute o processamento ETL nos dados. ETL, do inglês *Extract*, *Transform*, *Load* (Extrair, Transformar, Carregar), são ferramentas de software cuja função é a extração de dados de diversos sistemas, transformação desses dados conforme regras de negócios e por fim o carregamento dos dados e geralmente descreve o processo no qual dados de fontes de dados que podem ter estruturas diferentes são combinados e carregados (LOTH; VOGEL, 2022).

O principal objetivo do BI é obter insights acionáveis que levem a decisões mais inteligentes e melhores resultados de negócios. Outra maneira de classificar o BI é de uma perspectiva de tempo. Então podemos identificar três tipos de análise de dados: descritiva, preditiva e prescritiva (LACHEV, 2022).

O Power BI possibilita a análise de dados sem experiência técnica ou conhecimento de programação. A chave para isso é o conceito operacional, que converte ações do mouse, como arrastar e soltar, em consultas de dados. Isso permite que o usuário obtenha insights rapidamente dos dados e os compartilhe com outras pessoas (LOTH; VOGEL, 2022).

Apesar de permitir a análise de dados sem a necessidade de programação, o Power BI possui uma linguagem específica, chamada DAX (*Data Analysis Expressions*). A linguagem DAX é a linguagem de fórmulas utilizada pelo Power BI, uma ferramenta de *Business Intelligence* da Microsoft. Com o DAX, é possível criar cálculos e expressões para análises de dados em tabelas e gráficos. O DAX é uma linguagem de fórmulas que combina funções, operadores e constantes para criar expressões que permitem a criação de medidas, colunas calculadas e tabelas calculadas. Com a sua sintaxe intuitiva e poderosa, o DAX torna a análise de dados mais fácil e eficiente para os usuários do Power BI (HYMAN, 2022).

A Microsoft criou várias APIs e SDKs (*Software Development Kit*) que permitem a criação de visuais personalizados, conectores de fonte de dados e automação via PowerShell e outras linguagens de codificação. Como resultado, há um grande ecossistema estendido de visuais personalizados de terceiros, conectores, aplicativos e produtos complementares para o Power BI. Além disso, o Power BI se integra a outras linguagens de programação não pertencentes à Microsoft, como Python, R e *Scalable Vector Graphics* (SVG) (DECKLER, 2021).

Atualmente, a maioria das soluções de BI são consumidas por meio de uma interface da web. Um consumo típico de relatório experiência envolve não apenas abrir um relatório, mas também interagir com ele. No Power BI termos, isso se traduz em abrir um relatório e, em seguida, interagir com filtros, segmentações e relatórios visuais e navegação para outras páginas explicitamente ou por meio de marcadores e *drill* (MERCHANT, 2022).

Ademais, o Power BI é uma ferramenta da Microsoft que tem licença proprietária, ou seja, não é open source. Existem diferentes planos de licenciamento para o Power BI, incluindo uma versão gratuita com limitações de recursos, uma versão paga com mais recursos e acesso a serviços adicionais na nuvem, e uma versão para empresas com recursos de governança e segurança avançados (MICROSOFT, 2023).

2.9 Grafana

O Grafana é um aplicativo web amigável, baseado na criação de painéis visuais para expor gráficos e tabelas que são construídos em torno dos dados provenientes da infraestrutura de IoT. Ele também fornece a funcionalidade de criar alertas definindo regras com base no fluxo de dados de entrada, portanto, fornecendo algum grau de operação automática do ambiente IoT (JAPÓN, 2022).

Como uma ferramenta de código aberto amplamente utilizada, o Grafana pode ser implantado de um número quase ilimitado de maneiras e em escalas que variam de uma única instância em um Raspberry Pi de bolso até implantações multirregionais altamente disponíveis com centenas de nós (MCCOLLAM, 2022).

Além de permitir a instalação local, é possível também evitar todo o trabalho relacionado à instalação e configuração do Grafana, ao usar o serviço Grafana Cloud. Ao criar uma conta no Grafana Cloud pode-se também utilizar a plataforma (HERNANDEZ, 2022).

Embora existam muitas soluções no espaço de visualização de dados, Grafana está provando ser uma das mais empolgantes, exibindo rápido crescimento em escopo e recursos, amplas opções de implantação e suporte e uma comunidade entusiasmada que contribui para seu crescimento futuro (SALITURO, 2020).

Como o Grafana é amplamente utilizado e tem uma enorme comunidade por trás, muitos painéis são criados por essa comunidade, a qual fornece um serviço onde os usuários registrados podem publicar seus painéis em uma galeria, e qualquer pessoa pode baixá-los e instalá-los em suas próprias instâncias do Grafana (BASTOS; ARAÚJO, 2019).

Em muitas aplicações, necessita-se observar os dados ao longo do tempo, ou mesmo em tempo real, pois podem representar dados críticos. Se os dados passarem a ser preocupantes, talvez precisemos ser notificados imediatamente (SALITURO, 2020). Embora existam várias ferramentas poderosas de análise de dados no mercado que cumprem essas funções, de acordo com Salituro (2020) o Grafana possui vários recursos que o tornam uma escolha atraente:

- Rápido: o *back-end* do Grafana é escrito na nova linguagem Go do Google, tornando-o extremamente eficiente ao consultar fontes de dados ou alimentar milhares de pontos de dados para vários painéis de painel.

- Aberto: o Grafana oferece suporte a um modelo de plug-in para seus painéis de painel e fontes de dados. O número de plugins está crescendo constantemente, pois a comunidade Grafana contribui com entusiasmo para o projeto.
- Bonito: o Grafana aproveita a atraente e poderosa biblioteca D3. Muitas das ferramentas de painel populares, como DataDog e Zabbix, podem gerar rapidamente belos gráficos de milhares de canais de dados, mas oferecem apenas um controle limitado sobre os elementos de exibição. O Grafana fornece controle refinado sobre a maioria dos elementos gráficos, incluindo eixos, linhas, pontos, preenchimentos, anotações e legendas. Ele ainda oferece o tão procurado modo escuro.
- Versátil: o Grafana não está vinculado a uma tecnologia de banco de dados específica. Por exemplo, Kibana é um membro poderoso e conhecido da pilha *ElasticSearch, Logstash, Kibana* (ELK); ele só é capaz de visualizar as fontes de dados do *ElasticSearch*. Isso lhe dá a vantagem sobre o Grafana de uma melhor capacidade de integrar as ferramentas de análise do *ElasticSearch* em seus painéis gráficos. No entanto, devido à sua arquitetura de plug-in, o Grafana pode suportar uma variedade de bancos de dados cada vez maiores (na última contagem, mais de 30), desde *Relational Database Management System* (RDBMs) tradicionais, como MySQL e PostgresQL, até TSDBs (*Time Series Database*) modernos, como InfluxDB e Prometheus. Além de cada gráfico exibir dados de várias fontes de dados, um único gráfico também pode combinar dados de várias fontes de dados.
- Gratuito: Embora sejam ferramentas muito poderosas, o DataDog e o Splunk são pacotes comerciais e, como tal, cobram taxas para gerenciar todos os conjuntos de dados, exceto os menores. Se você quiser experimentar, o Grafana está disponível gratuitamente sob a licença de código aberto Apache e, se você planeja executá-lo em empresa, pode-se adquirir suporte em camadas.

Desse modo, o Grafana é uma plataforma de análise e visualização de dados amplamente utilizada na Indústria 4.0 e em projetos de IoT (Internet das Coisas). Com interface intuitiva e personalizável, o Grafana permite monitorar e analisar dados em

tempo real de sensores, dispositivos e sistemas, facilitando a tomada de decisões e a identificação de anomalias. Na Indústria 4.0, o Grafana pode ser usado para monitorar a produção, prever falhas em equipamentos e melhorar a eficiência energética. Já em projetos de IoT, o Grafana é uma ferramenta poderosa para visualizar dados de sensores e atuadores em tempo real, identificar tendências e padrões, e tomar decisões baseadas em dados precisos e atualizados (CARREIRO; NASCIMENTO; SOUSA, 2022).

2.10 Aplicação Web com Python

A criação de interfaces web é uma das áreas mais importantes no desenvolvimento de aplicações modernas. Python é uma linguagem muito popular para a criação de aplicações web e oferece diversas opções de *frameworks* para desenvolvimento de interfaces (UZAYR, 2022).

O desenvolvimento de interfaces web com Python envolve a utilização de um *framework* web, que fornece um conjunto de ferramentas e bibliotecas para facilitar a criação de aplicações web (AGRAWAL, 2022).

Ou seja, um *framework* pode ser entendido como um componente que auxilia no desenvolvimento de aplicações, sem ter que criar do zero. Na linguagem Python, os *frameworks* mais populares são Django e Flask (GOMES; SERUFFO; SILVEIRA, 2024).

A escolha do framework é uma etapa muito importante na elaboração de um projeto, pois uma escolha errada pode significar um consumo de armazenamento, memória e processamento desnecessário, ou mesmo outros problemas que poderiam ser evitados. Por isso, é importante conhecer a natureza do projeto, pensar no produto e no valor que se deseja gerar, e então ponderar os critérios de escolhas (GOMES; SERUFFO; SILVEIRA, 2024).

2.10.1 *Framework* Flask

Flask é um *framework* web leve e flexível para o desenvolvimento de aplicações web em Python. Ele é uma excelente opção para quem deseja criar aplicações web

de forma rápida e fácil, sem a complexidade de outros *frameworks* mais robustos (SINGH, 2021).

Uma das principais características do Flask é a sua simplicidade. Ele possui uma estrutura de código simples, facilitando o aprendizado e a utilização por desenvolvedores iniciantes. Além disso, o Flask é altamente modular, permitindo que os desenvolvedores escolham as bibliotecas que desejam usar em cada projeto, o que proporciona uma maior flexibilidade (ARONOWITZ, 2021).

Outra vantagem do Flask é a sua facilidade de extensão. Ele possui uma grande variedade de extensões disponíveis, que oferecem uma ampla gama de funcionalidades para a criação de aplicações web, incluindo autenticação de usuários, manipulação de formulários, integração com bancos de dados, entre outras (PERALTA, 2023).

O Flask pode ser aplicado em soluções de IoT no contexto da Indústria 4.0 para criar aplicações web para monitorar e controlar dispositivos conectados. Na Indústria 4.0, a IoT desempenha um papel importante ao conectar dispositivos, sensores e sistemas para coletar dados em tempo real e permitir a tomada de decisões mais robustas e rápidas (PATIL; VIJAYALASHMI; TAPASKAR, 2017).

Com o Flask, é possível desenvolver aplicações web que se comunicam com os dispositivos IoT, permitindo o monitoramento e o controle de parâmetros, tais como temperatura, umidade, vibração e consumo de energia. Esses dados podem ser coletados por sensores e transmitidos por meio de protocolos como MQTT, CoAP e HTTP para a aplicação Flask, que os processa e exibe em tempo real para o usuário (HOSSAIN et al., 2022).

Um exemplo de aplicação do Flask em IoT no contexto da Indústria 4.0 é o controle de qualidade em uma linha de produção. Sensores instalados na linha de produção podem coletar dados como a qualidade do produto, a velocidade da linha, a temperatura, entre outros. Esses dados são transmitidos para a aplicação Flask, que os processa e os exibe para o usuário em tempo real, permitindo que o usuário tome decisões e ajuste a produção conforme necessário (YUKHIMETS; YUDINKOV, 2019).

Outro exemplo é a gestão de ativos em uma fábrica. Dispositivos IoT, como sensores de vibração e sensores de consumo de energia, podem coletar dados dos equipamentos de produção, transmitindo-os para a aplicação Flask. A aplicação pode, então, analisar os dados em tempo real para identificar possíveis problemas e notificar

o usuário para a manutenção preventiva, evitando assim paradas de produção e reduzindo o tempo de inatividade (ABHISHEK; BHASKER; PONRAJ, 2021).

2.10.2 Dash

Dash é um framework web de código aberto desenvolvido pela empresa Plotly que permite a criação de aplicações web interativas em Python. Com o Dash, é possível criar painéis de dados interativos, relatórios dinâmicos, *dashboards* e muito mais. Ele oferece uma ampla variedade de componentes interativos, como gráficos, tabelas, *sliders*, *dropdowns* e caixas de texto, que podem ser personalizados e integrados em uma única página da web (SCHROEDER; MAYER; WARD, 2022).

Uma das principais vantagens do Dash é a sua facilidade de uso. Com apenas algumas linhas de código, é possível criar uma aplicação web interativa e visualmente atraente. Além disso, o Dash é baseado em Flask, o que permite uma maior flexibilidade e personalização da aplicação (DABBAS, 2021).

O Dash também oferece uma ampla variedade de opções de personalização. É possível personalizar a aparência e o comportamento dos componentes interativos, bem como integrar gráficos personalizados e outros elementos de design. Com a possibilidade de criar estilos próprios com CSS (*Cascading Style Sheet*) é possível criar uma experiência única e personalizada para os usuários (SCHROEDER; MAYER; WARD, 2022).

2.10.3 Bootstrap

Bootstrap é uma biblioteca de *front-end* de código aberto que permite a criação de páginas da web responsivas e móveis. Com o Bootstrap, é possível criar *layouts* de página flexíveis, componentes de interface do usuário, modelos de página e muito mais. Ele oferece uma ampla variedade de componentes pré-construídos, como botões, formulários, modais e barras de navegação, que podem ser facilmente personalizados e integrados em uma aplicação web (UZAYR, 2023).

No contexto do Dash, o Bootstrap pode ser usado para criar *layouts* responsivos e componentes de interface do usuário. O Dash Bootstrap Components é um pacote que permite o uso dos componentes do Bootstrap no Dash, permitindo a

criação de uma interface do usuário visualmente atraente e responsiva. Com este pacote, é possível adicionar barras de navegação, menus *drop-down*, botões de ação, tabelas, formulários e muito mais (DABBAS, 2021).

O uso do Bootstrap no Dash permite que o desenvolvedor se concentre na lógica da aplicação e na visualização de dados, enquanto o Bootstrap cuida do layout e dos componentes da interface do usuário. Isso torna o processo de criação de aplicações web em Dash muito mais eficiente e produtivo (SCHROEDER; MAYER; WARD, 2022).

2.10.4 Plotly

Plotly é uma biblioteca de visualização de dados interativa que permite a criação de gráficos, mapas e outros tipos de visualizações em Python, criado pela empresa Plotly. A biblioteca é baseada em JavaScript e permite a criação de visualizações interativas que podem ser facilmente integradas em uma aplicação web (SCHROEDER; MAYER; WARD, 2022).

No contexto do Dash, o Plotly é frequentemente usado para criar gráficos interativos que permitem a análise e exploração de dados em tempo real. O Dash Plotly é um pacote que permite o uso dos gráficos Plotly em uma aplicação Dash. Com este pacote, é possível criar gráficos de barras, gráficos de linha, gráficos de dispersão, gráficos de caixa, gráficos de superfície e muito mais (DABBAS, 2021).

O uso do Plotly no Dash permite a criação de visualizações interativas que permitem ao usuário explorar e analisar os dados de forma dinâmica. Com o pacote Dash Plotly, é possível criar gráficos que respondem a eventos, como seleções do usuário, filtragem de dados e atualizações em tempo real. Além disso, o Plotly também oferece recursos de animação, que permitem a criação de gráficos dinâmicos e atrativos (SCHROEDER; MAYER; WARD, 2022).

O Dash Plotly também é altamente personalizável, permitindo que o desenvolvedor personalize a aparência dos gráficos, incluindo fontes, cores, títulos e muito mais. Com a combinação do Plotly e Bootstrap, é possível criar uma interface do usuário visualmente atraente e responsiva (SCHROEDER; MAYER; WARD, 2022).

2.10.5 Arquitetura da Aplicação Web

Após abordado os conceitos sobre os principais elementos necessários para o desenvolvimento de uma aplicação web baseada em linguagem Python, faz-se necessário mesclar essas bibliotecas para desenvolver uma arquitetura completa. A priori, faz-se necessário estabelecer uma compreensão básica dos componentes e fases envolvidos. A Figura 8 mostra uma simples sequência pela qual as solicitações e respostas fluem quando um usuário acessa o aplicativo de um navegador (DABBAS, 2021).

Figura 8 - Componentes de um aplicativo em servidor

Fonte: Autor (2023)

A solicitação começa à esquerda da Figura 8 e, em seguida, passa por vários componentes até chegar ao Dash, que executa o script app.py. Em seguida, o código da aplicação gera uma resposta que passa pelos mesmos componentes na direção oposta, até chegar ao navegador do usuário (DABBAS, 2021).

De acordo com Dabbas (2021), o processo inicia-se no navegador, que significa qualquer protocolo de transferência de hipertexto (HTTP) cliente. Quando o usuário insere um localizador uniforme de recursos (URL), o navegador faz uma solicitação ao servidor da web. O trabalho do servidor da Web é lidar com as solicitações recebidas. O problema é que o servidor não executa o código Python, então deve haver alguma maneira de receber as solicitações, interpretá-las e retornar as respostas, e o recurso necessário para tornar isso factível, o que é feito pelo WSGI Server.

Web Server Gateway Interface (WSGI) é um conjunto de convenções recomendadas para serem implementadas pelo software de servidor da Web para encaminhar a solicitação de um cliente para um aplicativo da Web escrito em Python

ou em uma estrutura da Web baseada em Python. É uma interface independente de implementação entre o servidor da web e um aplicativo da web. As especificações do padrão WSGI foram desenvolvidas com o objetivo de fornecer um terreno comum para aplicativos da Web portáteis escritos em Python (LATHKAR, 2021).

2.11 Algoritmos de Decisão Multicritério

Algoritmos de decisão multicritério são utilizados para ajudar na tomada de decisões que envolvem várias opções avaliadas por múltiplos critérios. Nesses casos, não é possível simplesmente escolher a opção com a melhor pontuação geral, pois a importância relativa de cada critério pode variar e diferentes opções podem ser mais adequadas para diferentes critérios. O método AHP (*Analytic Hierarchy Process*) é um dos mais populares algoritmos de decisão multicritério e é amplamente utilizado em diversas áreas, como negócios, engenharia e ciência da computação (GOMES; SERUFFO; SILVEIRA, 2024).

O AHP é baseado na ideia de que as preferências do tomador de decisão podem ser expressas em termos de comparações em pares. O processo envolve a construção de uma matriz de comparação em pares, na qual as opções são comparadas duas a duas para cada critério. Em seguida, é calculado o vetor de peso para cada critério, que representa a importância relativa de cada critério na tomada de decisão. Por fim, é calculado um vetor de peso para cada opção, que representa a adequação da opção para cada critério (MU; PEREYRA-ROJAS, 2018).

O AHP é uma técnica poderosa para a tomada de decisões multicritério, pois permite a consideração de múltiplos critérios e suas importâncias relativas na tomada de decisão. Além disso, o AHP pode ser usado em diferentes tipos de problemas, como escolha de fornecedores, seleção de projetos, avaliação de desempenho e muito mais (BRUNELLI, 2015).

O AHP também pode ser implementado em software, o que torna o processo de tomada de decisão mais eficiente e produtivo. Existem várias ferramentas de software disponíveis que usam o AHP para ajudar na tomada de decisões multicritério. Essas ferramentas geralmente apresentam uma interface de usuário fácil de usar que permite a construção de matrizes de comparação em pares, cálculo de vetores de peso e análise de resultados (GARAPATI et al., 2022).

2.12 Gestão de Ativos

A Gestão de Ativos é um processo de gerenciamento que busca maximizar o valor dos ativos de uma organização, minimizando os riscos e custos associados a esses ativos. Esses ativos podem incluir equipamentos, instalações, recursos humanos, propriedade intelectual, entre outros (BALZER; SCHORN, 2022).

A Gestão de Ativos é uma abordagem estratégica que ajuda as organizações a melhorar sua eficiência operacional e reduzir custos, ao mesmo tempo em que maximiza o valor dos ativos. O objetivo principal da Gestão de Ativos é garantir que os ativos estejam disponíveis e em condições adequadas para atender às necessidades do negócio (HIGHAM; CHALLENDER; WATTS, 2022).

A Gestão de Ativos é composta por várias etapas, incluindo a identificação e classificação dos ativos, a avaliação dos riscos associados a esses ativos, a implementação de políticas e procedimentos para gerenciar esses ativos e a monitorização e revisão contínuas dessas políticas e procedimentos (BALZER; SCHORN, 2022).

No contexto de Gestão de Ativos, a manutenção é uma atividade fundamental para garantir que os ativos da organização estejam em condições adequadas para desempenhar suas funções de maneira eficiente e eficaz. A manutenção pode ser definida como qualquer atividade que tenha como objetivo preservar ou restaurar um ativo a um estado em que ele possa desempenhar suas funções de maneira satisfatória (HIGHAM; CHALLENDER; WATTS, 2022).

Existem vários tipos de manutenção que podem ser implementados em uma organização, dependendo do tipo de ativo e das necessidades específicas da organização. De acordo com Stephens (2022), os principais tipos de manutenção são:

- Manutenção Corretiva: a manutenção corretiva é realizada após a ocorrência de uma falha no ativo. A manutenção corretiva pode ser planejada ou não planejada. A manutenção corretiva não planejada é geralmente mais cara do que a manutenção corretiva planejada, pois requer uma resposta imediata para resolver o problema.
- Manutenção Preventiva: a manutenção preventiva é realizada com base em um cronograma definido previamente. O objetivo da manutenção preventiva é evitar a ocorrência de falhas no ativo, realizando atividades

de manutenção antes que o problema ocorra. A manutenção preventiva pode incluir inspeções, lubrificação, ajustes, limpeza, entre outras atividades.

- Manutenção Preditiva: a manutenção preditiva é baseada no monitoramento contínuo do ativo para detectar possíveis problemas antes que ocorram falhas. A manutenção preditiva envolve o uso de sensores e outras tecnologias de monitoramento para coletar dados sobre o desempenho do ativo e, em seguida, analisar esses dados para prever quando a manutenção será necessária.

A manutenção preditiva pode ser realizada de duas maneiras: por meio de inspeção pontual, com ferramentas e equipamentos de medição, ou por meio de monitoramento online dos ativos. Na inspeção pontual, os inspetores realizam verificações regulares dos ativos, utilizando ferramentas de medição, como termômetros, medidores de vibração, câmeras termográficas, entre outros, para coletar dados sobre o desempenho do ativo. Esses dados são analisados para identificar tendências e padrões, que podem indicar uma possível falha ou a necessidade de manutenção (CHEBEL-MORELLO, 2017).

Por outro lado, o monitoramento online dos ativos é uma abordagem mais avançada, na qual sensores são instalados nos ativos para coletar dados em tempo real sobre o desempenho do ativo. Esses dados são enviados para um sistema central de processamento, onde são analisados por meio de ferramentas de análise de dados e algoritmos de machine learning, permitindo prever quando uma falha pode ocorrer e quando a manutenção será necessária (PENG, 2020).

A principal diferença entre essas abordagens é que a inspeção pontual requer que os inspetores visitem o local dos ativos para coletar dados, enquanto o monitoramento online permite que os dados sejam coletados continuamente, independentemente da localização geográfica dos ativos. Além disso, a inspeção pontual pode ser perigosa em locais de difícil acesso ou perigosos para os inspetores, como em ambientes industriais com altas temperaturas, altas pressões ou materiais perigosos (BENICIO, 2017).

2.13 Diagnósticos de Falhas em Motores Elétricos

Os motores elétricos são peças fundamentais em diversas aplicações industriais, desde simples ventiladores até grandes sistemas de produção. No entanto, eles também estão sujeitos a falhas que podem causar prejuízos financeiros e interrupções na produção. Por isso, é importante entender as principais causas de falhas em motores e seus efeitos para que a manutenção possa ser realizada de forma preventiva (RODRÍGUEZ, 2020).

Algumas das principais causas de falhas em motores elétricos são desalinhamento, folga nos rolamentos, sobrecarga e harmônicos na rede. E os efeitos mais comuns são aumento de vibração e temperatura (BENICIO, 2017).

O desalinhamento ocorre quando o eixo do motor e o eixo da carga não estão alinhados corretamente, o que pode levar a uma série de problemas, como vibração excessiva, desgaste prematuro dos rolamentos e danos nos acoplamentos (ONÍLIO et al., 2021).

A folga ocorre quando os rolamentos estão desgastados ou mal ajustados, o que pode levar a vibrações excessivas e aumento da temperatura, causando danos ao isolamento do motor e das bobinas (ONÍLIO et al., 2021).

A sobrecarga ocorrer por diversos motivos, como excesso de carga, desequilíbrio na tensão da rede elétrica e falhas no controle de velocidade. A sobrecarga pode levar a uma série de problemas, como aumento de temperatura, vibração excessiva e desgaste prematuro dos componentes do motor (ONÍLIO et al., 2021).

Harmônicos na rede elétrica também podem causar falhas em motores elétricos, os harmônicos podem levar a aumento de temperatura, vibrações excessivas e danos aos componentes do motor (THOMSON, 2020).

Cada uma dessas falhas apresenta um padrão de vibração característico, que pode ser detectado e analisado por meio de equipamentos de medição de vibração. A análise no domínio da frequência é realizada por meio de um espectro de frequência, que é gerado a partir dos sinais de vibração coletados pelo equipamento. Esse espectro permite identificar as frequências predominantes e suas respectivas amplitudes, que indicam a presença de vibrações anormais (THOMSON, 2020).

As falhas em motores elétricos podem levar a interrupções na produção, causando prejuízos financeiros e redução na eficiência do processo produtivo. É importante destacar que a manutenção preventiva e a monitoração contínua são fundamentais para evitar falhas em motores elétricos (HUGHES; DRURY, 2019).

2.14 Sensor de Corrente SCT-013

O sensor de corrente SCT-013 é um dispositivo amplamente utilizado para medir a corrente elétrica em sistemas de baixa e média tensão. Ele é capaz de medir correntes de até 100A, e é muito popular em aplicações de monitoramento de consumo de energia (CONTI et al., 2022).

De acordo com (YILDIZ; BURUNKAYA, 2019), o SCT-013 é um tipo de sensor de corrente não invasivo, o que significa que ele não precisa ser conectado diretamente ao fio condutor para medir a corrente. Em vez disso, o sensor é instalado em torno do cabo, permitindo que a corrente seja medida sem interromper o circuito, conforme ilustrado na Figura 9.

Figura 9 - SCT-013

Fonte: Autor (2023)

O funcionamento do SCT-013 é baseado no princípio do transformador de corrente. O sensor consiste em um núcleo de ferrite e um enrolamento secundário. Quando a corrente flui pelo condutor, um campo magnético é gerado em torno do cabo, que é detectado pelo enrolamento secundário do sensor. Esse enrolamento é

projetado para fornecer uma tensão proporcional à corrente medida, que pode ser lida por um microcontrolador ou outro dispositivo eletrônico (CONTI et al., 2022).

Uma das vantagens do SCT-013 é a sua facilidade de uso. Ele pode ser facilmente instalado em qualquer cabo elétrico, sem a necessidade de interromper o circuito. Além disso, como não há contato direto com o cabo, não há risco de choque elétrico durante a instalação ou uso (MIRON-ALEXE, 2016).

Outra vantagem do SCT-013 é a sua alta precisão. Ele é capaz de medir correntes com precisão de até 1%, o que o torna ideal para aplicações que requerem medições precisas de corrente elétrica (KHWANRIT et al., 2018).

O SCT-013 pode ser usado em uma ampla variedade de aplicações, incluindo monitoramento de consumo de energia em casas e edifícios comerciais, monitoramento de corrente em sistemas de energia solar, monitoramento de corrente em motores elétricos e outros equipamentos elétricos (WAHLBRINCK, 2018).

2.15 Sensor MPU-6050

O MPU-6050 é um módulo acelerômetro e girômetro (doravante designado como o seu antecessor analógico, o giroscópio), o qual também é conhecido como Gy-521. Este módulo é baseado no chip MPU6050 da InvenSense, que possui um acelerômetro de 3 eixos e um giroscópio de 3 eixos baseado em tecnologia *Micro Electro Mechanical System* (MEMS), integrados em um único chip. Ele é capaz de medir aceleração linear e angular em tempo real, permitindo que o sistema faça ajustes precisos de orientação e movimento (HASSAN et al., 2019). A Figura 10 ilustra o sensor MPU-6050.

Figura 10 - MPU-6050

Fonte: Autor (2023)

O MPU6050 também é capaz de medir a temperatura ambiente ao seu redor. Ele conta com um sensor de temperatura integrado, que é utilizado para compensar as leituras do acelerômetro e do giroscópio, uma vez que suas medições podem ser influenciadas pela variação de temperatura. Além disso, a temperatura medida pelo MPU6050 também pode ser útil em projetos que envolvem o controle térmico de equipamentos eletrônicos (INVENSENSE INC., 2013).

O MPU6050 é bastante utilizado em projetos de controle de drones, pois permite detectar a inclinação do dispositivo em relação ao solo e fazer ajustes na velocidade e direção de voo, entre outras aplicações (HASSAN et al., 2019).

O giroscópio mede a velocidade angular ou rotação em torno de um eixo específico. Ele funciona com base no princípio de conservação do momento angular, em que uma vez que um objeto começa a girar (ARMENISE et al., 2010). Para o caso do MPU-6050, ele possui três giroscópios vibratórios de taxa MEMS independentes, que detectam a rotação em torno dos eixos X, Y e Z. Quando os giroscópios giram em torno de qualquer um dos eixos dos sentidos, o Efeito Coriolis causa uma vibração que é detectada por um *pickoff* capacitivo. O sinal resultante é amplificado, demodulado e filtrado para produzir uma tensão que é proporcional à taxa angular. Esta tensão é digitalizada usando um chip conversor individual analógico-digital (ADCs) de 16 bits para amostrar cada eixo. A gama completa dos sensores giroscópios podem ser programados digitalmente para ±250, ±500, ±1000 ou ±2000 graus por segundo. A amostra ADC a taxa é programável de 8.000 amostras por segundo, até 3,9 amostras por segundo e selecionável pelo usuário. Os filtros passa-baixa permitem uma ampla gama de frequências de corte (INVENSENSE INC., 2013).

Já o acelerômetro mede a aceleração linear ou a mudança na velocidade de um objeto em uma ou mais direções (ARMENISE et al., 2010). No caso no MPU-6050, O acelerômetro de 3 eixos do MPU-60X0 usa massas de prova separadas para cada eixo. Aceleração ao longo de um eixo particular induz deslocamento na massa de prova correspondente, e sensores capacitivos detectam o deslocamento diferencial. A arquitetura do MPU-60X0 reduz a suscetibilidade dos acelerômetros a variações de fabricação, bem como à deriva térmica. Quando o dispositivo é colocado em uma superfície plana, ele mede 0g nos eixos X e Y e +1g no eixo Z.

O fator de escala dos acelerômetros é calibrado na fábrica e é nominalmente independente da tensão de alimentação. Cada sensor tem um ADC sigma-delta dedicado para fornecer saídas digitais. A faixa de fundo de escala da saída digital pode ser ajustada para ±2g, ±4g, ±8g ou ±16g.

Em conjunto, o giroscópio e o acelerômetro são capazes de fornecer informações precisas sobre a orientação e movimento de um objeto no espaço tridimensional, permitindo a construção de sistemas de monitoramento e controle mais precisos e confiáveis (CAO, 2023).

2.16 Transformada Rápida de Fourier

A transformada rápida de Fourier (FFT, na sigla em inglês) é um algoritmo que realiza a transformada de Fourier discreta (DFT) de uma sequência de dados, tornando o processo mais eficiente e rápido. A DFT é uma ferramenta matemática que converte uma série temporal de amostras em um espectro de frequência, mostrando a composição das diferentes frequências que compõem o sinal. Isso é útil em muitas aplicações, incluindo processamento de sinais, processamento de áudio e vídeo, comunicações e controle de sistemas (HOLTON, 2020).

O algoritmo FFT divide a sequência de dados em subconjuntos menores e aplica a transformada de Fourier a esses subconjuntos, reduzindo significativamente o número de cálculos necessários. Isso permite que a FFT processe sequências de dados muito grandes em um tempo razoável (SCHWARZINGER, 2022).

A FFT é usada em muitas aplicações em IoT e monitoramento de vibração em motores. Por exemplo, ela pode ser usada para analisar os dados coletados por sensores de vibração em um motor, a fim de identificar padrões de vibração que podem indicar falhas no motor. Essa análise pode ser realizada em tempo real, permitindo que os técnicos de manutenção identifiquem e resolvam problemas antes que eles causem danos mais graves ou interrupções no processo produtivo. Além disso, a FFT pode ser usada para separar os sinais de diferentes componentes do motor, facilitando a identificação da causa raiz de um problema específico (RODRÍGUEZ, 2020).

Segundo Belabed, Jemmali e Souani (2018), chama-se de transformada discreta de Fourier uma sequência de N termos $x(0)$, $x(1)$, ..., $x(n-1)$, a sequência N termos $x(0)$, $x(1)$, ..., $x(n-1)$, definido como:

$$X(p)=\sum_{n=0}^{N-1}x(n)e^{-j2\pi\frac{np}{N}} \tag{1}$$

Praticamente, os N termos $x(n)$ são N amostras de uma amostra sinal analógico $X_n=x(nT_e)$, onde T_e é o período de amostragem, e os N termos $X(p)$ correspondem a uma aproximação de frequência da transformada de Fourier deste sinal para os N pontos.

Desenvolvendo-se a equação 1, a seguinte otimização é obtida:

$$X(p)=\sum_{n=0}^{\left(\frac{N}{2}-1\right)}x_{2n}e^{-j\frac{2\pi}{N}(2n)p}+\sum_{n=0}^{\left(\frac{N}{2}-1\right)}x_{2n+1}e^{-j\frac{2\pi}{N}(2n+1)p} \tag{2}$$

$$X(p)=\sum_{n=0}^{\left(\frac{N}{2}-1\right)}x_{2n}e^{-j\frac{2\pi}{N/2}(np)}+e^{-j\frac{2\pi}{N}p}\sum_{n=0}^{\left(\frac{N}{2}-1\right)}x_{2n+1}e^{-j\frac{2\pi}{N/2}(np)} \tag{3}$$

$$X(p)=A_p+W_N^P*B_p \tag{4}$$

Onde A_p, W_N^P e B_p são respectivamente dados por:

$$A_p=\sum_{n=0}^{\left(\frac{N}{2}-1\right)}x_{2n}e^{-j\frac{2\pi}{N/2}(np)} \tag{5}$$

$$B_p=\sum_{n=0}^{\left(\frac{N}{2}-1\right)}x_{2n+1}e^{-j\frac{2\pi}{N/2}(np)} \tag{6}$$

$$W_N^P=e^{-j\frac{2\pi}{N}p} \tag{7}$$

Como a DFT é periódica, utiliza-se essa especificação para obter:

$$X\left(p+\frac{N}{2}\right)=A_p+W_N^P * B_p \tag{8}$$

A Equação (4) e (8) definem a estrutura básica do algoritmo FFT, onde A_p e B_p são duas entradas complexas, cada uma composta por uma parte real e uma parte imaginária.

Como em várias outras áreas de Processamento Digital de Sinais (DSP, na sigla em inglês), a maioria dos cientistas e engenheiros utilizam eficientes implementações de software do algoritmo FFT, amplamente disponíveis para máquinas de uso geral e, cada vez mais, para microprocessadores e até microcontroladores que incluam núcleos DSP com FFTs codificadas (HOLTON, 2020).

2.17 Filtro de Kalman

O filtro de Kalman é um algoritmo que utiliza uma série de dados observados ao longo do tempo, que contém ruído e outras imprecisões, para estimar variáveis desconhecidas com mais precisão (LI et al., 2015).

De acordo com Lewis, Xie e Popa (2008), para descrever um sistema com ruído, define-se a equação dinâmica estocástica discreta, como:

$$x_{k+1}=Ax_k+Bu_k+Gw_k \tag{9}$$

$$z_k=Hx_k+v_k \tag{10}$$

onde, x_k, z_k, A, B, H, w_k, u_{k-1}, v_k são, respectivamente, o vetor de estados, o vetor de observação, a matriz que processa o vetor de estado e especifica a relação entre o estado atual do sistema e o próximo estado, a matriz que processa o vetor de entradas, a matriz de observação, o vetor de ruído do sistema, o vetor de entrada do sistema e o vetor de ruído de observação.

Vale ressaltar que, para satisfazer a equação, w_k e v_k são assumidos vetores de dimensões compatíveis com as equações de estado e de saída, respectivamente, sendo do tipo Gaussiano, descorrelacionados e cujas matrizes de covariância são simétricas e semi-definidas positivas. Os vetores w_k e v_k podem assumir valores negativos, desde que exibam média nula.

Desse modo, $\overline{w}_k = 0$ e covariância $\overline{w_k w_k^T} = Q$, que é simbolizado como $w_k \sim (0, Q)$. A barra superior denota o valor esperado e o ruído de medição $v_k (0, R)$. Ambos $\{w_k\}$ e $\{v_k\}$ são considerados processos estacionários de ruído branco, ou seja, suas funções de covariância satisfazem:

$$P_{w_{k_1} w_{k_2}} = \overline{w_{k_1} w_{k_2}^T} = Q\delta(k_1 - k_2) \tag{11}$$

onde, $\delta(k)$ é o delta de *Kronecker*, e a versão correspondente para v_k.

Nessas circunstâncias, o estado x_k é uma variável aleatória com média $\overline{x}_k$ e covariância P_{x_k}. Da mesma forma, a saída z_k é aleatória $(\overline{z}_k, P_{z_k})$. Observa-se que $\{x_k\}$ e $\{z_k\}$ são processos estocásticos, em geral não estacionários.

Sendo $\hat{x}_k \in R^n$ definido como a estimativa de estados anterior derivada da equação de transição de estados no momento de $k-1$, e $\hat{x}_{\bar{k}}$ definido como a posterior estimativa de estados de medições no momento de k. Os desvios, ou erros, são descritos por:

$$e_{\bar{k}} = x_k - \hat{x}_{\bar{k}} \tag{12}$$

$$e_k = x_k - \hat{x}_k \tag{13}$$

As equações de desvio da estimação da covariância são definidas como:

$$P_{\bar{k}} = E\left[e_{\bar{k}} \cdot e_k^{-T}\right] \tag{14}$$

$$P_k = E\left[e_k \cdot e_k^T\right] \tag{15}$$

Deve-se, então, encontrar uma equação de estimação de status $\hat{x}_k$ que calcule o estado posterior, para obter as equações do filtro de Kalman. Para isso, faz-se necessário que a fórmula do cálculo de $\hat{x}_k$ seja a combinação linear de estimativa da diferença ponderada entre medições verdadeiras e o valor de previsão medida. As seguintes equações de previsão e atualização da teoria do filtro de Kalman são obtidas. Equações de previsão são definidas da seguinte forma:

$$\hat{x}_{\bar{k}} = A\hat{x}_{k-1} + Bu_{k-1} \tag{16}$$

$$P_{\bar{k}} = AP_{k-1}A^T + Q \tag{17}$$

O Q representa a covariância do ruído de processo w, enquanto R é denotada a covariância do ruído de medição v. Esses valores também podem mudar com o tempo, mas nesse algoritmo eles devem ser constantes, enquanto $P_{\bar{k}}$ representa o erro de covariância estimada no tempo k (TAMAS et al., 2008). As equações de atualização são definidas da seguinte forma:

$$K_k = P_{\bar{k}}H^T\left(HP_{\bar{k}}H^T + R\right)^{-1} \tag{18}$$

$$\hat{x}_{k-1} = \hat{x}_{\bar{k}} + K_k\left(z_k - H\hat{x}_{\bar{k}}\right) \tag{19}$$

$$P_k = \left(I - K_kH\right)P_{\bar{k}} \tag{20}$$

Onde, K_k, $\hat{x}_k$, P_k e I são, respectivamente, a matriz de ganho de Kalman; o valor ótimo do filtro; a matriz densidade espectral, ou matriz de covariância, do erro de estimação de estado; e a matriz identidade.

3 METODOLOGIA

Este capítulo aborda a metodologia utilizada para construir um sistema supervisório SCADA com ênfase em automação IOT. Para atender aos requisitos de segurança, eficiência e simplicidade do sistema, foram adotadas várias etapas. Inicialmente, foi realizado um levantamento dos requisitos do sistema, identificando as principais funcionalidades e características. Em seguida, foi feita a arquitetura do sistema, usando softwares como Fritzing, Proteus e AutoCAD.

O coletor IOT, chamado no trabalho de DAQ-IOT (*Data Acquisition - Internet of Things*), foi projetado com base em um microcontrolador ESP32, que realiza a coleta, processamento e envio de dados de um sensor MPU6050 para medir vibração e temperatura em motores, e três SCT-013 para coleta de corrente trifásica. Os dados são enviados para um broker MQTT instalado em um Raspberry Pi, e a partir dele um servidor lê esses dados.

Para facilitar a simulação, foi criado um circuito de comando auxiliar para ligar um motor através de contactoras e botoeiras, os quais garantem maior segurança para as simulações e testes no projeto.

Além disso, foram configurados os bancos de dados usados, sendo o InfluxDB e o SQL Server, um banco de dados de séries temporais e outro relacional, respectivamente.

Para a implementação do sistema supervisório, fez-se necessário configurar um servidor em Linux e outro em Windows, com a instalação e configuração dos recursos necessários, como InfluxDB, Grafana, Docker, Azure Data Studio, Mosquitto e MQTT Explore. Vale ressaltar que apenas um dos servidores faz-se necessário para o uso na solução final, e a variação de sistemas operacional fez-se apenas para mostrar como usar o sistema em ambos os sistemas operacionais mais utilizados atualmente.

Em seguida, um código em C++ foi elaborado e implantado no ESP32, fazendo-se uso de algoritmos como filtro de Kalman e cálculos de desbalanceamento de fases na coleta das correntes trifásicas. Além disso, é abordada a configuração da transmissão de dados via Wi-Fi com protocolo de comunicação MQTT.

Um código em Python foi desenvolvido para assinar os dados do broker MQTT no Raspberry Pi, e enviá-los via OPC UA, bem como armazená-los no InfluxDB e SQL

Server. Outro código em Python criou a aplicação web em Flask, para exibir os dados do sistema IOT. Também foi elaborado um painel no Grafana para visualizar os dados em tempo real via browser, bem como um painel no supervisório Genesis64 da ICONICS e outro no Power BI.

Finalmente, foram configurados alertas e notificações, por meio de código em Python. Essas etapas são descritas em detalhes ao longo deste capítulo e os códigos disponíveis em https://github.com/leonn3/iotcore.

3.1 Decisão Multicritério

A metodologia utilizada para a construção do sistema SCADA (*Supervisory Control and Data Acquisition*) em aplicação web teve início com a identificação dos requisitos necessários para a aplicação. Para isso, foram aplicados questionários para três grupos distintos: academia, startup e indústria, a fim de traçar um perfil estrutural da aplicação com base nas perspectivas do público analisado.

Em seguida, realizou-se uma análise exploratória dos dados obtidos nos questionários, buscando identificar as principais necessidades e demandas do público-alvo. Além disso, foi realizada uma análise de correlação para verificar a relação entre as diferentes variáveis e requisitos identificados na etapa anterior.

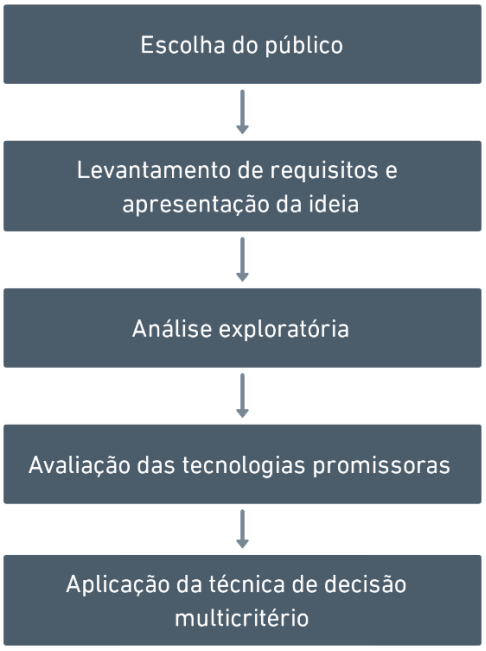
Com base nos resultados dessas análises, foram levantadas técnicas e tecnologias promissoras para serem utilizadas nas diversas etapas da construção da aplicação, em especial a linguagem de programação. Linguagens de programação para escolha foram C#, Java, Python e Delphi, e os critérios performance, confiabilidade, legibilidade, custo, suporte, flexibilidade, simplicidade e experiência do desenvolvedor.

A escolha errada para a construção de uma aplicação pode resultar em problemas, como falta de recursos para expansão e otimização da aplicação, ou mesmo o aumento desnecessário no tamanho do aplicativo. Por isso, a escolha da tecnologia ideal deve levar em conta o problema que se quer resolver e o valor que se deseja gerar.

Para a escolha da linguagem de programação, foi aplicado o método de decisão multicritério AHP (*Analytic Hierarchy Process*), que consiste em uma técnica matemática utilizada para ajudar na tomada de decisão em situações que envolvem

múltiplos critérios. Com base nos resultados obtidos a partir do AHP, definiu-se a linguagem de programação a ser utilizada. Em resumo, a Figura 11 ilustra os passos realizados para essa abordagem das escolhas.

Figura 11 - Passos para aplicação da técnica de decisão multicritério



Fonte: Autor (2023)

Importante ressaltar que o processo de aplicação do método AHP para a escolha da tecnologia mais adequada para a construção do sistema SCADA foi descrito em detalhes em Gomes, Seruffo e Silveira (2024), que faz parte do mesmo projeto. Esse artigo abordou especificamente o uso do AHP na tomada de decisão, desde a formulação do problema até a obtenção dos resultados e interpretação.

3.2 Arquitetura do Sistema

Os requisitos do sistema foram definidos com base nos objetivos do projeto e nas necessidades dos usuários. O sistema deve ser capaz de coletar dados de três sensores de corrente SCT-013 de até 100A e um sensor MPU-6050, que coletam dados de vibração e temperatura; transmitir os dados via MQTT para um *broker* em um Raspberry Pi; visualizar os dados em tempo real em um aplicativo *mobile*; permitir

que os clientes finais visualizem os dados coletados das grandezas medidas no motor de indução.

Desse modo, a arquitetura mínima do sistema foi projetada para atender aos requisitos definidos. O sistema é composto pelos seguintes componentes:

- Três sensores de corrente SCT-013, que mede até 100A;
- Um sensor MPU-6050, o qual coleta dados de vibração e temperatura;
- Um DAQ-IOT, composto de um microcontrolador ESP-32 associado a um circuito básico com componentes eletrônicos;
- Um Raspberry Pi; o qual gerencia o fluxo de comunicação através do protocolo de comunicação MQTT;
- Um servidor que hospeda diferentes aplicações para que os usuários se conectem aos dados aquisitados;
- Clientes finais, que podem acessar os serviços como dashboards, hospedados no servidor, ou acessar diretamente os dados através do *broker* MQTT, como por exemplo uma aplicação mobile.

Os sensores são conectados ao DAQ-IOT, que é responsável por coletar os dados e transmiti-los via MQTT para o Raspberry Pi, o qual serve de broker MQTT. Esse broker é utilizado para receber e distribuir os dados para os *subscribers*, que são o aplicativo *mobile* e o servidor.

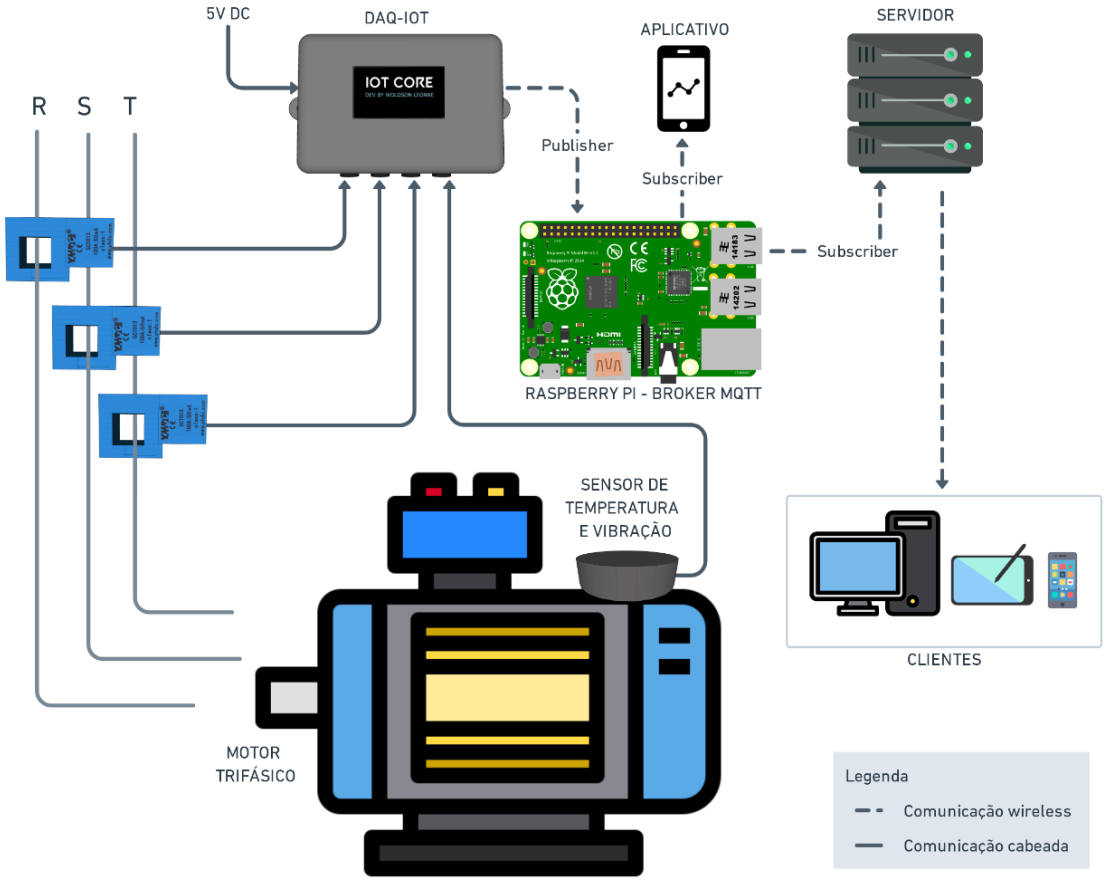
O aplicativo *mobile* permite visualizar os dados em tempo real e o servidor permite que os clientes finais acessem e visualizem também os dados coletados por diferentes serviços e aplicações.

O sistema possui três interfaces: física, de comunicação e de usuário. A interface física é composta pelas conexões entre os sensores e o DAQ-IOT. A interface de comunicação utiliza o protocolo MQTT para transmitir os dados coletados pelos sensores. O protocolo MQTT é utilizado por ser leve, eficiente e seguro. Enquanto a interface de usuário é composta pelo aplicativo *mobile* e pelas aplicações do servidor. O aplicativo *mobile* permite visualizar os dados em tempo real, enquanto o servidor permite que os clientes finais acessem e visualizem os dados coletados.

O aplicativo *mobile* foi utilizado para demonstrar a flexibilidade e facilidade de visualização dos dados coletados pelos sensores, tornando a automação mais moderna e simplificada em comparação com as arquiteturas clássicas e obsoletas.

Com essa arquitetura proposta, é possível obter uma visualização em tempo real dos dados de corrente, vibração e temperatura, de forma distribuída e acessível a partir de diferentes dispositivos, incluindo smartphones e computadores, proporcionando maior praticidade e eficiência no monitoramento do motor de indução. A Figura 12 exemplifica a arquitetura do sistema proposto.

Figura 12 - Arquitetura do Sistema



Fonte: Autor (2023)

Uma das vantagens dessa arquitetura é a transmissão de dados via *wireless*, que proporciona maior mobilidade e flexibilidade na instalação e monitoramento do sistema. Ao eliminar a necessidade de cabos para transmissão de dados, reduz-se a complexidade e os custos de instalação e manutenção do sistema, além de evitar problemas com interferências eletromagnéticas e de conexão.

O uso de cabos para a transmissão de dados pode apresentar diversas desvantagens, tais como: limitações de alcance e mobilidade, interferências

eletromagnéticas, necessidade de instalação de cabos adicionais para cada novo sensor, dificuldade de manutenção e substituição de cabos danificados, entre outros.

Com a transmissão de dados via *wireless*, a arquitetura do sistema proposto permite maior flexibilidade na instalação dos sensores, podendo ser posicionados em locais mais estratégicos e de difícil acesso, sem a necessidade de cabos adicionais. Isso torna o sistema mais eficiente e prático para monitorar o motor de indução em tempo real, sem a necessidade de grandes investimentos em infraestrutura e instalação.

A arquitetura do servidor é um aspecto muito importante do sistema proposto, pois é responsável por gerenciar e armazenar o fluxo de dados. A utilização de várias aplicações para a aquisição, tratamento, armazenamento e transmissão dos dados permite que o sistema seja mais robusto, flexível e escalável, atendendo às necessidades dos usuários finais.

O BRIDGE.py é um *script* em python que desempenha um papel fundamental na arquitetura do servidor. Ele é responsável por adquirir os dados do broker MQTT, assinar as variáveis desejadas, armazenar os dados tanto no SQL Server quanto no InfluxDB e retransmitir esses dados através do protocolo de comunicação OPC UA. Essa retransmissão permite que outras aplicações possam acessar os mesmos dados em tempo real, facilitando a integração com outras ferramentas de automação industrial e oferecendo mais opções de acesso aos dados coletados.

A aplicação em Flask é outra parte importante da arquitetura do servidor, pois permite que os usuários acessem e visualizem os dados coletados através de uma interface web responsiva e moderna. Essa aplicação coleta os dados diretamente do banco de dados SQL Server, oferecendo uma forma simples e eficiente de visualizar os dados em tempo real.

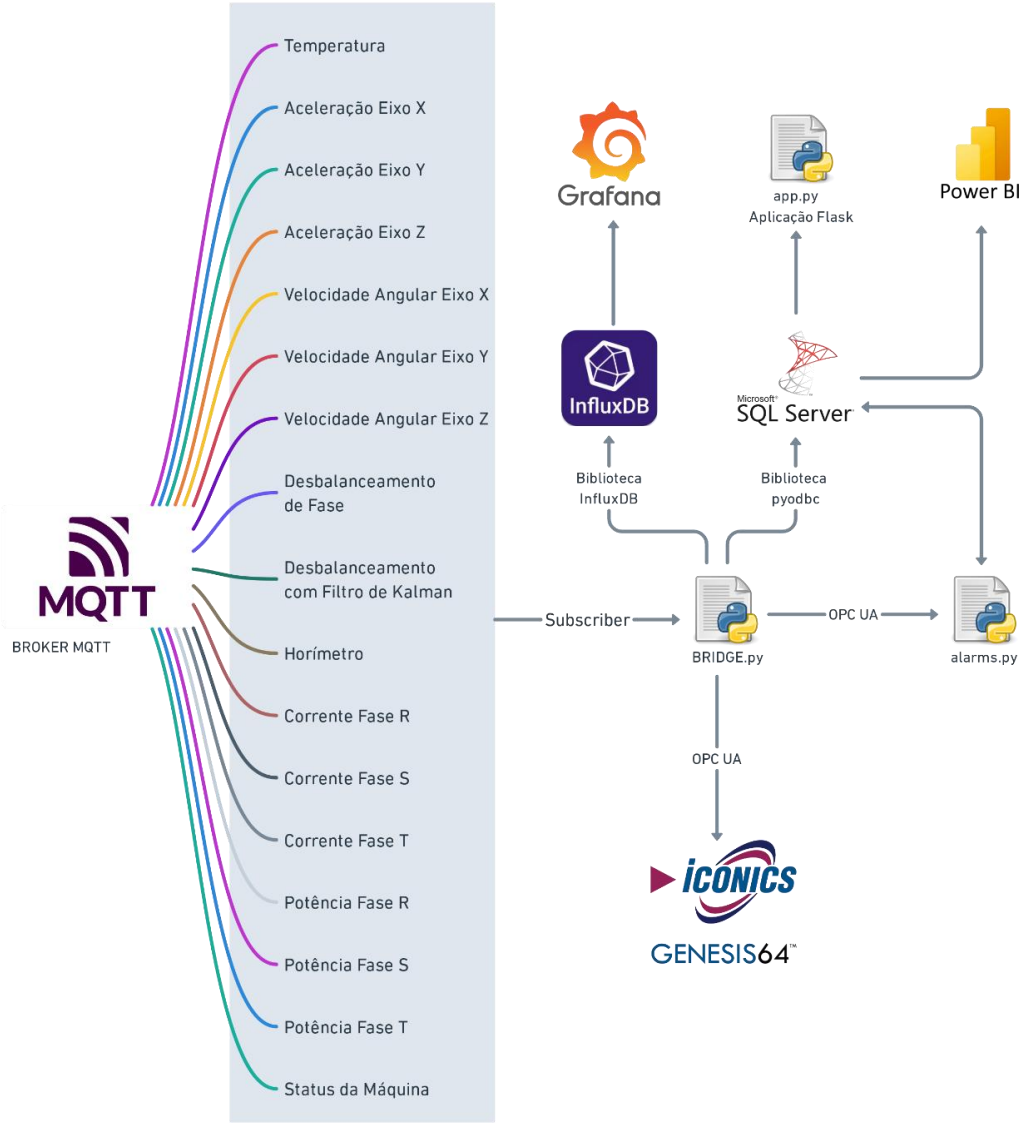
O algoritmo de alarmes é um exemplo de como pode-se utilizar *scripts* em Python para configurar alarmes no sistema. Esse algoritmo lê os dados coletados pelos sensores e, caso detecte alguma anomalia ou valor fora dos limites definidos, envia um e-mail automático informando aos responsáveis.

O Power BI é uma ferramenta de visualização de dados que pode se conectar à base de dados SQL Server para gerar relatórios e gráficos mais avançados, permitindo uma análise mais profunda dos dados coletados.

O Grafana é outra ferramenta de visualização de dados, a qual se conecta ao banco de dados InfluxDB para exibir os dados em tempo real. Ele oferece uma interface gráfica interativa, permitindo que os usuários visualizem e analisem os dados de forma mais dinâmica e personalizada, além de permitir configurar alarmes e notificações via e-mail.

Por fim, o Genesis64 é um supervisório que pode ser utilizado para visualizar os dados coletados em tempo real e realizar o controle do processo industrial. Ele se conecta aos dados através do protocolo OPC UA, permitindo que os usuários monitorem o processo de forma mais completa e eficiente. A Figura 13 ilustra a arquitetura do servidor.

Figura 13 - Arquitetura do servidor



Fonte: Autor (2023)

3.3 Arquitetura do DAQ-IOT

O sistema de coleta de dados, denominado de DAQ-IOT, tem a função de coletar, processar e enviar os dados via MQTT, através de um microcontrolador ESP-32.

Nesse sentido, é de grande importância a definição da arquitetura desse coletor, como os circuitos elétricos, e as configurações de portas do microcontrolador utilizadas pelos sensores.

3.3.1 Configuração do Gy-521 MPU6050

O MPU-6050 é um módulo com um acelerômetro triaxial e um giroscópio triaxial. A relação de pinos entre o microcontrolador e o sensor, bem como a descrição é dada pela Tabela 2.

Tabela 2 - Relação de pinos para conexão MPU-6050 ao ESP-32

Pino MPU-6050	Pino ESP-32	Descrição
VCC	3V3	Alimentação
GND	GND	Referência
SCL	GPIO 22	Comunicação I2C
SDA	GPIO 21	Comunicação I2C
XDA	Não usado	Usado para fazer a interface de outros sensores I2C com o MPU-6050
XCL	Não usado	
AD0	Não usado	Pino para alterar o endereço I2C
INT	Não usado	Pino de interrupção

Fonte: Autor (2023)

O microcontrolador pode ser programado por diversos ambientes de desenvolvimento integrado (IDE), mas para o projeto em questão, optou-se por utilizar a IDE Arduino, por ser simples, leve e intuitiva. Para programar o ESP-32 para realizar a coleta de dados do MPU-6050, fez-se necessário a instalação de algumas bibliotecas, sendo elas: “*adafruit mpu6050*” versão 2.0.3, “*Adafruit Unified Sensor*” versão 1.0.2, e “*Adafruit Bus IO*” versão 1.7.1. A versão da biblioteca é uma referência importante, pois a implementação do código pode mudar a sintaxe de alguns termos conforme a versão utilizada.

3.3.2 Configuração do SCT-013

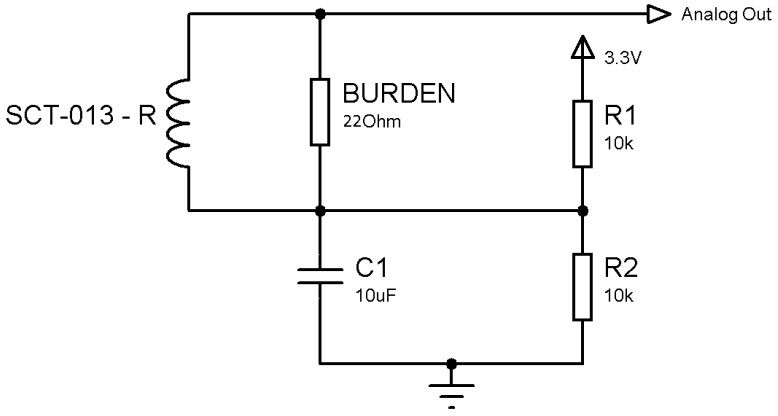
O SCT-013-000 é um sensor capaz de medir corrente elétrica em circuitos de corrente alternada (AC). Ele é composto por um transformador com uma relação de espiras de 1:2000, que gera uma variação de corrente no seu secundário. Essa variação, em conjunto com um resistor de carga (R_L), produz uma variação de tensão que pode ser interpretada como a corrente elétrica a ser medida.

Existem diferentes modelos do sensor SCT-013, com variações na faixa de medição e na razão de enrolamento. No caso do modelo SCT-013-000, ele é capaz de medir correntes elétricas de até 100A. O sensor é projetado com um conector P2 para conexão.

O sensor tem uma relação de 1:2000 e mede até 100A para o modelo escolhido, desse modo, 100A no primário representa 50mA no secundário. Assim, o sensor é ideal para aplicações embarcadas, pois o transformador reduz a corrente no secundário, e essa relação entre variação de corrente pode ser traduzida para tensão, de modo a estabelecer uma relação entre tensão e corrente.

Essa necessidade de um circuito auxiliar para criar a relação entre variação de corrente com variação de tensão, surge decorrente da natureza das entradas do microcontrolador, o qual lê um valor de tensão nos terminais. Assim, faz-se necessária a utilização do circuito da Figura 14.

Figura 14 - Circuito básico para aquisição de corrente



Fonte: Autor (2023)

A função do R1 e R2 é ajustar o ponto zero de medição através da criação de um divisor de tensão. Como a corrente é alternada, a tensão de entrada poderá variar de 0 a 3,3 volts, que é o range da entrada analógica do microcontrolador.

O componente crucial para a medição de corrente elétrica é o resistor de carga, também conhecido como *shunt* ou resistor de *Burden*. Ele permite que utilize-se a lei de Ohm para calcular a corrente. O cálculo do resistor de *Burden* se dá através de:

$$R_B = \left(\frac{\left(\frac{V_s}{2} \right)}{\frac{I_{\max}.1,414}{C_t}} \right) \tag{21}$$

Onde, V_S é a tensão máxima da fonte, I_{max} é a corrente máxima no primário, em RMS, e Ct é o número de espiras. Desse modo, para uma entrada de 3,3V, relação de espiras de 1:2000, e a máxima corrente RMS de 100A, tem-se:

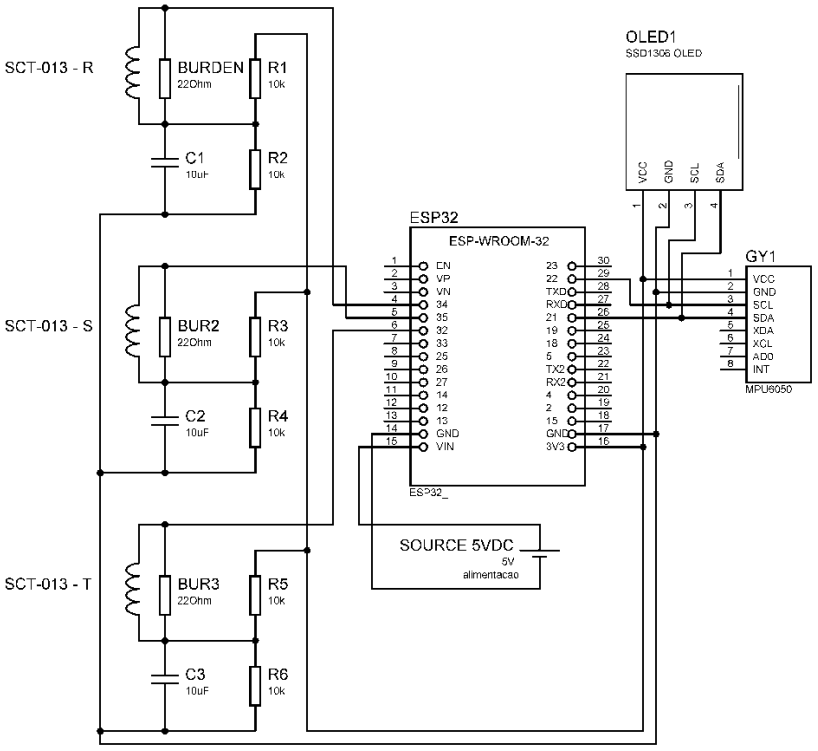
$$R_B = \left(\frac{\left(\frac{3,3}{2} \right)}{\frac{100.1,414}{2000}} \right) = \frac{1,65}{0,0707} = 23,338 \tag{22}$$

Com o valor obtido, pode-se escolher um valor comercial. Para o projeto utilizou-se o resistor de 22 Ohms. Com isso, pode-se facilmente calcular o valor do resistor de *Burden* para outros valores de entrada, como por exemplo Arduino, que opera em 5V, e outros modelos do SCT-013.

3.3.3 Circuito Completo de Aquisição

Para o propósito do DAQ-IOT, faz-se necessário mesclar o esquemático de ligações do sensor MPU-6050 e o circuito de aquisição de corrente com o SCT-013 generalizado para realizar três medições, ou seja, a corrente trifásica do motor. Com isso, obtém-se o circuito completo, conforme ilustrado na Figura 15.

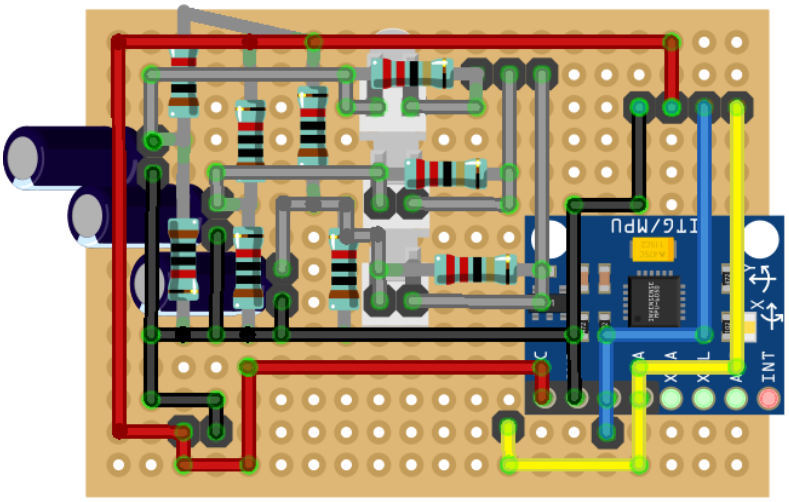
Figura 15 - Circuito DAQ-IOT



Fonte: Autor (2023)

Os protótipos iniciais foram realizados via protoboard, todavia faz-se necessário a elaboração de um circuito compacto, seguro, eficiente e robusto. Para isso, elaborou-se o *layout* do circuito impresso, facilitando implementação do mesmo ao definir as trilhas e lugares dos componentes, conforme ilustra a Figura 16.

Figura 16 - Layout PCB do DAQ-IOT

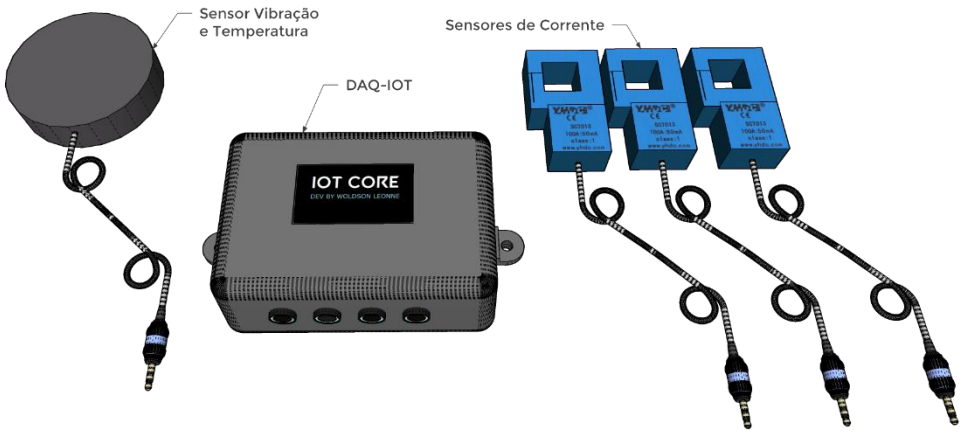


Fonte: Autor (2023)

Como uma solução final, ao levar em consideração o fator manutenção, optou-se por realizar uma montagem modularizada, na qual o microcontrolador e sensores se encaixem na placa que contém os demais componentes eletrônicos.

Essa conexão de encaixes ocorre através de conectores *jst*, e pinos macho e fêmea de *11mm*. Enquanto a conexão dos sensores no case ocorre através de conexões *P2*. Desse modo, caso o microcontrolador apresente falha, a nível de manutenção basta desencaixar o microcontrolador da placa eletrônica e encaixar um novo. Caso um dos sensores apresente defeito, a troca ocorre de maneira similar, retirando o sensor com o *plug P2* macho, do case com o conector *P2* fêmea. Desse modo, não há a necessidade de trocar o DAQ-IOT completo, o que é muito benéfico do ponto de vista de custos. A Figura 17 ilustra o DAQ-IOT e componentes periféricos, que são os sensores.

Figura 17 - DAQ-IOT modelado em 3D



Fonte: Autor (2023)

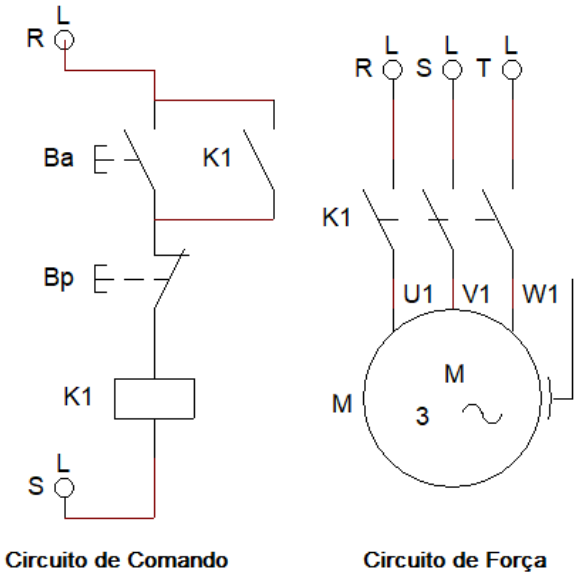
3.4 Circuito de Comando Auxiliar

O circuito auxiliar tem a finalidade de controlar a partida e parada do motor, utilizando botões (ou botoeiras) e contactoras. A partida direta é um método de partida de motores elétricos em que a tensão nominal da rede é aplicada diretamente aos terminais do motor, sem a utilização de dispositivos de partida, como *soft-starters* ou inversores de frequência. Esse método é adequado para motores de pequeno porte e

baixa potência. Optou-se por elaborar esse circuito para garantir maior flexibilidade e segurança nas simulações e testes.

O circuito auxiliar de partida direta é composto por uma botoeira de comando, que permitem ao operador acionar o motor a partir de uma posição remota. As botoeiras são ligadas a contactoras, que são dispositivos de manobra que permitem controlar a corrente elétrica que passa pelo motor. As contactoras têm a função de estabelecer e interromper o circuito elétrico do motor. A Figura 18 ilustra o circuito elétrico para comando auxiliar de uma carga.

Figura 18 - Circuito de comando auxiliar para partida direta



Fonte: Autor (2023)

No circuito auxiliar de partida direta, a contactora principal é acionada pela botoeira de acionamento (Ba), que envia um sinal elétrico para a bobina da contactora (K1), fechando assim o circuito elétrico do motor. A contactora principal também pode ser desligada por uma botoeira de parada (Bp), que envia um sinal elétrico para a bobina da contactora, abrindo o circuito elétrico do motor.

3.5 Raspberry Pi e Broker MQTT

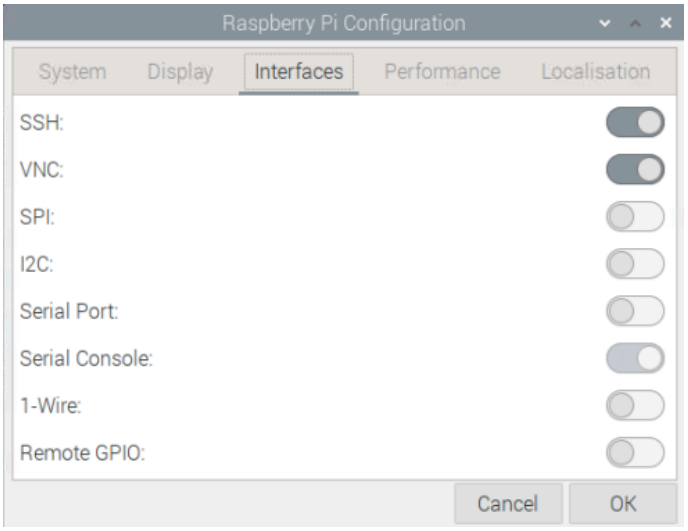
A instalação do broker MQTT no Raspberry Pi (RPi) pode ser realizada em diferentes sistemas operacionais (SO) suportados pela arquitetura do Raspberry Pi,

mas para o projeto em questão utilizou-se o Raspberry Pi OS, conhecido antigamente como Raspbian. Fez-se necessário um cartão de memória de 16GB classe 10 para a instalação do SO, além de um computador alternativo para realizar o download do assistente de instalação disponível do site oficial do Raspberry chamado de *Raspberry Pi Imager*, os detalhes da instalação encontram-se disponíveis no Apêndice A.

No RPi, deve-se realizar algumas configurações importantes antes de instalar o broker MQTT. A priori, deve-se instalar e habilitar o VNC (*Virtual Network Computing*), que é um sistema de compartilhamento gráfico de desktop que usa o *Remote Frame Buffer Protocol* para remotamente controlar outro computador. Dessa forma, pode-se realizar as configurações de forma remota no Raspberry Pi a partir de um computador remoto na mesma rede, facilitando o uso de periféricos como monitor, mouse e teclado.

Desse modo, realizou-se o download e instalação do VNC server no Raspberry Pi, e o download e instalação do VNC Viewer em um computador para acesso remoto. Após a instalação, pode-se conectar ao RPi através do VNC Viewer, todavia necessita de uma configuração adicional, deve-se permitir que o RPi aceite conexões via VNC. Para isso, deve-se abrir as configurações do RPi, e habilitar a opção do VNC, conforme Figura 19.

Figura 19 - Habilitação VNC no Raspberry Pi



Fonte: Autor (2023)

Com a interface de VNC habilitada, deve-se configurar uma nova conexão remota através do VNC Viewer, necessitando somente inserir o endereço IP (*Internet Protocol*) do RPi, e sem seguida inserir as credenciais de usuário do RPi.

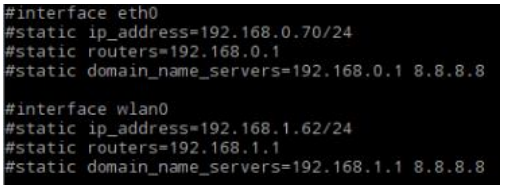
O próximo passo para a configuração ideal, é definir o RPi com IP estático, pois o endereço IP será um parâmetro de identificação do broker MQTT, e esse parâmetro será referência no código do DAQ-IOT, e a variação do mesmo representa um erro de conectividade entre o broker e DAQ-IOT. Desse modo, para configurar o RPi com IP fixo, deve-se abrir o terminal e digitar o comando "*sudo nano /etc/dhcpcd.conf*", conforme a Figura 20.

Figura 20 - Comando para configurar o Raspberry Pi com IP fixo

Fonte: Autor (2023)

Em seguida, o arquivo "*dhcpcd.conf*" é aberto no próprio terminal, e então deve-se localizar o trecho da Figura 21 e remover o "#" da frente de cada linha, inserindo o endereço IP fixo desejado tanto para conexão wireless como cabeada.

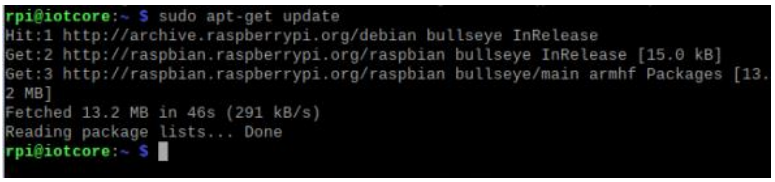
Figura 21 - Edição do arquivo dhcpcd.conf



Fonte: Autor (2023)

O broker MQTT escolhido foi o Mosquitto, para proceder a instalação, deve-se primeiramente atualizar a lista de pacotes, com o comando "*sudo apt-get update*" via terminal, conforme a Figura 22.

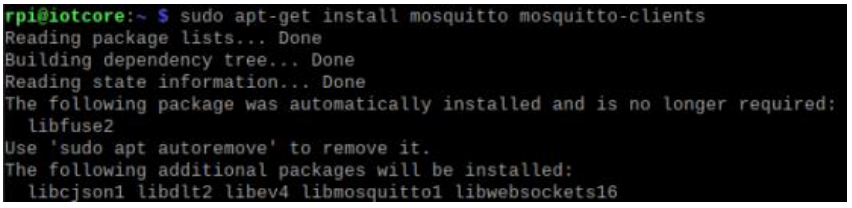
Figura 22 - Comando de atualização da lista de pacotes



Fonte: Autor (2023)

Com a lista de pacotes atualizadas, pode-se então instalar o *mosquitto broker* e *mosquitto client* com o comando “*sudo apt-get install mosquitto mosquitto-clients*”, conforme a Figura 23.

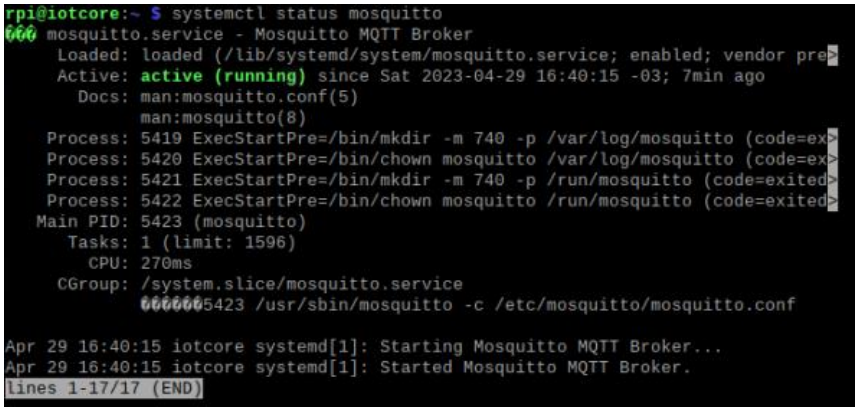
Figura 23 - Comando para instalar o Mosquitto



Fonte: Autor (2023)

Após a instalação, o serviço Mosquitto deve começar automaticamente, pode-se verificar o status do serviço utilizando o comando “*systemctl status mosquitto*”, conforme Figura 24.

Figura 24 - Comando para verificar status do serviço mosquitto



Fonte: Autor (2023)

Desse modo, a configuração básica do *broker* é definida. Vale ressaltar que é importante configurar também a senha, para prover maior segurança ao *broker*. Todavia, para fins de desenvolvimento e testes, não configurou-se um usuário e senha, ou seja, o *broker* está aberto na rede.

Em aplicação de desenvolvimento, não configurar uma senha não representa grandes riscos, mas em aplicação em produção, deve-se realizar a configuração de usuário e senha para acessar o *broker* e garantir maior segurança aos dados.

3.6 Servidor Linux

O servidor abordado na arquitetura do sistema é um elemento fundamental e um dos mais importantes no sistema. O servidor em Linux foi então virtualizado e configurado para receber as aplicações do sistema. Nesse sentido, o VMware é uma solução líder de mercado para virtualização de servidores, que permite a criação de máquinas virtuais (VMs) em um ambiente virtualizado.

Com a instalação de diversas aplicações, como banco de dados e aplicações de visualização de dados, é possível obter uma infraestrutura robusta e escalável para suportar as demandas de uma empresa.

Existem várias distribuições do Linux, e para o servidor optou-se por utilizar o Ubuntu 20.04.5 LTS 64bits, com 8GB de memória RAM, processador Intel® Core™ i7-8565U CPU 1.80GHz × 2, capacidade de armazenamento de 60GB, e GNOME versão 3.36.8.

O processo de instalação do SO não será abordado em detalhes, mas resumidamente deve-se realizar o download do imagem do SO do site oficial da distribuição Linux, e no VMWare criar uma nova máquina virtual. Deve-se então selecionar a imagem baixada e proceder com a instalação do Linux. Importante ressaltar que ao criar uma máquina virtual, pode-se definir os parâmetros da máquina, como a quantidade de memória e de armazenamento, dentro dos limites da máquina física real.

Após a instalação, pode-se então realizar algumas configurações importantes e instalações essenciais para que a máquina sirva como servidor. O primeiro passo, similar ao que ocorreu com o Raspberry Pi, é definir a máquina com um IP fixo, pois nela serão feitas configurações de banco de dados, e o endereço IP é um parâmetro levado em consideração para definir configurações importantes de armazenamento.

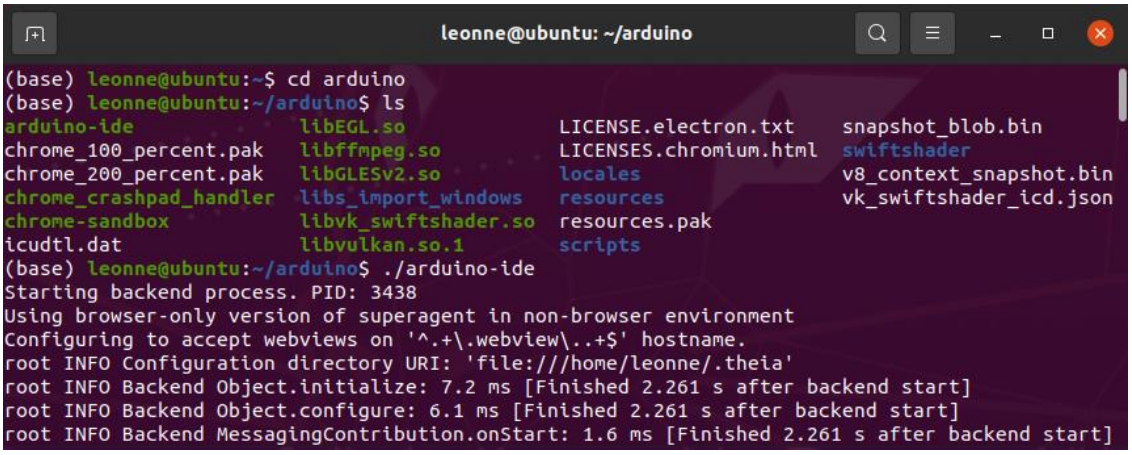
Para definir como IP fixo, deve-se abrir as configurações de rede, selecionar *IPv4*, modificar a seleção de automático para manual, e inserir o *IP* fixo que se deseja. Para a aplicação escolheu-se o *IP 192.168.1.29*. Como existem poucos dispositivos na rede, não corre tanto risco de um outro dispositivo assumir esse mesmo endereço *IP*. Além disso, um servidor basicamente sempre permanece conectado, o que garante mais ainda boa confiabilidade no endereçamento manual fixo. Apesar disso, é recomendado utilizar reserva de IPs na configuração do servidor DHCP.

3.6.1 IDE Arduino

Para programar o ESP32, faz-se necessário um ambiente de desenvolvimento integrado, para isso, optou-se pelo Arduino, um ambiente com uma grande comunidade e suporte. A priori, realizou-se o download do instalador no site oficial do Arduino e realizou-se a instalação.

Após a instalação, faz-se necessário algumas configurações importantes. Deve-se abrir o software instalado, e para isso requer conhecimentos de Linux. Diferentemente do Windows, para abrir o software deve-se entrar no diretório raiz que foi instalado, via terminal, e colocar o comando *"./arduino-ide*" conforme a Figura 25.

Figura 25 - Acesso a IDE Arduino

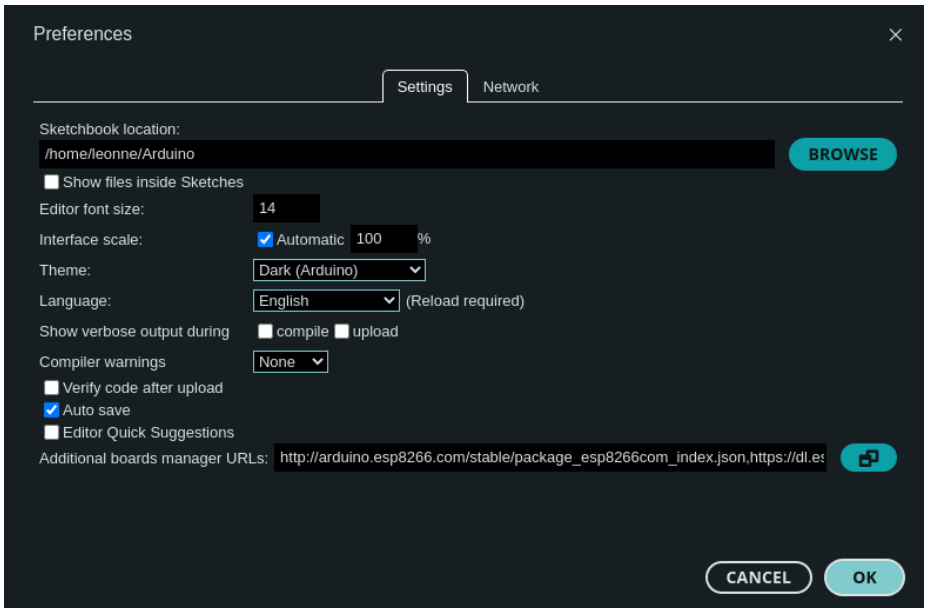


Fonte: Autor (2023)

Com isso, o software Arduino é carregado. Vale ressaltar que não se deve fechar o terminal em questão, pois o software também será fechado juntamente. Caso seja necessário ouros comandos via terminal, deve-se abrir um novo terminal.

Através da IDE do Arduino, pode-se programar em várias placas, como Arduino UNO, Arduino Mega, entre outras. O microcontrolador a ser utilizado é o ESP32, e nesse sentindo, é necessário instalar os recursos dessa placa dentro do ambiente de desenvolvimento. Para tanto, deve-se incluir um arquivo em json (*JavaScript Object Notation*), em "*preferences*" disponível no menu "*file*". O arquivo json está disponível através da URL "*https://dl.espressif.com/dl/package_esp32_index.json*", conforme a Figura 26.

Figura 26 - Inclusão da placa ESP-32 na IDE Arduino



Fonte: Autor (2023)

O outro endereço em exibido trata-se da inclusão do ESP8266, e ilustra bem como adicionar outras placas, basta apenas separar as URL por vírgula. Ao adicionar a URL, as opções de placas aparecerão disponíveis "*board*", no menu "*tools*".

Com o ambiente de desenvolvimento instalado, pode-se então programar o ESP32, para isso necessita-se conectar a placa ao computador, e conectar a máquina virtual posteriormente. Em seguida, deve-se desbloquear a porta USB na qual está conectado o microcontrolador. Para isso, deve-se inserir o comando "*sudo chmod a+rw /dev/ttyUSB0*" via terminal, caso seja o endereço *USB0*. Para outros endereços basta trocar o último dígito, modificar de zero para um, dois, ou seja, qual for o endereço que esteja a placa.

3.6.2 InfluxDB

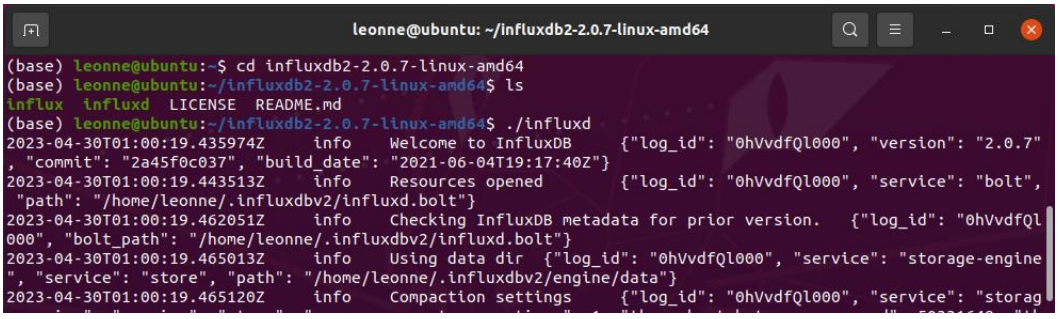
Para instalação e configuração do banco de dados de séries temporais InfluxDB, deve-se realizar o download do instalador. Nessa configuração realizou-se o download do arquivo em binário, através do comando "*wget https://dl.influxdata.com/influxdb/releases/influxdb2-2.7.1-linux-amd64.tar.gz*" via terminal, em seguida, descompactou-se o arquivo através do comando "*tar xvfz influxdb2-2.7.1-linux-amd64.tar.gz*", conforme ilustrado na Figura 27.

Figura 27 - Download do influxDB

Fonte: Autor (2023)

Para ativar o serviço, deve-se entrar na pasta descompactada com o InfluxDB, e abrir o arquivo "*influxd*", conforme ilustrado na Figura 28. Vale ressaltar que não deve-se fechar o terminal em questão, pois o serviço ficará indisponível.

Figura 28 - Habilitação do serviço do InfluxDB

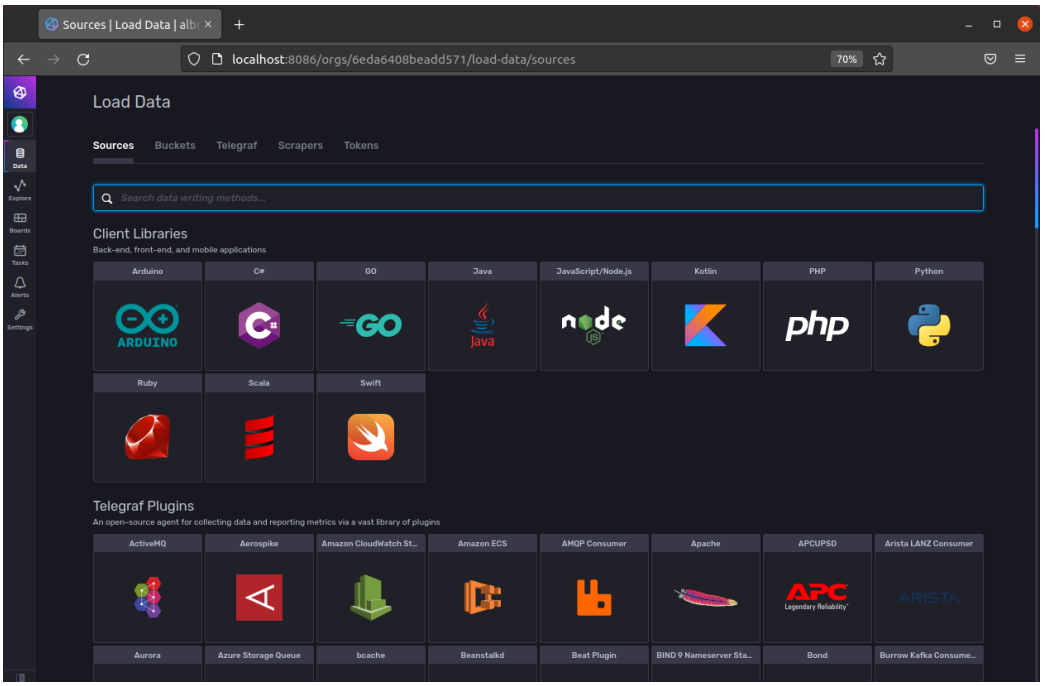


Fonte: Autor (2023)

Ao realizar o comando para iniciar o serviço, automaticamente a interface do banco de dados fica disponível na porta *8086*, por default. Ou seja, pode-se acessar o serviço ao abrir o navegador, inserir o endereço IP da máquina que está servindo a aplicação, e a porta *8086*.

Para primeiro acesso, a senha e usuário padrão é "*admin*", e em seguida a aplicação solicita configurar uma senha e usuário como administrador. Desse modo, após realizar essa configuração, pode-se utilizar a aplicação como banco de dados de séries temporais, que possui uma interface atraente e com várias possibilidades de fontes de dados diferentes, conforme ilustrado na Figura 29.

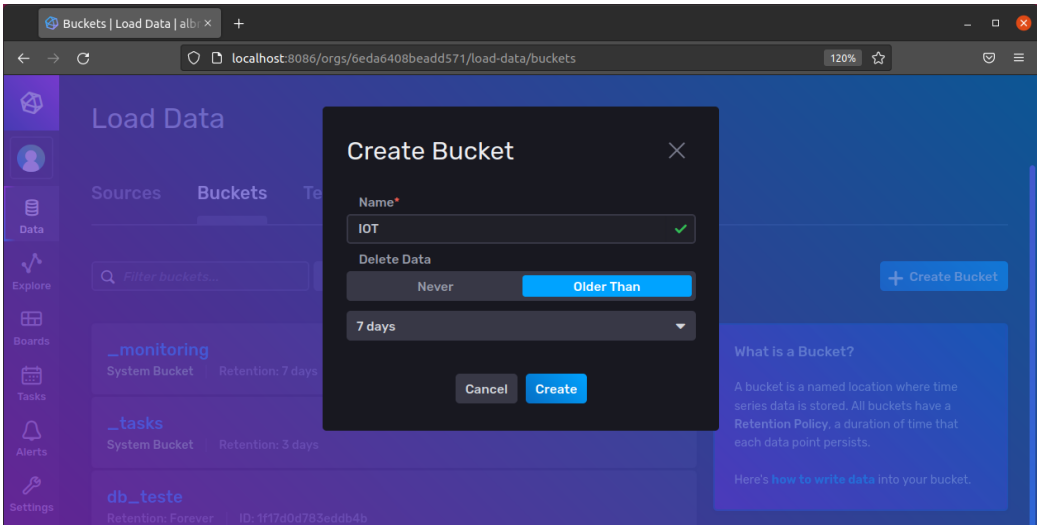
Figura 29 - Tela de conexão de fonte de dados do InfluxDB



Fonte: Autor (2023)

Para armazenar os dados, deve-se criar um banco. Para isso, no menu "*Buckets*" deve-se clicar em "*Create Bucket*", uma nova janela aparecerá solicitando o nome do banco e opção de retenção dos dados desse banco, que define o máximo de tempo que os dados ficarão retidos. Configurou-se com retenção de 7 dias, ou seja, pode-se recuperar os dados do banco até 7 dias atrás, após isso, os dados são deletados automaticamente. Essa configuração é observada na Figura 30.

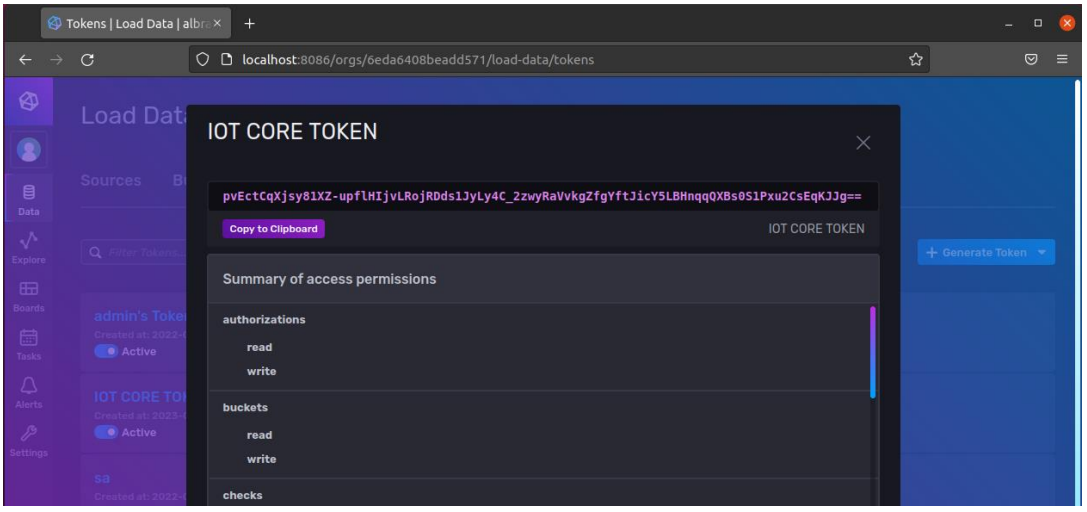
Figura 30 - Criação de Bucket no InfluxDB



Fonte: Autor (2023)

Diferentemente de bancos de dados como SQL Server, que a autenticação ocorre através *login* e senha, no InfluxDB esse acesso pode ocorrer através de uma configuração de *token* de acesso. Várias *tokens* podem ser criadas e com vários níveis de acesso, e a diferentes *Buckets*, a Figura 31 ilustra a criação de um *Token* para acesso a todos os *Buckets* e com todas as permissões concedidas.

Figura 31 - Token de acesso ao InfluxDB



Fonte: Autor (2023)

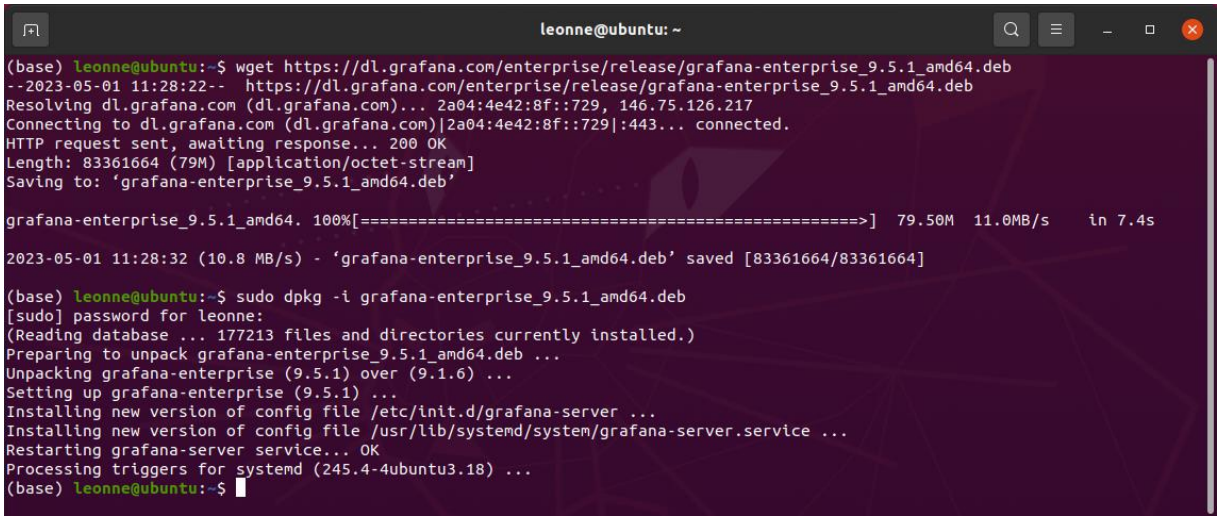
Esse *token* será utilizado em dois momentos, conforme a arquitetura do projeto. O primeiro momento será a utilização pelo *script* em Python chamado "*BRIDGE*", o qual, através da biblioteca do InfluxDB, armazenará os dados coletados. E, em um segundo momento esse *token* será utilizado para configurar o InfluxDB como uma fonte de dados para o Grafana, para a elaboração de painéis e visualização dos dados em tempo real.

3.6.3 Grafana

O Grafana é uma aplicação essencial pra a visualização dos dados coletados pelo DAQ-IOT, desse modo, a priori faz-se necessário instalar o mesmo. A página web de *download* do Grafana possui várias orientações de instalações, bem como uma documentação completa, com vídeos tutoriais para os primeiros passos com o Grafana. Para instalar o Grafana, utilizou-se dois comandos, um para baixar o

instalador, e outro para descompactar e instalar o Grafana, os comandos foram "*wget https://dl.grafana.com/enterprise/release/grafana-enterprise_9.5.1_amd64.deb*" e o comando "*sudo dpkg -i grafana-enterprise_9.5.1_amd64.deb*", como ilustra na Figura 32.

Figura 32 - Instalação do Grafana



Fonte: Autor (2023)

Após a instalação, o serviço do Grafana inicia automaticamente, mas caso isso não ocorra, pode-se iniciar o serviço com o comando "*sudo systemctl start grafana-server*", e verificar o status do serviço com o comando "*sudo systemctl status grafana-server*", conforme ilustrado na Figura 33.

Figura 33 - Iniciar serviço Grafana e verificar status

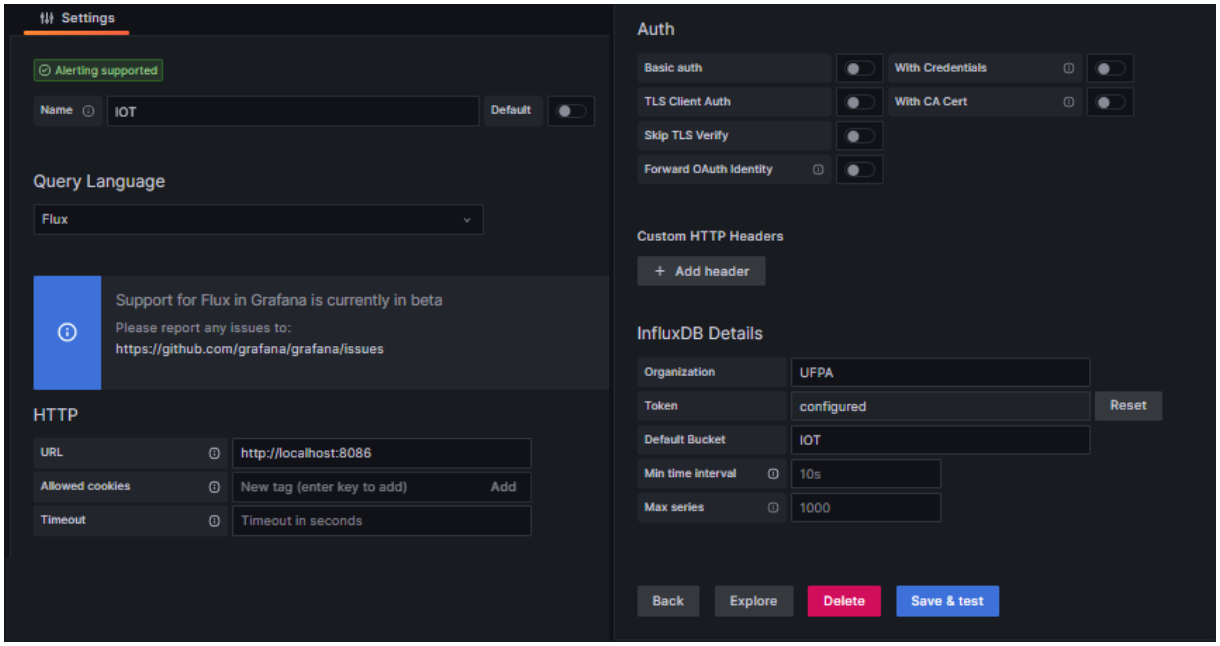
Fonte: Autor (2023)

Conforme observado na Figura 33, nota-se que o serviço está ativo, desse modo, pode-se acessar o Grafana através de um navegador web, de forma similar de acesso ao InfluxDB. O painel do Grafana pode ser acessado digitando o endereço *IP*

do servidor seguido da porta *3000*. Por exemplo: “*http://<IP_DO_SERVIDOR>:3000*”. Na primeira vez de acesso ao painel, faz-se necessário criar um usuário e senha para acessar o Grafana.

Com o Grafana instalado e com o serviço ativo, faz-se necessário configurar a fonte de dados do InfluxDB no Grafana. Desse modo, na tela inicial do Grafana, deve-se clicar no ícone de engrenagem no menu lateral e selecionar "*Data Sources*". Em seguida, clicar no botão "*Add data source*" e selecionar o tipo de fonte de dados "*InfluxDB*". Os campos de configuração aparecerão, e então deve-se preencher as informações de conexão do InfluxDB, como o endereço *IP* do servidor e as credenciais de acesso. Em seguida, deve-se clicar no botão "*Save & Test*" para salvar as configurações e testar a conexão com o InfluxDB. O banco de dados criado no InfluxDB chama-se IOT, desse modo, a Figura 34 ilustra essa configuração.

Figura 34 - Configuração de fonte de dados InfluxDB no Grafana



Fonte: Autor (2023)

Desse modo, através de linguagem flux, realizou-se a criação das dashboards para visualização dos dados em tempo real. Outro aspecto importante trata-se da possibilidade de múltiplas fontes de dados, inclusive SQL Server.

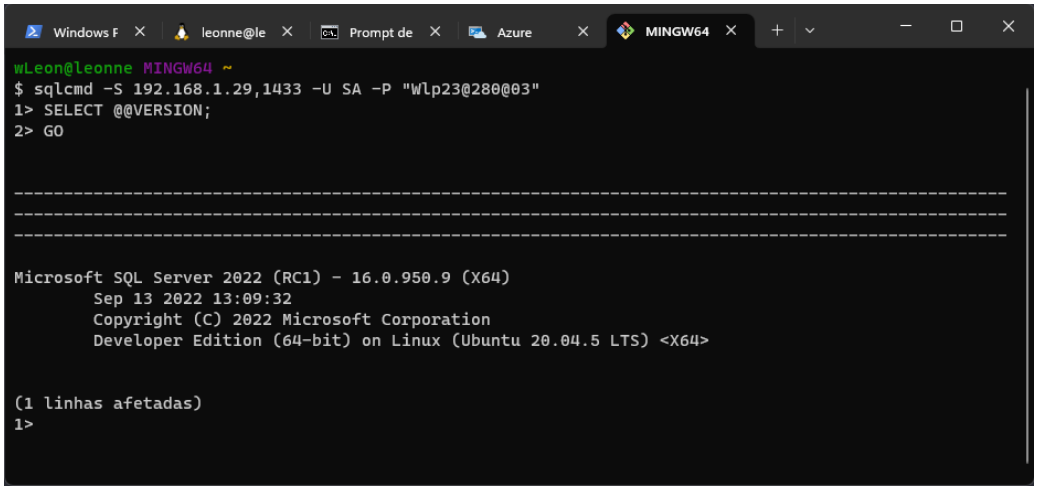
3.6.4 Docker

A instalação do Docker no servidor Linux, com detalhes de versão utilizada, é descrita no Apêndice B. O objetivo da instalação do Docker deve-se apenas para a instalação do SQL Server, que é nativamente do sistema operacional Windows. Todavia, pode-se utilizar todas as aplicações como imagens no Docker, como o InfluxDB e Grafana, por exemplo.

Para verificar se a instalação obteve sucesso, pode-se verificar através de uma autenticação via gerenciador de banco de dados, como o SQL Management Studio ou Azure Data Studio, ou até mesmo tentar *logar* através do terminal do Linux.

Todavia, realizou-se essa verificação utilizando o Git Bash, em um outro computador na mesma rede, com sistema operacional Windows, para verificar também se o acesso na rede está configurado corretamente e permitido. Pois o acesso pode ser configurado somente no *host* local da máquina servidor, e para a aplicação faz-se necessário que o acesso esteja disponível em rede. A Figura 35, ilustra o resultado dessa verificação de conexão, bem como exibição da versão utilizada.

Figura 35 - Teste de conectividade no banco SQL Server e verificação da versão



Fonte: Autor (2023)

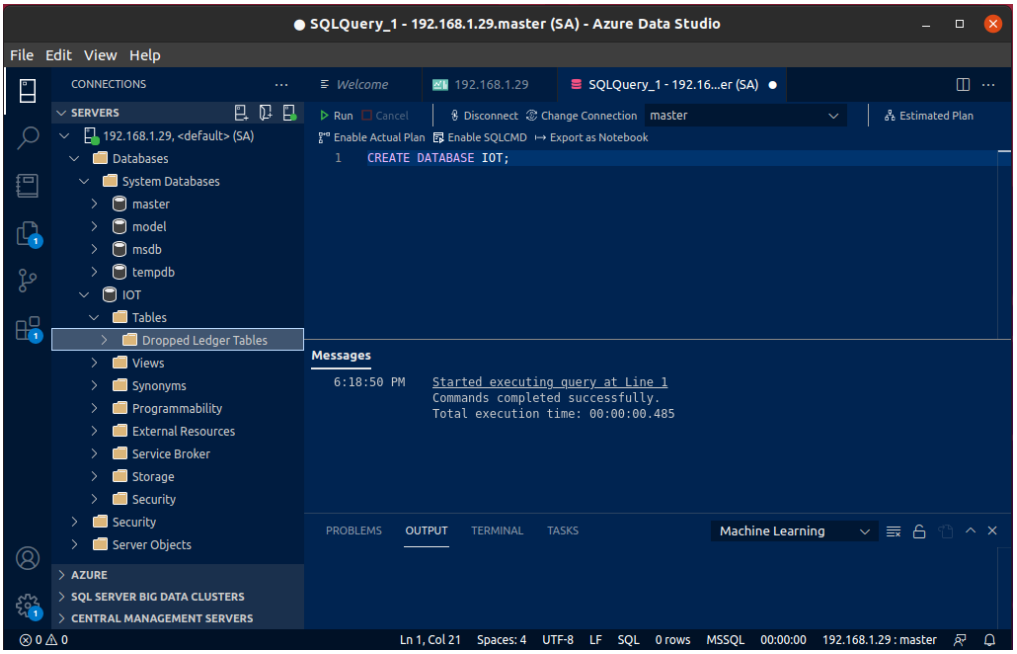
Para a verificação o Git Bash é opcional, essa verificação pode ser realizada também pelo Windows PowerShell e Prompt de Comando. Desse modo, o servidor foi configurado para armazenar dados em um banco SQL Server através do Docker.

3.6.5 Azure Data Studio

Para realizar a instalação do Azure Data Studio, deve-se primeiramente realizar o download do instalador com extensão "*.deb*" do site oficial da Microsoft, em seguida, deve-se realizar a instalação através do arquivo. Após a instalação, pode-se facilmente acessar o banco de dados SQL Server anteriormente criado. Para acessar o banco de dados, deve-se escolher a opção "*Create a Connection*", uma janela solicitando os dados para autenticação aparecerá. Após a autenticação, os dados do banco estarão disponíveis para gerenciamento.

Faz-se necessário criar um banco e tabela específico para a aplicação. Para isso, deve-se executar a query "*CREATE DATABASE IOT;*" para criar um banco chamado "*IOT*". Após a operação, o banco deve aparecer listado na árvore de navegação em "*Databases*", conforme ilustra a Figura 36.

Figura 36 - Criação do banco de dados IOT



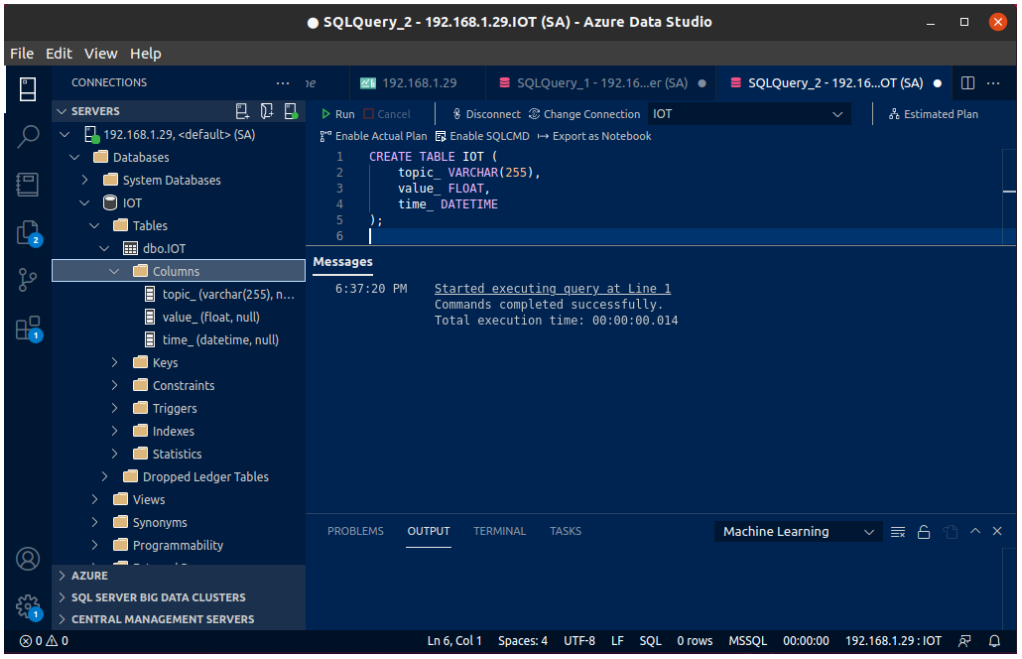
Fonte: Autor (2023)

Para armazenar os dados, faz-se necessário a criação de uma tabela dentro do banco IOT. Para isso, criou-se uma tabela chamada IOT com as colunas "*topic_*", "*value_*" e "*time_*". A coluna "*topic_*" receberá o tópico MQTT, o qual é uma cadeia de caracteres, ou seja, *VARCHAR*. A coluna "*value_*" receberá o valor da variável do

tópico, e deve ser do tipo *FLOAT*, pois é um decimal. A coluna "*time_*" receberá o valor de data e hora que a variável foi lida, e deve ser do tipo "*DATETIME*". O nome da coluna e tipo dos dados da coluna são informações importantes para a criação da tabela, e ao passar dados com tipos diferentes para essa tabela ocasiona em erros inesperados.

Para a criação dessa tabela, utilizou-se a query SQL "*CREATE TABLE IOT (topic_ VARCHAR(255), value_ FLOAT, time_ DATETIME);*". A Figura 37 ilustra a query, o resultado de execução e tabela criada, com as colunas especificadas, dentro da árvore do banco IOT. Importante ressaltar que a query deve ser executada dentro do banco IOT e não em outro banco, como o banco do sistema.

Figura 37 - Criação da tabela IOT



Fonte: Autor (2023)

Desse modo, o banco de dados foi configurado conforme a necessidade para armazenar os dados do protocolo MQTT. A inserção dos dados ocorrerá via *script* em Python chamado "*BRIDGE.py*", o qual realiza a leitura dos dados via assinante MQTT e realiza a inserção dos dados nesse banco criado. Desse modo, posteriormente, outras aplicações podem visualizar esses dados armazenados, via Grafana, aplicação em Flask, Power BI ou Genesis64.

3.6.6 IDE Anaconda

A Anaconda é uma das principais IDEs para desenvolvimento de projetos de ciência de dados, análise de dados e machine learning. A principal característica da Anaconda é a inclusão de uma grande variedade de pacotes e bibliotecas de análise de dados em Python, o que permite aos usuários realizar tarefas como visualização de dados, modelagem estatística e análise de *big data* de forma rápida e eficiente.

A Anaconda é composta de duas partes principais: o Anaconda Navigator, uma interface gráfica que permite aos usuários gerenciar ambientes virtuais, instalar novos pacotes e acessar aplicativos de terceiros, e o Anaconda Prompt, uma interface de linha de comando que permite aos usuários executar tarefas específicas, como criar um ambiente virtual ou instalar pacotes específicos. Alguns passos para instalação, uso e telas do Anaconda encontram-se descritos no Apêndice D.

O software anaconda é essencial para gerenciar os pacotes de bibliotecas necessárias para o desenvolvimento da aplicação em Flask, bem como demais scripts como o "*BRIDGE.py*". Além disso, o ambiente para a escrita dos códigos é realizado através do software de terceiros chamado "*Spyder*". Esse software é uma IDE avançada de edição e testes de scripts em Python. A Figura 45 ilustra o ambiente de desenvolvimento integrado Spyder.

3.6.7 Aplicações opcionais

Além do Anaconda e os demais softwares instalados, outras aplicações foram instaladas, mas não são essenciais para o funcionamento do sistema, tornando-se a instalação opcional, e por esse motivo não serão abordados em detalhes. Os softwares opcionais utilizados foram o Telegraf, Mosquitto, HiveMQ, MQTT Explorer e UAExpert.

Utilizou-se o Telegraf como uma ferramenta para monitorar o servidor, coletando dados de memória RAM utilizada, CPU, processos com maior consumo de memória, e armazenamento disponível no SSD (*Solid State Drive*). Como o servidor possui uma função vital para o sistema, qualquer erro relacionado ao mesmo, como falta de memória, representam um risco para a operação. O Telegraf realiza o envio

desses dados para o InfluxDB, como maneira de armazenar o comportamento histórico do servidor.

O Mosquitto e HiveMQ são brokers MQTT, utilizados para testes opcionais de broker. Apesar de utilizar o Mosquitto no Raspberry Pi para servir como broker, pode-se utilizar o broker MQTT Mosquitto no próprio servidor, e a instalação do mesmo serviu apenas como testes de funcionalidade, mas não fazem parte da arquitetura final. Esse teste surgiu em decorrência de dúvidas acerca de até que ponto a utilização do Raspberry Pi em sistemas de automação torna-se confiável. Por outro lado, a utilização do HiveMQ surgiu da necessidade de testes de um *broker* alternativo, além de testes de soluções de broker em nuvem.

O MQTT Explorer e UAExpert foram utilizados apenas para visualizar de maneira rápida e fácil os dados transmitidos através do protocolo de comunicação MQTT e OPC UA, respectivamente. A utilização dos mesmos é simples e não necessita de configurações complexas, necessita-se apenas autenticar com o endereço *IP* e portas utilizadas pelo *broker* MQTT e servidor OPC UA.

3.7 Sketch do DAQ-IOT

O *sketch* do DAQ-IOT, ou código, foi escrito em linguagem de programação C++. Esse código é responsável pelo controle do microcontrolador, e define como é realizada a aquisição dos sinais dos sensores, modelagem de dados, e envio desses dados via protocolo de comunicação MQTT.

A priori, realizou-se a importação de algumas bibliotecas importantes, as bibliotecas utilizadas são: *Arduino*, *EmonLib*, *WiFi*, *driver/adc*, *Adafruit_MPU6050*, *Adafruit_Sensor*, *Wire*, *PubSubClient* e *SimpleKalmanFilter*. Em resumo, essas bibliotecas são necessárias para:

- "*Arduino*" fornece funções e definições básicas para estruturar o código, a biblioteca "*EmonLib*" é uma biblioteca para medir o consumo de energia elétrica, e fornece funções para interagir com sensores de corrente, como o SCT-013.
- "*WiFi*" permite ao ESP-32 se conectar a uma rede Wi-Fi e realizar operações relacionadas à rede.

- “*driver/adc*” é usada para controlar o conversor analógico-digital (ADC) do microcontrolador, ela fornece funções para configurar e ler valores do ADC.
- “*Adafruit_MPU6050*” e “*Adafruit_Sensor*” fornecem funções para ler os dados do sensor MPU6050 e realizar cálculos com eles.
- “*Wire*” é uma biblioteca de comunicação I2C, ela fornece funções para enviar e receber dados por meio do protocolo I2C.
- “*PubSubClient*” é uma biblioteca para criar clientes MQTT no microcontrolador, ela permite que o microcontrolador publique e assine mensagens em um *broker* MQTT.
- “*SimpleKalmanFilter*” é uma biblioteca que implementa um filtro de Kalman simples, ela é usada para estimar o valor verdadeiro de uma variável com base em medições imprecisas.

Após a importação das bibliotecas, define-se algumas variáveis relacionadas ao horímetro, bem como o estado inicial das mesmas, as variáveis são “*tempo_referencia*” e “*tempo_total_ligado*”, ambas iguais à zero.

Em seguida, cria-se uma instância de um objeto “*SimpleKalmanFilter*” chamado “*simpleKalmanFilter*” com determinados parâmetros. O construtor da classe “*SimpleKalmanFilter*” é chamado com três argumentos. Esses argumentos definem os parâmetros iniciais do filtro de Kalman.

O primeiro argumento, representa a incerteza inicial na estimativa da variável que está sendo filtrada. Quanto maior o valor, maior a incerteza inicial. O segundo argumento representa a incerteza inicial na estimativa da taxa de mudança da variável, assim como o primeiro argumento, quanto maior o valor, maior a incerteza inicial. O terceiro argumento, representa o ruído do processo, que é uma medida da variabilidade do sistema que está sendo filtrada, esse valor afeta como o filtro de Kalman se ajusta às mudanças nos dados de entrada.

Na sequência, cria-se uma instância de um objeto “*Adafruit_MPU6050*” chamado “*mpu*”. Essa instância do objeto é usada para interagir com o sensor MPU6050. Com a instância “*mpu*” criada, pode-se chamar métodos dessa classe para acessar as funcionalidades do MPU6050.

Em seguida, define-se os pinos GPIO onde os sensores de corrente SCT-013 estão conectados. Os valores *32*, *34* e *35* são os números dos pinos do *GPIO* na placa do ESP-32, e cada entrada representa a leitura de uma fase, sendo *R*, *S* e *T*.

Posteriormente, define-se alguns parâmetros para leitura de corrente. O parâmetro "*HOME_VOLTAGE*" define a tensão nominal da carga na qual fará a aquisição de corrente. E, "*ADC_BITS*" e "*ADC_COUNTS*", são parâmetros que definem o número de *bits* e o número de contagens do conversor analógico-digital (ADC). O ADC foi configurado com uma resolução de 10 *bits*.

Logo após, cria-se três instâncias da classe "*EnergyMonitor*" chamadas "*emon1*", "*emon2*" e "*emon3*", essa classe faz parte da biblioteca *EmonLib* e serão usadas posteriormente para configurar e ler os sensores de corrente conectados aos pinos *GPIO* especificados.

Na sequência, define-se as credenciais de Wi-Fi, para que o ESP-32 possa se conectar à rede, e define-se também a configuração do cliente MQTT, em "*mqtt_server*" define-se o endereço *IP* do *broker* MQTT, ou seja, o endereço *IP* do Raspberry Pi. Em seguida, definiu-se uma instância do cliente Wi-Fi "*espClient*" e uma instância do cliente MQTT "*cliente*". A instância do cliente MQTT é inicializada com a instância do cliente Wi-Fi. Essas instâncias são usadas para estabelecer uma conexão Wi-Fi e uma conexão MQTT com o servidor especificado.

A função "*connectToWiFi()*" configura o modo do Wi-Fi para estação (*WIFI_STA*), define o nome do host como "*esp32-pilot*" e inicia a conexão com a rede Wi-Fi usando o SSID e a senha fornecidos. Em seguida, ela verifica o status da conexão Wi-Fi e tenta se conectar até 15 vezes. Se a conexão não for estabelecida, o dispositivo entra em modo de sono profundo (*esp_deep_sleep_start()*) por um minuto antes de tentar novamente.

A função "*reconnect()*" é responsável por estabelecer uma conexão com o servidor MQTT. Ela entra em um *loop* até que o cliente MQTT esteja conectado. Dentro do *loop*, um *ID* de cliente aleatório é gerado e usado para a conexão MQTT. Se a conexão for estabelecida com sucesso, é possível publicar ou se inscrever em tópicos MQTT. Caso contrário, é aguardado um atraso de cinco segundos antes de tentar novamente a conexão.

A função "*calc_imbalance()*" calcula o desequilíbrio de corrente em um sistema trifásico com base nas correntes medidas em cada fase. A função recebe três

parâmetros: “*ampsr*”, “*ampss*” e “*ampst*”, que representam as correntes medidas em cada uma das fases *R*, *S* e *T*, respectivamente. As variáveis “*rs*”, “*st*” e “*rt*” são calculadas como as diferenças absolutas entre as correntes das fases correspondentes. Os valores de “*rs*”, “*st*” e “*rt*” são armazenados em um *array* chamado “*dif*”. A variável “*maximo*” é inicializada com o valor de “*dif[0]*”, ou seja, o primeiro elemento do *array*. Um *loop* é executado para percorrer os elementos restantes do *array* “*dif*”. Cada elemento é comparado com o valor atual de “*máximo*”, e se for maior, atualiza-se o valor de máximo. A média das correntes “*ampsr*”, “*ampss*” e “*ampst*” é calculada e armazenada na variável “*m*”. O desequilíbrio é calculado como a razão entre o valor máximo (*maximo*) e a média (*m*), multiplicado por 100 para obter o resultado em porcentagem.

Em seguida, várias configurações e inicializações são feitas dentro da função “*setup()*”. Configurou-se o ADC (Conversor Analógico-Digital) para o canal “*ADC1_CHANNEL_6*” com uma atenuação de *11 dB* e uma resolução de leitura de 10 *bits*. Iniciou-se a comunicação serial com uma taxa de transmissão de 115200 *bps*, conectou-se à rede Wi-Fi usando a função “*connectToWiFi()*” definida anteriormente e configurou-se o cliente MQTT para se conectar ao servidor MQTT no endereço “*mqtt_server*” na porta *1883*.

Posteriormente, inicializou-se os objetos da classe *EnergyMonitor* (*emon1*, *emon2* e *emon3*) com os canais ADC especificados (*ADC_INPUT_CH1*, *ADC_INPUT_CH2* e *ADC_INPUT_CH3*, respectivamente) e um número de calibração de 291.

Logo após, inicializou-se o sensor MPU6050. Se a inicialização falhar, o programa entra em um loop infinito. E definiu-se o alcance do acelerômetro do MPU6050 para *±8G*, o alcance do giroscópio do MPU6050 para *±500* graus por segundo e a largura de banda do filtro passa-baixa do MPU6050 para *5 Hz.*

A função “*loop()*” tem a função de executar os algoritmos responsáveis em ler os dados que provém dos sensores, bem como realizar demais cálculos e enviar via protocolo de comunicação MQTT, em vários ciclos em um loop infinito.

Na função “*loop()*”, a priori verifica-se se o cliente MQTT está conectado. Se não estiver, a função “*reconnect()*” é chamada para tentar reconectar. Em seguida, a função “*client.loop()*” é chamada para manter a comunicação MQTT ativa.

A variável “*lastMeasurement*” é atualizada com o valor atual do tempo em milissegundos usando a função “*millis()*”. Isso marca o momento da última medição. Em seguida, calcula-se a corrente eficaz (RMS) para cada fase usando as funções “*calcIrms()*” do objeto “*emon1*”, “*emon2*” e “*emon3*”. Cada função recebe um parâmetro que representa a quantidade de amostras por leituras que serão realizadas para calcular a corrente.

Em seguida, calcula-se a potência para cada fase multiplicando a corrente (*amps*) pelo valor da tensão nominal (*HOME_VOLTAGE*). Calcula-se o desequilíbrio (*imbalance*) usando a função “*calc_imbalance()*” e passando as correntes das três fases como argumento.

Posteriormente, atualiza-se a estimativa do estado (*estimated_value*) usando o filtro de Kalman. A função “*updateEstimate()*” é chamada do objeto “*simpleKalmanFilter*” e recebe o desequilíbrio (*imbalance*) como argumento.

Para identificar se o equipamento está ligado ou desligado, utiliza-se de um laço condicional, que verifica se a média das correntes das três fases é maior que 0,37, se a condição for verdadeira, significa que o equipamento está ligado. A variável “*tempo_total_ligado*” é atualizada com a diferença entre o tempo atual em milissegundos (*millis()*) e o valor de referência “*tempo_referencia*”. Isso calcula o tempo total em que o equipamento ficou ligado. A variável “*status*” é definida como "*on*", indicando que o equipamento está ligado. A variável “*sec_*” é incrementada em 1, representando o número de segundos desde que o equipamento foi ligado.

Pressupõe-se que o equipamento fique ligado por um longo período de tempo, então exibir essa informação somente em segundos não torna-se ideal. Para isso, o tempo total em que o equipamento ficou ligado é calculado em dias, horas, minutos e segundos.

A variável “*tempo_total_ligado*” representa o tempo em milissegundos que o equipamento ficou ligado. Ela é dividida por 1000 para converter para segundos. O valor resultante é armazenado na variável “*tempo_ligado*”. Em seguida, são calculados os valores de segundos, minutos e horas a partir de “*tempo_ligado*”. Os minutos são obtidos dividindo “*tempo_ligado*” por 60, e as horas são obtidas dividindo os minutos por 60. Um laço condicional é então definido, para verificar se os valores de “*sec_*”, “*min_*” e “*hrs_*” atingiram os limites de contagem, se “*sec_*” for maior que

59, significa que passou um minuto completo, então incrementa-se “*min_*” em 1 e reseta-se “*sec_*” para *0*. O mesmo princípio se aplica aos minutos e horas.

A *string* “*horímetro*” é construída para representar o tempo de funcionamento do equipamento no formato "dias, horas, minutos, segundos". Os valores de “*dias*”, “*hrs_*”, “*min_*” e “*sec_*” são convertidos em *strings* e concatenados nessa ordem para formar a *string* final.

Em seguida, são obtidos os novos eventos do sensor MPU6050, que incluem leituras dos dados de aceleração, giroscópio e temperatura. As variáveis “*a*”, “*g*” e “*temp*” são do tipo “*sensors_event_t*” e são passadas como parâmetros para o método “*getEvent()*” do objeto “*mpu*”, do tipo “*Adafruit_MPU6050*”. Esse método atualiza as variáveis com os valores mais recentes dos eventos do sensor. Com isso, pode-se acessar os dados dos eventos do sensor utilizando as variáveis “*a*”, “*g*” e “*temp*”. Cada uma dessas variáveis contém informações específicas do evento, como a leitura da aceleração, giroscópio e temperatura.

Após essa etapa, os valores dos sensores e outras variáveis são convertidos em *strings* e publicados no tópico MQTT correspondente. Algumas variáveis são convertidas em *strings* utilizando a função “*dtostrf()*” e, em seguida, publicadas no tópico MQTT correspondente usando o método “*client.publish()*”:

- “*temp.temperature*” é convertido em “*tempString*” e publicado no tópico "*esp32/temperature*".
- “*a.acceleration.x*” é convertido em “*AccXString*” e publicado no tópico "*esp32/accelerationX*".
- “*a.acceleration.y*” é convertido em “*AccYString*” e publicado no tópico "*esp32/accelerationY*".
- “*a.acceleration.z*” é convertido em “*AccZString*” e publicado no tópico "*esp32/accelerationZ*".
- “*g.gyro.x*” é convertido em “*GyroXString*” e publicado no tópico "*esp32/rotationX*".
- “*g.gyro.y*” é convertido em “*GyroYString*” e publicado no tópico "*esp32/rotationY*".
- “*g.gyro.z*” é convertido em “*GyroZString*” e publicado no tópico "*esp32/rotationZ*".

- “*ampsR*” é convertido em “*ampRString*” e publicado no tópico "*esp32/current_R*".
- “*ampsS*” é convertido em “*ampSString*” e publicado no tópico "*esp32/current_S*".
- “*ampsT*” é convertido em “*ampTString*” e publicado no tópico "*esp32/current_T*".
- “*wattR*” é convertido em “*wattRString*” e publicado no tópico "*esp32/watt_R*".
- “*watts*” é convertido em “*wattSString”* e publicado no tópico "*esp32/watt_S*".
- “*watt*” é convertido em “*wattTString*” e publicado no tópico "*esp32/watt_T*".
- “*imbalance*” é convertido em “*imbalanceString*” e publicado no tópico "*esp32/imbalance*".
- “*estimated_value*” é convertido em “*imbalance_estString*” e publicado no tópico "*esp32/imbalance_est_kf*".
- O valor da *string* “*status*” é publicado no tópico "*esp32/status*".
- O valor da *string* “*horímetro*” é publicado no tópico "*esp32/horimetro*".

Desse modo, as variáveis de interesse são enviados para o servidor MQTT por meio do objeto “*cliente*” usando o método “*publish().*

3.8 BRIDGE.py

O script *BRIDGE.py*, como a tradução do inglês sugere, trata-se de uma “ponte” para o fluxo de dados. Esse script tem a função de ler os dados que fluem através do protocolo de comunicação MQTT e armazenar em banco de dados, bem como servir de “*gateway*” para converter o protocolo MQTT para OPC UA.

A priori, faz-se necessária a importação de várias bibliotecas. As bibliotecas foram instaladas via terminal, através do comando “*pip*” o qual é uma ferramenta de linha de comando utilizada para instalar e gerenciar pacotes Python. Ele é utilizado para instalar pacotes de bibliotecas de terceiros, atualizar pacotes existentes e remover pacotes que não são mais necessários. Em resumo, as principais bibliotecas necessárias para o projeto são definidas como:

- *opcua*: é uma biblioteca para a comunicação com servidores OPC-UA (*Open Platform Communications - Unified Architecture*).
- *time*: é uma biblioteca que fornece funções para trabalhar com tempo, como atrasos e medições de tempo.
- *paho.mqtt.client*: é uma biblioteca para a comunicação com servidores MQTT (*Message Queuing Telemetry Transport*).
- *datetime*: é uma biblioteca para trabalhar com datas e horários.
- *influxdb_client*: é uma biblioteca para se conectar e enviar dados para um banco de dados InfluxDB.
- *json*: é uma biblioteca para trabalhar com dados no formato JSON (*JavaScript Object Notation*).
- *influxdb*: é uma biblioteca para se conectar e consultar um banco de dados InfluxDB.
- *subprocess*: é uma biblioteca para executar comandos do sistema operacional.
- *os*: é uma biblioteca para fornecer funções para interagir com o sistema operacional.
- *pandas*: é uma biblioteca para trabalhar com dados em formato de tabela.
- *pyodbc*: é uma biblioteca para se conectar e enviar consultas para bancos de dados usando o ODBC (*Open Database Connectivity*).
- *pymssql*: é uma biblioteca para se conectar e enviar consultas para bancos de dados SQL Server usando o protocolo TDS (*Tabular Data Stream*).

Em seguida, definiu-se as informações de conexão com o banco de dados SQL Server e estabeleceu-se a conexão. As variáveis “*serverdb*”, “*database*”, “*username*” e “*password*” são utilizadas para armazenar o endereço do servidor, o nome do banco de dados, o nome do usuário e a senha necessários para a conexão.

A conexão com o banco de dados é estabelecida utilizando a função “*pymssql.connect()*”, que recebe os parâmetros “*server*”, “*user*”, “*password*” e “*database*” para estabelecer a conexão. O retorno da função é armazenado na variável “*conn*”.

Em seguida, é criado um *cursor* para a conexão com o banco de dados utilizando o método "*cursor()*" da variável "*conn*", que é armazenado na variável "*cursor*". O *cursor* é uma interface para executar consultas e comandos SQL no banco de dados.

Após as configurações com o banco SQL Server, cria-se um servidor OPC UA e define-se variáveis para monitoramento de parâmetros em tempo real. A biblioteca "*opcua*" é utilizada para a criação do servidor OPC UA. É criada uma instância da classe "*Server()*" e definido um "*endpoint*" para a conexão do cliente com o servidor.

Posteriormente, é criado um objeto "*Parameters*" e definido uma série de variáveis para armazenamento de parâmetros em tempo real. Estas variáveis incluem informações como temperatura, aceleração, rotação, corrente elétrica, potência, status e horímetro.

As variáveis são adicionadas ao objeto "*Parameters*" utilizando o método "*add_variable()*" com os parâmetros "*addspace*" e o nome da variável.

Definiu-se que as variáveis são graváveis utilizando o método "*set_writable()*", o que significa que é possível atualizar os valores destas variáveis através de uma conexão OPC UA. O servidor é iniciado utilizando o método "*start()*" da instância "*Server()*".

Após inicialização do servidor OPC UA, inicializa-se uma conexão com o InfluxDB. Primeiramente, são definidos alguns parâmetros de configuração do banco de dados, como o token de acesso, a organização e o bucket a ser utilizado para armazenamento dos dados.

Em seguida, é criado um objeto "*InfluxDBClient()*" da biblioteca "*influxdb_client*" para se conectar ao banco de dados InfluxDB. Este objeto recebe como parâmetros a URL do banco de dados e o token de acesso.

O objeto "*write_api*" é inicializado com as opções de escrita síncrona, utilizando o método "*write_api()*" do objeto "*InfluxDBClient()*". Este objeto é utilizado para escrever dados no banco de dados de séries temporais.

O *hostname* da máquina é obtido através do comando "*hostname -I*" e armazenado na variável "*hostname*". Em seguida, é criada uma instância do objeto "*InfluxDBClient()*" com os parâmetros definidos anteriormente e inicializado o objeto "*write_api()*" para escrita síncrona.

Em seguida, define-se callbacks que serão chamados quando eventos específicos ocorrerem durante a execução do cliente MQTT, em resumo tem-se as callbacks “*on_connect*”, “*on_publish*”, “*on_subscribe*” e, uma das mais importantes, chamada “*on_message*”.

- *on_connect*: É chamado quando o cliente MQTT se conecta ao *broker* MQTT. Ele recebe quatro argumentos: “*cliente*”, “*userdata*”, “*flags*” e “*rc*”. “*cliente*” é o objeto cliente que se conectou, “*userdata*” é um objeto definido pelo usuário que é passado ao cliente MQTT durante a inicialização, “*flags*” é um dicionário contendo flags de conexão e “*rc*” é o código de retorno da conexão. O *callback* é utilizado para imprimir uma mensagem informando que a conexão foi estabelecida.
- *on_publish*: É chamado quando uma mensagem é publicada com sucesso no broker MQTT. Ele recebe quatro argumentos: “*cliente*”, “*userdata*”, “*mid*” e “*properties*”. “*cliente*” e “*userdata*” são os mesmos que no *callback* “*on_connect*”. “*mid*” é o identificador da mensagem publicada e “*properties*” é um dicionário opcional contendo as propriedades da mensagem. O *callback* é utilizado para imprimir o identificador da mensagem publicada.
- *on_subscribe*: É chamado quando um tópico é inscrito com sucesso no *broker* MQTT. Ele recebe cinco argumentos: “*cliente*”, “*userdata*”, “*mid*”, “*granted_qos*” e “*properties*”. “*cliente*”, “*userdata*” e “*properties*” são os mesmos que nos *callbacks* “*on_connect*” e “*on_publish*”. “*mid*” é o identificador da inscrição e “*granted_qos*” é a qualidade de serviço concedida pelo *broker* MQTT. O *callback* é utilizado para imprimir uma mensagem informando que o tópico foi inscrito.

A callback chamada “*on_message*” é usada para processar mensagens recebidas em um cliente MQTT. O primeiro parâmetro da função, "*client*", é o objeto de cliente MQTT que recebeu a mensagem. O segundo parâmetro, "*userdata*", é um objeto personalizado que o usuário pode definir ao criar o cliente. O terceiro parâmetro, "*msg*", é a mensagem recebida.

O código nessa callback primeiro obtém a hora atual no formato UTC (*Universal Time Coordinated*) e, em seguida, verifica se a mensagem recebida é do tópico "*esp32/horimetro*" ou não. Se não, ele cria um objeto JSON que contém a medição,

as *tags*, o tempo e o valor da mensagem recebida. Além disso, o código executa uma consulta SQL para inserir os dados recebidos em uma tabela "IOT" no banco de dados.

Se a mensagem recebida for do tópico "*esp32/horimetro*", o código cria um objeto JSON semelhante, mas com o valor convertido em uma *string*. Ele executa uma consulta SQL diferente para atualizar a tabela "*HORIMETRO*" com o valor recebido e a hora atual.

Por conseguinte, o código envia os dados JSON recebidos para o InfluxDB e atualiza os valores do servidor OPC UA para serem usados por outros aplicativos. O valor da mensagem recebida é definido em diferentes variáveis no servidor OPC UA, dependendo do tópico em que a mensagem foi recebida.

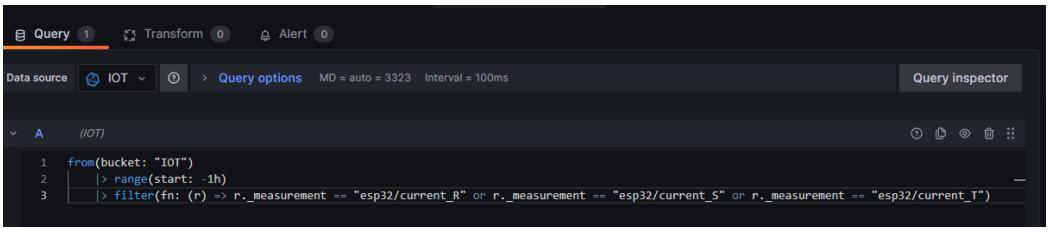
Finalmente, define-se um cliente MQTT chamado “*python_linux*”, realiza-se a conexão com o *broker* MQTT inserindo informações de usuário, senha, endereço *IP* e porta de comunicação. Com a conexão estabelecida com sucesso, subscreve-se o tópico "*esp32/#*" com qualidade de serviço 0 (*QoS 0*). A utilização do caractere curinga "#" permite a inscrição em todos os subtópicos de "*esp32*", sem precisar se inscrever em cada um deles separadamente.

3.9 Dashboards no Grafana

Para construção dos painéis da dashboard no Grafana, utilizou-se *querys* de consulta no banco de dados InfluxDB, o qual já foi configurado previamente. Para construir consultas no InfluxDB, pode-se utilizar a linguagem de consulta InfluxQL ou a nova linguagem de consulta chamada Flux. No Grafana, a linguagem padrão é Flux, que é mais avançada e poderosa do que a InfluxQL.

Por exemplo, para a construção de um painel que exibe os gráficos em tempo real das variáveis de corrente (fase *R*, fase *S*, fase *T*) em um mesmo gráfico, adicionou-se um novo painel e utilizou-se a linguagem Flux para consultar os dados e automaticamente os dados serem exibidos. A query para a elaboração desse exemplo de um dos gráficos da dashboard no Grafana pode ser visualizado na Figura 38.

Figura 38 - Consulta Flux para construção de painel no Grafana



Fonte: Autor (2023)

Essa consulta filtra os dados dos últimos 60 minutos (*start: -1h*) do *bucket* chamado "*IOT*" e filtrando apenas as medições que têm o nome "*esp32/current_R*", "*esp32/current_S*" ou "*esp32/current_T*" (*filter*). O operador “|>” é usado para passar o resultado da expressão anterior para a próxima. No caso, o resultado de “*from*” é passado para “*range*”, que é passado para “*filter*”. E, assim, todas as demais informações da dashboard foram criadas.

3.10 Aplicação web em Flask

Para a criação do script que serve a aplicação em Flask, a priori faz-se necessário importar várias bibliotecas essenciais a nível de *back-end* e *front-end*, para criar um painel de controle interativo e visualmente atraente. Em resumo, tem-se:

- “*dash*”: A principal biblioteca do Dash.
- “*dash.dependencies*”: Fornece as dependências necessárias para definir as entradas e saídas em aplicativos Dash.
- “*dash_core_components*” e “*dash_html_components*”: Fornecem componentes pré-construídos de interface do usuário para criar o *layout* do painel de controle.
- “*plotly*”: Uma biblioteca para criar gráficos interativos e personalizáveis.
- “*plotly.graph_objs*”: Fornece objetos gráficos para criar diferentes tipos de gráficos usando o Plotly.
- “*collections.deque*”: Uma classe de coleção para criar uma fila de duas pontas, que pode ser útil para armazenar e manipular dados.
- “*dash_daq*”: Fornece componentes e controles adicionais de interface do usuário para aplicativos Dash.

- “*Flask*”: Um *framework* da web para Python usado pelo Dash para servir o aplicativo.
- “*flask.jsonify*” e “*flask.request*”: Utilitários para trabalhar com dados JSON e lidar com solicitações HTTP no Flask.
- “*dash_bootstrap_components*”: Estende o Dash com componentes e estilos adicionais do *framework* Bootstrap.
- “*dash_bootstrap_templates*”: Fornece *templates* e temas para aplicativos Dash.
- “*dash_table*”: Um componente para exibir tabelas de dados em aplicativos Dash.
- “*plotly.express*”: Uma interface de alto nível para o Plotly, para criar visualizações interativas com uma sintaxe simplificada.
- “*datetime*” e “*timedelta*”: Módulos para trabalhar com datas e horários em Python.
- “*numpy*” e “*pandas*”: Bibliotecas para trabalhar com dados numéricos e tabulares em Python.
- “*pyodbc*”: Uma biblioteca Python para conectar-se e interagir com bancos de dados usando ODBC.
- “*plotly.subplots*”: Um módulo para criar *subplots* no Plotly.

Posteriormente, são definidas as configurações de conexão para um banco de dados SQL Server e a criação de um servidor Flask para hospedar o aplicativo Dash.

As configurações de conexão do SQL Server incluem:

- “*sql_server*”: O endereço IP ou o nome do host do servidor SQL Server.
- “*sql_database*”: O nome do banco de dados que será acessado.
- “*sql_username*”: O nome de usuário usado para autenticar no servidor SQL Server.
- “*sql_password*”: A senha usada para autenticar no servidor SQL Server.

Em seguida, cria-se uma instância do objeto Flask chamada “*server_flask*”. O objeto Flask é necessário para executar o aplicativo Dash. Em seguida, é criada uma instância do objeto Dash chamada “*app*”, o parâmetro “*external_stylesheets*” é usado para adicionar o tema "*YETI*" do framework Bootstrap aos estilos do aplicativo. O objeto “*server_flask*” é passado como parâmetro para o objeto *app* para vinculá-los juntos. Isso permite que o servidor Flask seja usado para hospedar o aplicativo Dash.

Essa configuração prepara o ambiente para construir o painel de controle usando o Dash e estabelecer os dados para conexão com o banco de dados SQL Server.

Logo após, são definidos estilos de formatação em CSS (*Cascading Style Sheet*), para os elementos visuais do aplicativo Dash. Esses estilos são usados para controlar o *layout*, a aparência e a formatação dos componentes na interface do usuário. Por exemplo, "*CONTENT_STYLE*" define o estilo do conteúdo principal do aplicativo, define as margens, o preenchimento e o espaço em branco do conteúdo, enquanto que "*MENU_STYLE*" define o estilo do menu.

Posteriormente, são definidos alguns componentes de navegação e interação com o usuário para o aplicativo Dash. Esses componentes incluem itens de menu, *links*, botões e outros elementos visuais. Acerca dos componentes definidos tem-se:

- *nav_item*: Define um item de menu de navegação que contém um *link* de e-mail para suporte. O link de e-mail abre o programa de e-mail padrão com um destinatário, um assunto e um corpo pré-preenchidos.
- *nav_item2*: Define um segundo item de menu de navegação que contém um link externo para a plataforma Grafana. O *link* redireciona para um site externo.
- *dropdown*: Define um menu suspenso de navegação que contém várias opções, incluindo "*About*", "*Help*", "*Logout*" e opções de mudança de tema. As opções de mudança de tema são representadas por um componente "*ThemeSwitchAIO*" que permite ao usuário alternar entre diferentes temas.
- *dropdown2*: Define um segundo menu suspenso de navegação que contém opções para selecionar diferentes temas. Cada opção é representada por um item de menu que corresponde a um tema específico.
- *datepicker*: Define um seletor de data que permite ao usuário selecionar um intervalo de datas. O seletor é representado pelo componente "*dcc.DatePickerRange*" e inclui opções de personalização, como a formatação da data e os limites mínimo e máximo de data.
- *logo*: Define a barra de navegação superior do aplicativo, que inclui um logotipo, um título e os itens de menu e interação com o usuário definidos anteriormente. O logotipo é representado por um componente "*html.Img*"

e o título é representado por um componente “*dbc.NavbarBrand*”. Os itens de menu e interação são organizados em uma barra de navegação usando o componente “*dbc.Nav*” e são controlados por um componente “*dbc.Collapse*”.

Esses componentes de navegação e interação são adicionados ao layout do aplicativo Dash para permitir que o usuário acesse diferentes seções do aplicativo, interaja com os elementos e faça seleções específicas, como alterar temas ou selecionar datas.

Em seguida, o *layout* do aplicativo Dash é definido como uma estrutura hierárquica de componentes HTML, em resumo, têm-se:

- *html.Div*: É o elemento raiz do *layout* e contém todos os outros componentes do aplicativo Dash
- *logo*: É a barra de navegação superior do aplicativo, que foi definida anteriormente.
- *html.Div(id="sidebar")*: É um elemento vazio que será usado para exibir o menu lateral com as opções de temas.
- *dbc.Row*: É uma linha que contém uma grade de colunas. Neste caso, existem várias colunas que exibem informações em forma de cartões e gráficos.
- *dbc.Col*: É uma coluna que contém conteúdo. Cada coluna representa um cartão ou gráfico que exibe informações específicas.
- *html.H5*: É um cabeçalho HTML de nível 5 que exibe um título para cada cartão.
- *html.H5(id='card1', className="card-title", style=STYLE_CARDS)*: É um elemento HTML que exibe o conteúdo dinâmico do cartão. O valor desse elemento é atualizado com base em uma função callback definida posteriormente.
- *dcc.Loading*: É um componente que exibe um ícone de carregamento enquanto um determinado conteúdo está sendo carregado. Neste caso, é usado para envolver o gráfico em cada coluna, fornecendo uma experiência visual agradável para o usuário.

- *dcc.Graph(id="graph")*: É um componente que exibe um gráfico. O *id* é usado para identificar o gráfico e conectar a função *callback* que gera o gráfico.
- *html.Br()*: É um elemento HTML que adiciona uma quebra de linha entre os elementos.

A estrutura do *layout* é dividida em várias seções, incluindo uma linha de cartões, análise de gráficos de séries temporais de várias variáveis e análises no domínio de frequência. Cada seção é organizada em linhas e colunas para criar um *layout* responsivo e fácil de usar.

O callback "*update_output*" é acionado quando as datas de início e término são alteradas no componente "*my-date-picker-range*". Ele recebe as datas selecionadas como entrada e retorna seis figuras que serão usadas para atualizar os gráficos no *layout*.

Dentro do *callback*, as datas são convertidas para objetos date usando "*date.fromisoformat*" e, em seguida, são formatadas em *strings* no formato "*YYYY-MM-DD HH:MM:SS*" usando "*strftime*". Essas *strings* são usadas para construir as consultas SQL para recuperar os dados relevantes do banco de dados.

A conexão é estabelecida com o banco de dados usando o "*pyodbc.connect*", e as consultas são executadas usando "*pd.read_sql*" para obter os dados em um *DataFrame* do Pandas.

Para cada figura, é utilizado o "*px.line*" ou "*px.area*" para criar um gráfico de linha ou área com base nos dados do *DataFrame*. As configurações de layout também são definidas para cada figura, como título do eixo y, cores, marcadores, margens e cores de fundo.

Após a criação das figuras, a conexão com o banco de dados é encerrada chamando "*cursor.close()*" e "*conn.close()*". Por fim, na callback "*update_output*", as seis figuras são retornadas como saída do callback, atualizando assim os gráficos no *layout* do aplicativo Dash. As seis figuras retornadas, respectivamente, são: gráfico de aceleração triaxial, gráfico de velocidade angular triaxial, gráfico de corrente trifásica, gráfico de temperatura, desbalanceamento de fase e desbalanceamento de fase com filtro de Kalman e tempo de operação.

O *callback* "*update_output2*" funciona de forma similar ao *callback* "*update_output*", ele é acionado quando as datas de início e término são alteradas no

componente “*my-date-picker-range*”, recebe as datas selecionadas como entrada e retorna uma figura (*figure*) com um *subplot* contendo seis gráficos de dispersão.

Nesse *callback*, para cada medida de aceleração e rotação, é realizado um cálculo da Transformada Rápida de Fourier (FFT). O sinal de entrada é extraído do *DataFrame* para cada medida, e o comprimento da amostra, as frequências e os valores da FFT são calculados usando “*np.fft.fftfreq*” e “*np.fft.fft*”. Os resultados são armazenados em *DataFrames* separados.

Em seguida, as seis figuras de dispersão são adicionadas ao *subplot* usando “*fig11.add_trace*”. Cada figura é baseada nos dados da FFT correspondentes e é configurada para exibir pontos e linhas de dispersão usando “*go.Scatter*”. O layout da figura é atualizado para definir a cor de fundo do papel e do gráfico como transparente, usando “*update_layout*”. A figura “*fig11*” é retornada como saída do *callback*, atualizando assim o gráfico no layout do aplicativo Dash.

O *callback* “*update_output_div*” é acionado quando as datas de início e término são alteradas no componente “*my-date-picker-range*”. Ele recebe as datas selecionadas como entrada e retorna várias saídas que serão atualizadas em elementos específicos do layout do aplicativo Dash. Os *DataFrames* “*df_temp*”, “*df_ikf*” e “*df_horimetro*” contêm os dados de temperatura, desequilíbrio estimado e valor do horímetro, respectivamente.

Em seguida, são calculadas as médias dos valores de temperatura (*mean_temp*) e desequilíbrio estimado (*mean_ifk*) usando “*np.mean*”. Os resultados são arredondados para duas casas decimais. Posteriormente, é realizada uma consulta para obter o status atual do LED (*power*) a partir do *DataFrame* “*df_status*”. Se o valor for igual a 1, o “*status_led*” é definido como “*True*” e o “*status_text*” como “*Power ON*”. Caso contrário, o “*status_led*” é definido como “*False*” e o “*status_text*” como “*Power OFF*”.

A última atualização (*last_update*) é definida como a data e hora atuais, convertida em uma *string* formatada usando “*datetimenow.strftime('%Y-%m-%d %H:%M:%S')*”. As saídas do *callback* são retornadas na seguinte ordem: “*status_led*” para atualizar o valor do indicador, “*status_text*” para atualizar o texto de status, “*last_update*” para atualizar o conteúdo do cartão 1, “*mean_temp*” para atualizar o conteúdo do cartão 2, “*mean_ifk*” para atualizar o conteúdo do cartão 3 e “*df_horimetro.value_*” para atualizar o conteúdo do cartão 4.

Por fim, o código “*if __name__ == '__main__':*” é uma condição que verifica se o *script* está sendo executado diretamente, ou seja, se é o ponto de entrada principal do programa.

Dentro desse bloco condicional, a função “*app.run_server()*” é chamada para iniciar o servidor do aplicativo Dash. O parâmetro “*host="0.0.0.0"*” especifica que o aplicativo será executado no endereço *IP* do servidor atual, tornando-o acessível a partir de qualquer endereço *IP*. O parâmetro “*port="8051"*” define a porta em que o servidor será executado.

Quando o *script* é executado diretamente, ou seja, quando é chamado diretamente pelo interpretador Python, o código dentro do bloco é executado. Isso inicia o servidor do aplicativo Dash e faz com que o aplicativo seja executado e permaneça em execução até ser interrompido.

Portanto, quando se executa esse *script* Python, o servidor do aplicativo Dash será iniciado e o aplicativo estará disponível em “*http://<IP DO SERVIDOR>:8051/*”. Os usuários podem acessar o aplicativo por meio desse URL para interagir com a interface e visualizar os gráficos e dados atualizados.

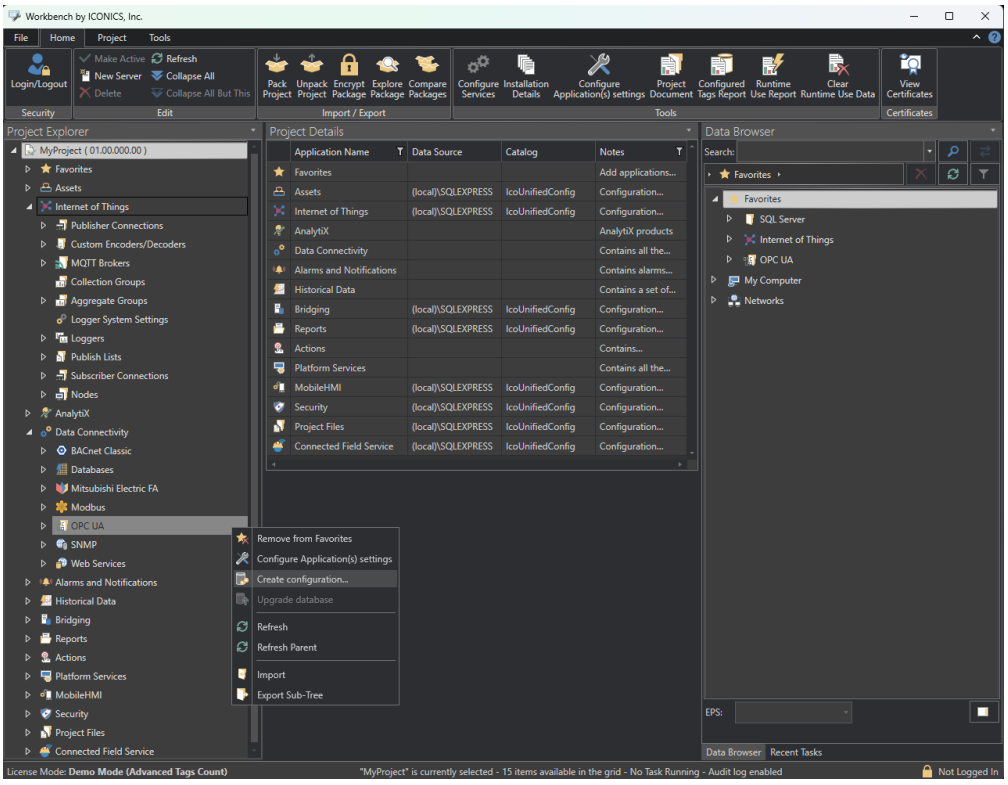
3.11 ICONICS GENESIS64

A aplicação do supervisório foi desenvolvida em no servidor Windows e um computador cliente. Para que os dados sejam visualizados em tempo real, basicamente conectou-se a aplicação desse supervisório ao servidor OPC UA configurado pelo script “*BRIDGE.py*”.

A priori, realizou-se o download e instalação do Genesis64, disponível no site oficial da ICONICS. A aplicação é licença paga, mas gratuito para download e demonstração por algumas horas, sujeito a renovação desse período trial.

Após a instalação completa, configurou-se a fonte de dados do servidor OPC UA. Ao abrir o software “*Workbench*”, na lateral aparecerá uma lista de tipos de conexão. Na opção “*OPC UA*”, ao clicar com o botão direito do mouse, aparecerá várias opções, conforme Figura 39, e ao selecionar “*Create configuration...*” uma nova janela abre-se, para realizar a configuração da fonte de dados OPC UA, e deve-se inserir informações do endereço *IP* e porta do servidor.

Figura 39 - Configuração fonte de dados Workbench



Fonte: Autor (2023)

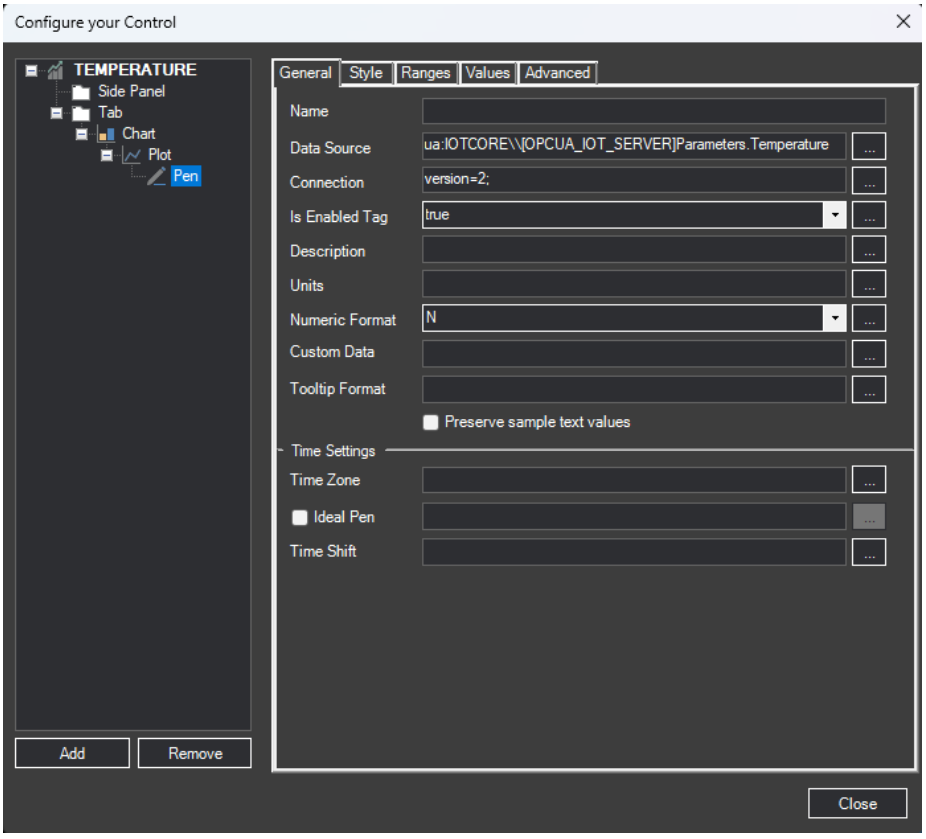
Após a configuração da fonte de dados OPC UA através do Workbench, pode-se realizar a construção do painel supervisório através do software GraphWorX64. O software oferece vários recursos, e que posteriormente podem ser utilizados. Mas, a princípio, o trabalho tem como intuito exibir as informações lidas do DAQ-IOT em gráficos de séries temporais, similar a aplicação web desenvolvida com Flask.

Para a construção dos gráficos em tempo real, no GraphWorX64, na aba "*Controls*", selecionou-se a opção "*TrendWorX64 Viewer*", essa opção permite a criação de gráficos em forma de séries temporais. Além disso, através da aba "*Controls*", utilizou-se outros recursos para construção do painel, como cards e tabelas. A criação é bem simples, no formato "*clique e arraste*", facilitando o desenvolvimento painel de forma rápida e visual, sem precisar de programação ou códigos complexos.

Ao entrar nas propriedades do gráfico, várias opções foram configuradas, mas para fins de layout e aspectos visuais, como cor, tamanho da fonte, tipo de fonte, legendas, entre outras opções. Mas, uma configuração em especial é a mais importante, a qual trata-se da configuração da fonte de dados. Para inserir uma linha

no gráfico de uma variável, deve-se adicionar um novo “*plot*” e em seguida adicionar um novo “*pen*”, que trata-se justamente de uma série temporal. A configuração da fonte de dados para o “*pen*”, é definida em “*Data Source*”, conforme Figura 40, na qual ilustra a configuração da série temporal que exibe os dados de temperatura.

Figura 40 - Configuração de dados para gráficos



Fonte: Autor (2023)

De modo similar, todos os demais gráficos foram construídos baseados nesse princípio. Ou seja, apenas selecionar o tipo de visual, configurar a fonte de dados que será exibida no visual, e realizar as configurações de aparência, como cor, textos, legendas, entre outras opções disponíveis. Vale ressaltar que o GraphWorX64 é apenas uma das mais variadas formas de construir supervisórios com o Genesis64, o qual possui recursos de desenvolvimento de telas web e mobile. Além disso, a proprietária ICONICS possui uma gama de outros softwares que permitem a construção de telas de supervisão modernas.

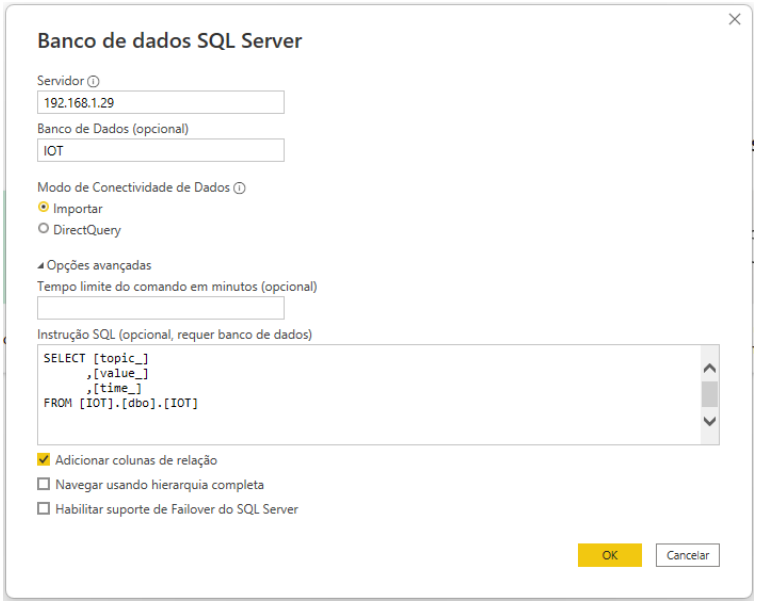
3.12 Power BI

Para a utilização do Power BI como meio de exibição de painéis do sistema IOTCORE, necessita-se realizar uma configuração similar a aplicação web Flask, para a fonte de dados. Ou seja, é necessário configurar o banco ao Power BI.

A priori, vale ressaltar que o Power BI, assim como o GENESIS64, foi instalado apenas no servidor Windows e em uma máquina cliente na rede com SO Windows, pois o software exige um sistema operacional Windows. Realizou-se então o download e instalação do Power BI, através do site oficial da Microsoft.

Em seguida, ao abrir o Power BI, na aba "*Página Inicial*" e categoria "*Dados*", selecionou-se a opção "*Obter dados*", e em seguida "*Banco de dados SQL Server*". Ao clicar em "*Conectar*", uma nova janela surge, para inserir as credenciais do banco de dados, conforme ilustra a Figura 41, com as informações inseridas.

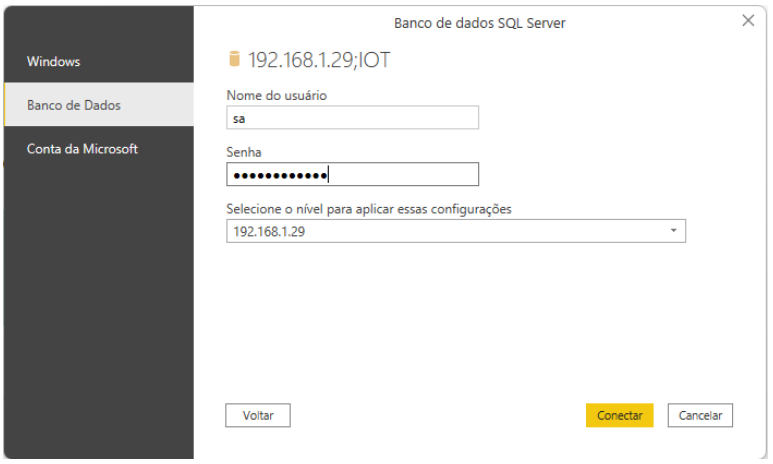
Figura 41 - Configuração banco de dados SQL Server no Power BI



Fonte: Autor (2023)

Em seguida, é solicitado as credenciais para conecta-se ao banco. Selecionou-se a opção "*Banco de Dados*", e nos campos destinados para autenticação foram inseridos o nome de usuário e senha, conforme Figura 42.

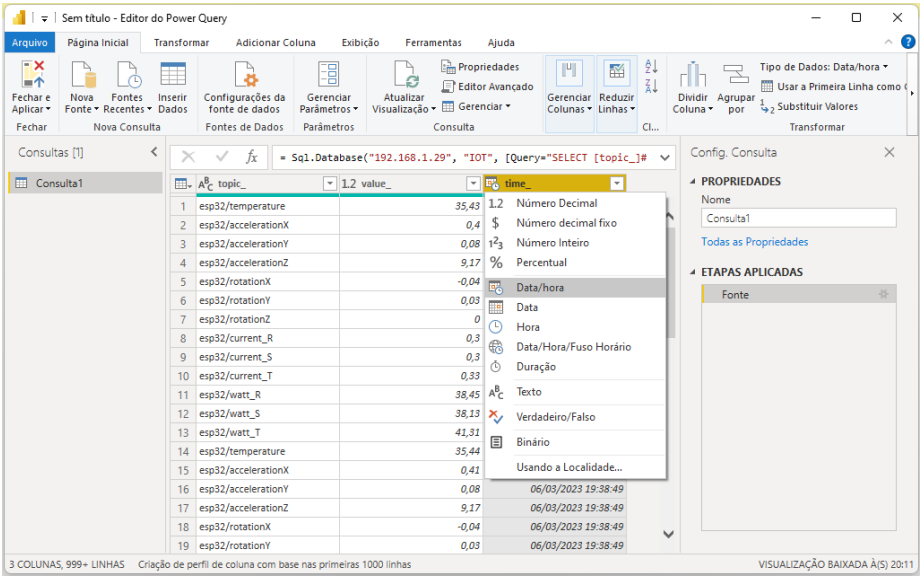
Figura 42 - Inserção das credenciais para conexão ao banco no Power BI



Fonte: Autor (2023)

Ao preencher os campos e selecionar "*Conectar*", os dados da tabela são pré-visualizados, com a opção de carregar diretamente os dados, ou realizar uma transformação nos dados. Como boa prática, selecionou-se a opção de transformar dados, pois algumas definições são importantes, como definir o tipo de variável, como texto, decimal, inteiro, entre outros, como ilustra a Figura 43.

Figura 43 - Configuração dos dados no Power Query

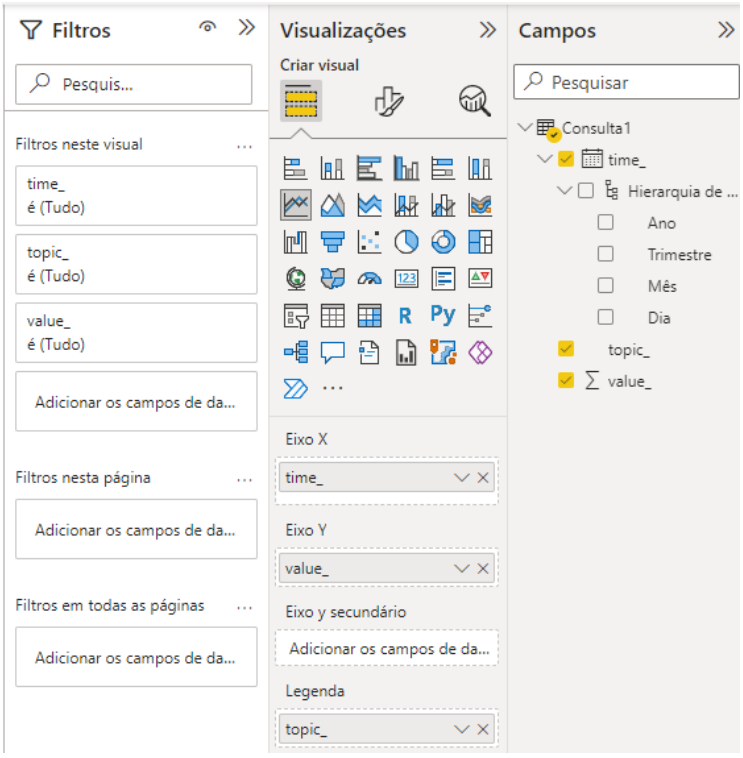


Fonte: Autor (2023)

Após configurar as colunas "*topic_*", "*value*" e "*time_*" respectivamente como texto, decimal e data/hora, carregou-se os dados para o relatório no Power BI. A criação de relatórios torna-se bem simples, similar ao GENESIS64, no formato "*clique*

e arraste". O Power BI possui uma aba com os elementos visuais disponíveis, e loco ao lado a fonte de dados que foi carregada do banco de dados, conforme ilustrado na Figura 44, a qual exibe a configuração para geração de todos os tópicos no tempo. Outros elementos como filtros, tabelas, entre outros foram utilizados.

Figura 44 - Modo de criação dos painéis no Power BI



Fonte: Autor (2023)

Após a construção da dashboard com os dados do banco de dados SQL, publicou-se o relatório do Power BI, o qual pode então ser acessado através de uma URL em qualquer browser em diversos computadores cliente.

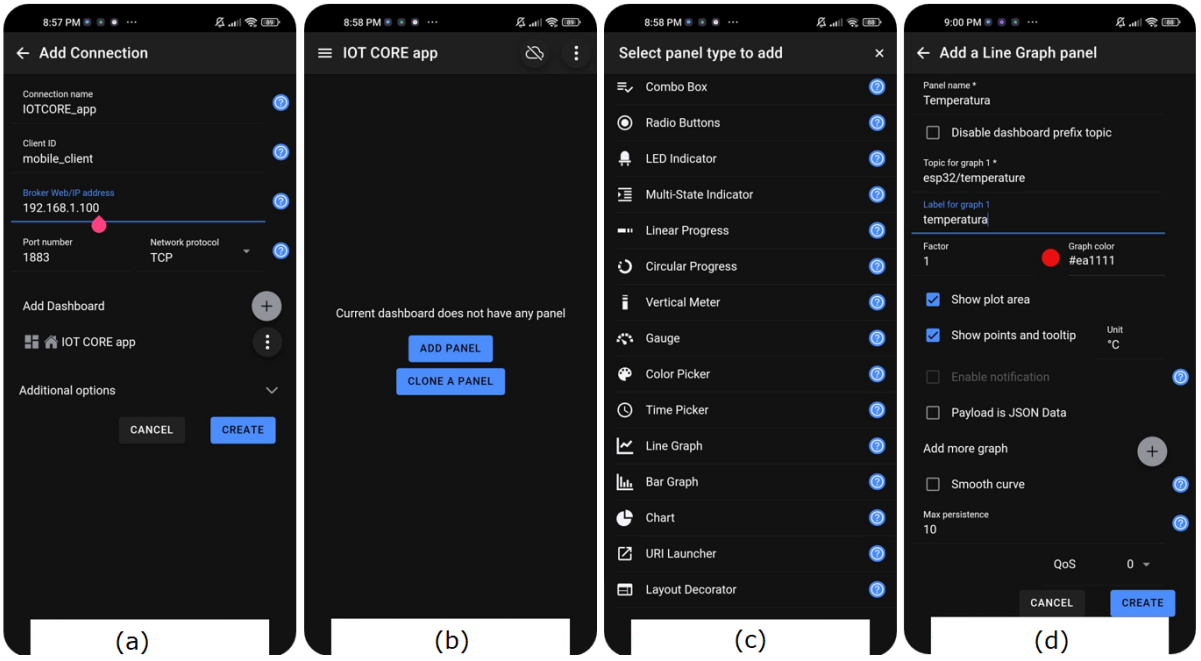
A atualização dos dados ocorre de maneira automática, pois a fonte de dados é conectada diretamente ao banco de dados SQL Server, não necessitando publicar novamente o relatório na web.

Com isso, pode-se realizar consultas em tempo real dos dados, de forma similar a consulta SQL realizada no aplicativo web com Flask. Ambas as soluções se tornam viáveis do ponto de vista de implementação, apesar das dificuldades atreladas a cada uma delas, posteriormente discutidas.

3.13 Aplicação Mobile

Para a visualização dos dados em tempo real, através do protocolo de comunicação MQTT, utilizou-se uma aplicação mobile chamada “*IoT MQTT Panel*”. Essa aplicação permite conectar-se a um broker MQTT e construir painéis sofisticados a partir da assinatura dos tópicos. A Figura 45 ilustra os passos para a configuração de um painel para exibir o gráfico em série temporal do tópico de temperatura, coletado pelo DAQ-IOT.

Figura 45 - Configuração painel mobile MQTT



Fonte: Autor (2023)

A priori, faz-se necessário adicionar uma nova conexão, conforme Figura 45 (a), o campo “*Connection name*” fica a critério do usuário, para a aplicação escolheu-se “*IOTCORE_app*”, “*Client ID*” é um campo que também fica a critério do usuário, porém deve ser único. Em seguida, preencheu-se o campo “*Broker Web/IP address*” com o endereço *IP* do *broker* MQTT, ou seja, com o endereço *IP* do Raspberry Pi. O campo “*Port number*” e “*Network protocol*” foram configurados como “*1883*” e “*TCP*”, respectivamente. Após realizar o preenchimento dos campos, selecionou-se a opção “*Add Dashboard*”, para poder criar um painel atrelado a essa configuração de cliente

MQTT. Ao selecionar “*CREATE*”, a dashboard criada aparecerá, conforme Figura 45 (b).

Para adicionar os elementos visuais na dashboard, selecionou-se a opção “*ADD PANEL*”, e uma lista de opções de elementos visuais são disponíveis para seleção, conforme ilustrado na Figura 45 (c). Para criar o gráfico em série temporal da temperatura, selecionou-se a opção “*Line Graph*”, e na tela da Figura 45 (d), pode-se realizar a configuração dos dados desse tipo de painel selecionado.

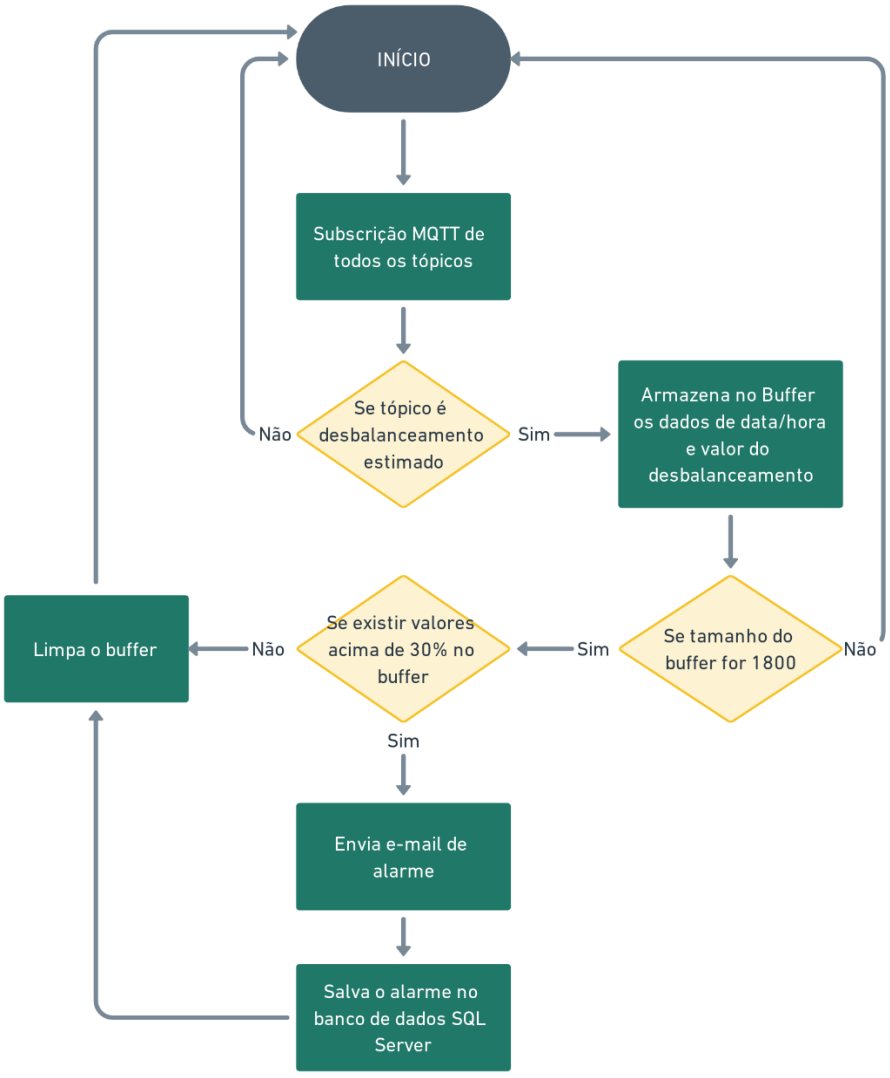
Para configurar a fonte de dados de temperatura, basicamente preencheu-se os campos “*Panel name*” com o nome da variável a ser exibida, ou seja, “*Temperatura*”, e em seguida preencheu-se o campo “*Topic for graph 1*” com o tópico MQTT que está configurado a variável *temperatura*, ou seja, o tópico “*esp32/temperature*”. Outras configurações como cor da linha, tamanho da linha, preencher a área abaixo do gráfico, exibir os valores lidos, unidade de engenharia e quantidade de pontos exibidos no gráfico, são também opcionalmente configurados.

Desse modo, todos os outros demais painéis, para as demais variáveis, são construídos no aplicativo mobile. Vale ressaltar que, para que essa leitura dos dados seja possível, o dispositivo mobile deve estar conectado na mesma rede Wi-Fi que o *broker* MQTT. Para casos de *broker* em nuvem, essa restrição não se faz necessária, necessitando apenas de acesso à internet.

3.14 Definição de Alarmes

Os sistemas supervisórios desempenham um papel crucial no monitoramento e controle de processos industriais. Uma das funcionalidades essenciais desses sistemas é a geração de alarmes para notificar os operadores e engenheiros sobre eventos críticos ou situações anormais. Com isso, elaborou-se um código para geração de alarmes via e-mail, o *script* em Python lê dados de um tópico MQTT, verifica o valor máximo e horário de ocorrência, salva as informações em um banco de dados SQL Server e envia um alerta por e-mail com essas informações. Para fins de testes, utilizou-se a variável de desbalanceamento de fase com filtro de Kalman, para verificação de anomalias, e a cada 30 minutos envia um alerta via e-mail caso haja valores acima do valor limite de 30%. A Figura 46 ilustra o fluxo para a geração desses alarmes.

Figura 46 - Fluxograma para geração de alarme



Fonte: Autor (2023)

O valor adotado de 1800 pontos para o buffer deve-se ao tempo necessário de 30 minutos, visto que a aquisição dos dados de cada variável ocorre a cada segundo. Desse modo, 30 minutos representa 1800 segundos.

A nível de manutenção, teve-se o cuidado de não somente enviar os alertas de anomalias, mas também enviou-se no e-mail breves orientações de como corrigir o problema. Desse modo, definiu-se o título do e-mail como: *"⚠ ALARME IOT CORE! Valor máximo de desbalanceamento de fase lido: {valor_maximo}. Ocorreu às: {hora_ocorrencia}*" e o corpo do e-mail como:

1) *Verifique a fonte de alimentação: Verifique se a rede elétrica está fornecendo uma tensão equilibrada nas três fases. Verifique a tensão de linha e a tensão de fase em cada fase usando um voltímetro adequado. Se houver uma diferença significativa nas tensões de fase, entre em contato com a companhia de energia elétrica para solucionar o problema.*
2) *Verifique a conexão do motor: Verifique as conexões elétricas do motor e certifique-se de que estão corretas. Verifique se não há fios soltos ou conexões defeituosas. Um problema de conexão incorreta pode levar ao desbalanceamento de fase.*
3) *Verifique os componentes do sistema: Verifique os componentes do sistema, como contatores, relés de sobrecarga e dispositivos de proteção térmica. Certifique-se de que estão funcionando corretamente e não estão causando desequilíbrio nas correntes trifásicas.*
4) *Verifique o motor: Se todas as outras possibilidades forem descartadas e o desbalanceamento de fase persistir, pode haver um problema interno no motor. Nesse caso, é recomendado entrar em contato com um técnico especializado em motores elétricos para fazer uma análise mais detalhada e realizar as medidas corretivas necessárias.*

Assim, o script pode ser adotado para todas as demais variáveis. Vale ressaltar que o valor limite, também conhecido como *threshold*, a nível de implementação mais robusta, deve ser armazenado em banco de dados, e elaborado maneira fácil de modificar os valores, visto que esses limites são definidos pela operação. A nível de testes, o valor de limite foi definido em própria linha de código.

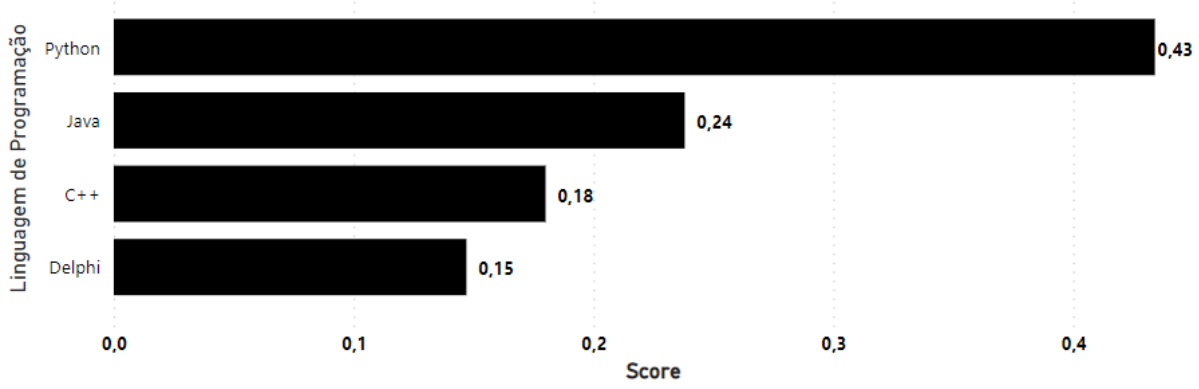
Para o envio de alertas via e-mail, utilizou-se a biblioteca “*win32*”, mas poderia utilizar-se a biblioteca “*smtp*” (*Simple Mail Transfer Protocol*). A principal diferença entre essas abordagens é que a biblioteca SMTP permite enviar e-mails diretamente usando um servidor SMTP específico, enquanto o pacote *win32* usa o cliente de e-mail padrão do Windows para enviar o e-mail. Isso significa que o pacote *win32* aproveita a configuração e as preferências do cliente de e-mail do usuário, como autenticação, assinatura e configurações de servidor.

4 RESULTADOS

4.1 Decisão Multicritério

O algoritmo de decisão multicritério foi utilizado para definir um conjunto de tecnologias para a construção de um sistema supervisório. Todavia, apenas utilizou-se os resultados de linguagem de programação para adotar no desenvolvimento da aplicação. Para a linguagem de programação, o Python mostrou-se mais promissor a ser adotado, conforme ilustra Figura 47.

Figura 47 - Resultado AHP para linguagem de programação



Fonte: Autor (2023)

A linguagem de programação em Python mostrou-se mais promissora do ponto de vista de performance, confiabilidade, legibilidade, custo, suporte, flexibilidade, simplicidade e experiência do desenvolvedor.

Desse modo, a principal linguagem utilizada foi Python, para salvar e ler dados de banco de dados, para ler variáveis através do protocolo de comunicação MQTT, para servir dados através do protocolo de comunicação OPC UA, enviar e-mail, entre outras funcionalidades. Além disso, Python mostra-se extremamente promissor para escalar a aplicação com algoritmos de *machine learning*. A opção pela linguagem Python ofereceu maior escalabilidade e flexibilidade ao sistema.

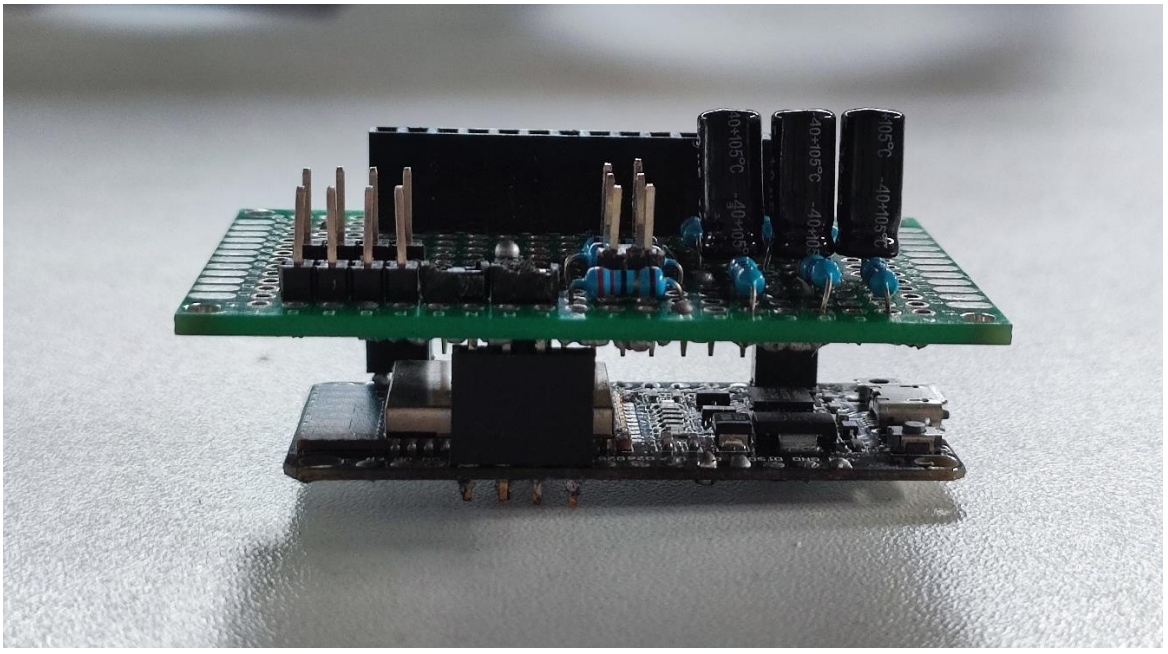
4.2 DAQ-IOT

O DAQ-IOT (*Data Acquisition - Internet of Things*) é um dispositivo composto por um módulo principal e periféricos, que desempenham um papel fundamental na coleta de dados da aplicação IOTCORE. O módulo principal, que serve como o núcleo do sistema, apresenta um design compacto com um case de plástico medindo aproximadamente *82x52x32mm*.

Esse módulo principal possui conectores *jack P2* fêmea que permitem a conexão dos periféricos, ou seja, dos sensores responsáveis pela aquisição dos dados. Ele é equipado com um ESP32, um microcontrolador poderoso e versátil, que realiza todo o processamento dos dados coletados pelos sensores.

Para garantir um funcionamento estável e seguro do ESP32, o módulo principal utiliza uma placa de circuito impresso (PCB) como uma espécie de "*shield*". Essa PCB é do tipo ilhada e foi cuidadosamente projetada para abrigar o ESP32 e fornecer a conexão adequada com os demais componentes. Durante o processo de fabricação, as trilhas da PCB são soldadas com precisão utilizando uma estação de solda, garantindo uma conexão sólida e confiável. A Figura 48 ilustra a PCB e o microcontrolador ESP32, componentes eletrônicos internos do módulo principal.

Figura 48 - PCB e microcontrolador ESP32



Fonte: Autor (2023)

Na Figura, nota-se a presença de pinos macho, eles servem para conexão interna dos conectores jst dos sensores de corrente, sensor de vibração e display OLED (*Organic Light-Emitting Diode*). Além disso, a utilização dessa arquitetura permite um fácil encaixe dos componentes, facilitando a instalação e manutenção.

Os periféricos incluídos no DAQ-IOT são três sensores SCT-013 e um sensor MPU6050. Os sensores SCT-013 são utilizados para medir correntes elétricas, enquanto o sensor MPU6050 é um dispositivo capaz de realizar medições de aceleração, rotação e temperatura. Vale ressaltar que, coletor de temperatura e vibração, foi disposta em uma base metálica cilíndrica com base magnética, e isso permite a fácil fixação do sensor ao motor. A Figura 49 ilustra o DAQ-IOT completo.

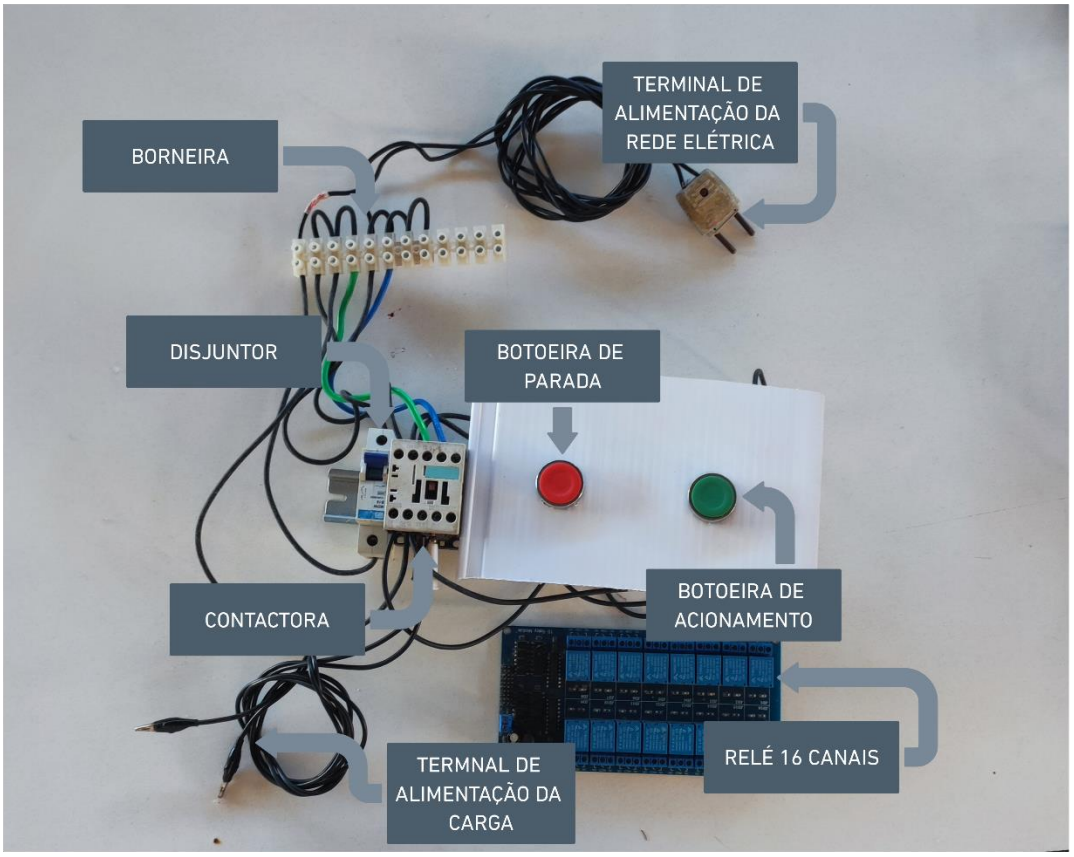
Figura 49 - DAQ-IOT

Fonte: Autor (2023)

4.3 Circuito de Comando Auxiliar

O circuito de comando auxiliar é um circuito básico para partida direta da carga para realização dos testes, ele confere maior segurança nos experimentos em ambiente controlado. Desse modo, montou-se o circuito com cabos, contactora e disjuntor elétrico. Além disso, incluiu-se no circuito uma placa de relé com 16 canais, para em trabalhos futuros controlar o acionamento e parada da carga através do DAQ-IOT, bem como implementar lógicas para intertravamento. A Figura 50 ilustra o circuito de comando auxiliar montado.

Figura 50 - Circuito de comando auxiliar



Fonte: Autor (2023)

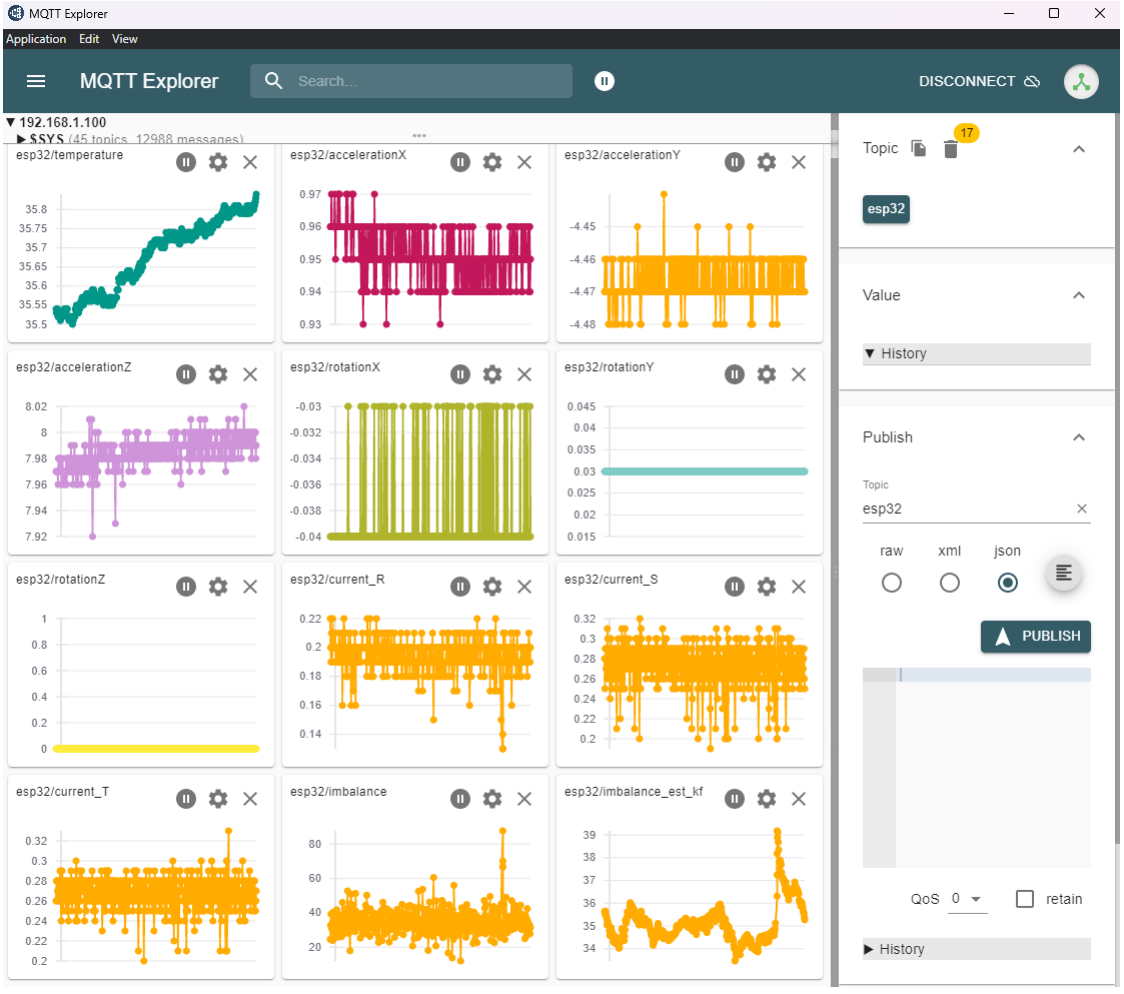
4.4 Comunicação via protocolo MQTT

O MQTT Explorer é uma ferramenta altamente eficaz para validar a comunicação MQTT. Antes de desenvolver softwares personalizados para supervisionar os dados, é fundamental garantir que os dados estejam disponíveis e acessíveis de maneira fácil. Com o MQTT Explorer, basta fornecer o endereço do broker MQTT, a porta de comunicação e as credenciais adequadas para autenticar e visualizar os dados disponíveis no broker. Essa ferramenta simplifica o processo de validação da comunicação MQTT, permitindo verificar rapidamente se os dados estão sendo corretamente publicados e recebidos, antes de prosseguir com a elaboração de soluções mais específicas para a supervisão dos dados. Desse modo, após autenticar no *broker* MQTT hospedado no Raspberry Pi, pode-se visualizar os dados em tempo real transmitidos pelo *broker*, conforme ilustram as Figuras 51 e 52.

Figura 51 - Dados transmitidos através do *broker* MQTT

Fonte: Autor (2023)

Figura 52 - Exibição gráfica dos dados no broker MQTT

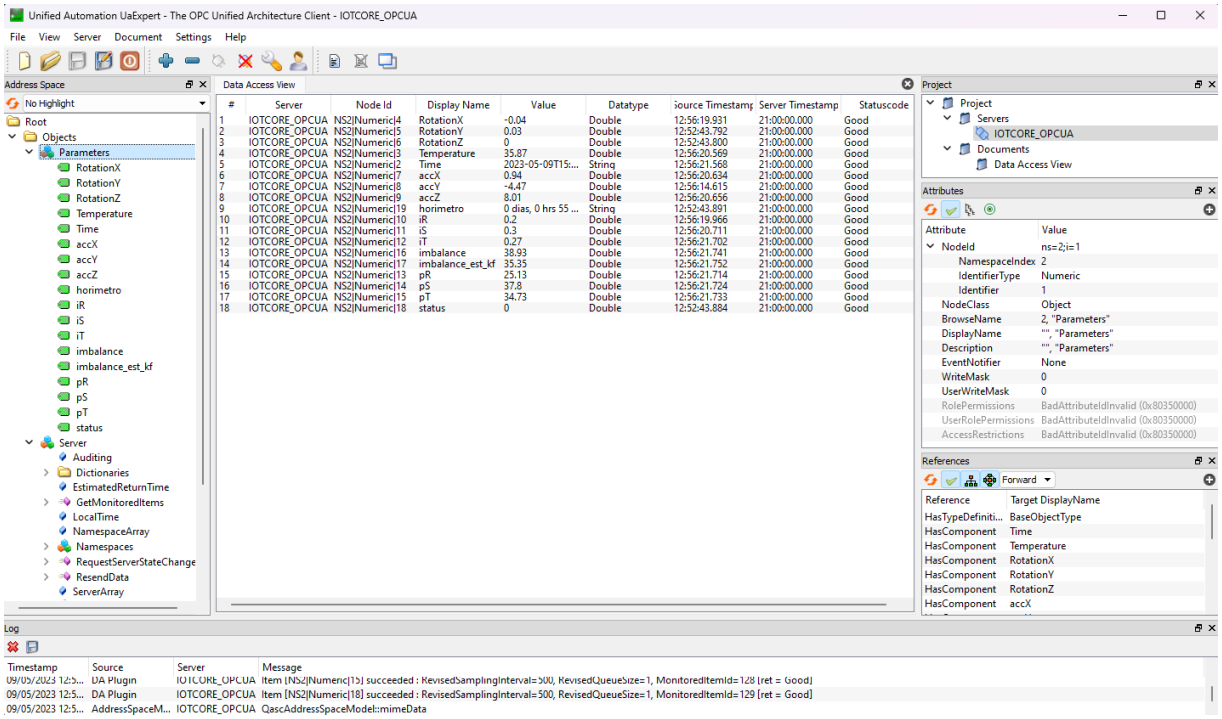


Fonte: Autor (2023)

4.5 Comunicação via protocolo OPC UA

O UA Expert é uma ferramenta essencial para validar a comunicação OPC UA de forma fácil e rápida, assim como o MQTT Explorer para MQTT. Ao fornecer o endereço do servidor OPC UA, ou seja, o endereço da máquina em que o *script BRIDGE.py* está sendo executado, o UA Expert permite visualizar e validar os dados recebidos em tempo real através do servidor OPC UA. A Figura 53 ilustra o resultado dos dados recebidos, proporcionando uma visão clara e imediata dos dados transmitidos pelo DAQ-IOT. Com essa ferramenta, é possível verificar a correta troca de informações entre os dispositivos e o servidor OPC UA, assegurando assim o bom funcionamento da comunicação OPC UA no sistema.

Figura 53 - Dados transmitidos via OPC UA



Fonte: Autor (2023)

Nota-se que os nomes das variáveis transmitidas são as mesmas configuradas pelo servidor OPC UA através do script *BRIDGE.py*, além disso o tempo de resposta foi instantâneo, com a taxa de transmissão de acordo com o tempo de atualização e transmissão definido no *script*, a conexão permaneceu estável e casada com a atualização dos dados através do protocolo MQTT.

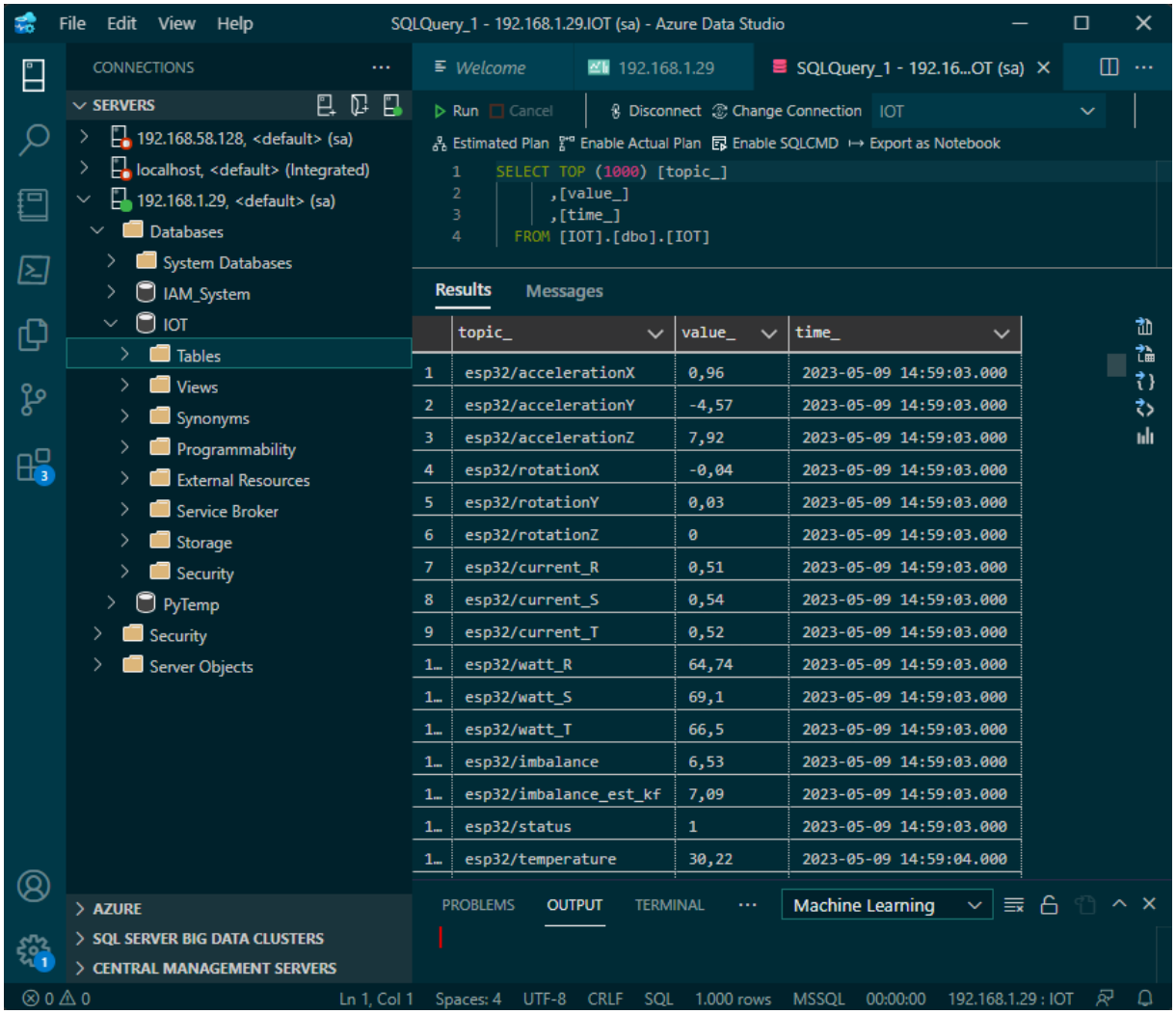
4.6 Armazenamento dos dados

4.6.1 SQL Server

Após a validação da disponibilidade dos dados através de comunicação MQTT e OPC UA, faz-se necessário validar o armazenamento desses dados. Para o armazenamento utilizou-se duas opções, um banco de dados relacional (SQL Server) e um banco de dados de séries temporais (InfluxDB).

O Azure Data Studio é uma ferramenta poderosa para validar e visualizar os dados armazenados no SQL Server. Utilizou-se o mesmo para acessar os dados. Ao navegar até a tabela "*IOT*", pode-se visualizar os dados, conforme ilustra a Figura 54.

Figura 54 - Dados armazenados no banco SQL Server

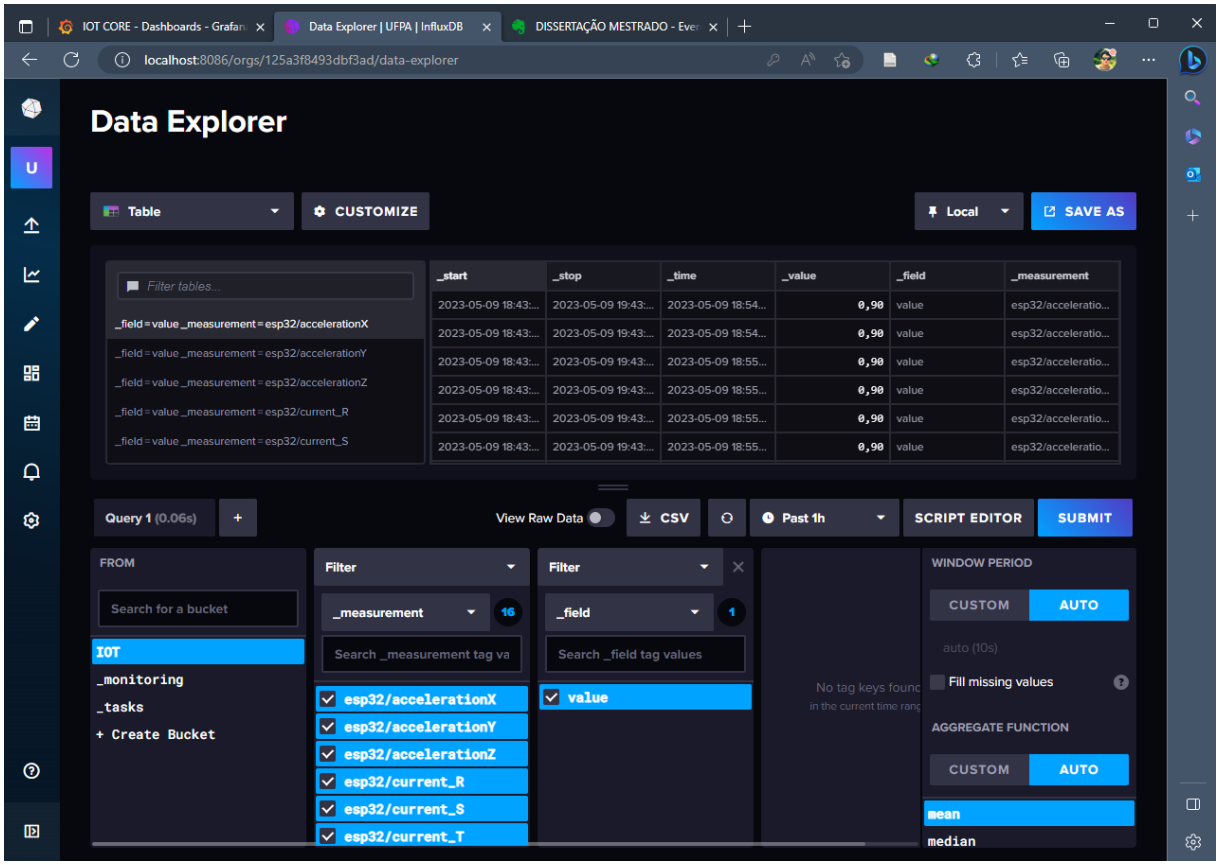


Fonte: Autor (2023)

4.6.2 InfluxDB

De maneira similar a validação do armazenamento dos dados no SQL Server, no InfluxDB ocorre de tal maneira a essa validação, ou seja, faz-se necessário autenticar-se no banco de dados e consultar os dados recebidos, para verificar se eles estão sendo armazenados corretamente. Para isso, inseriu-se a URL no browser "*http://localhost:8086*" para acessar o InfluxDB, em seguida inseriu-se os dados de login e senha. Na aba "*Data Explorer*" ao clicar em cima do bucket criado, chamado "*IOT*", pode-se observar todas as variáveis transmitidas pelo DAQ-IOT e armazenadas no InfluxDB em tempo real através do *script BRIDGE.py*. A Figura 55 ilustra a consulta dos dados no InfluxDB para validação do armazenamento nesse banco.

Figura 55 - Dados armazenados no banco de dados InfluxDB

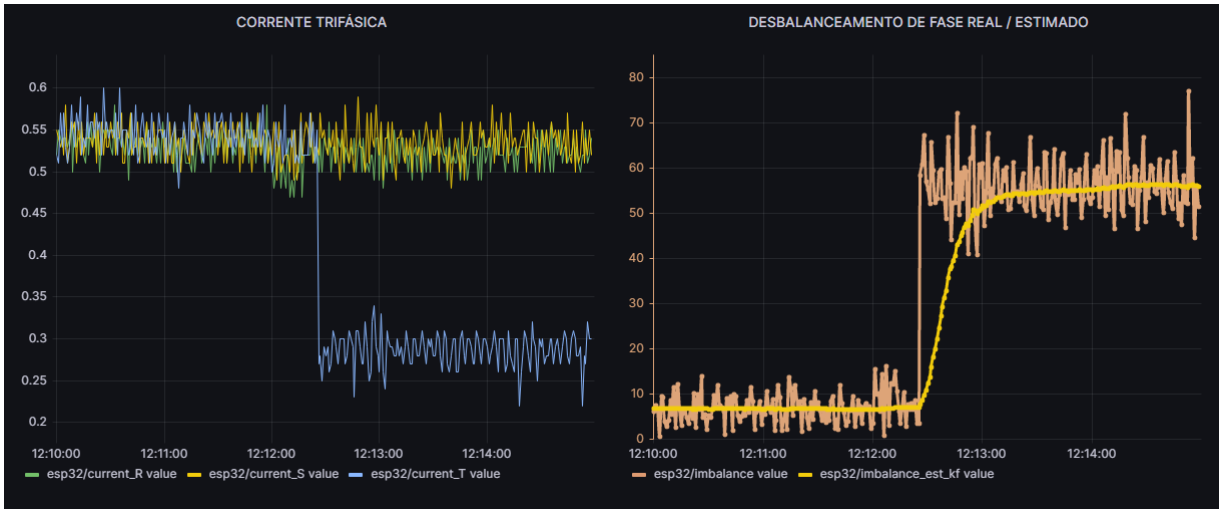


Fonte: Autor (2023)

4.7 Desbalanceamento de fases

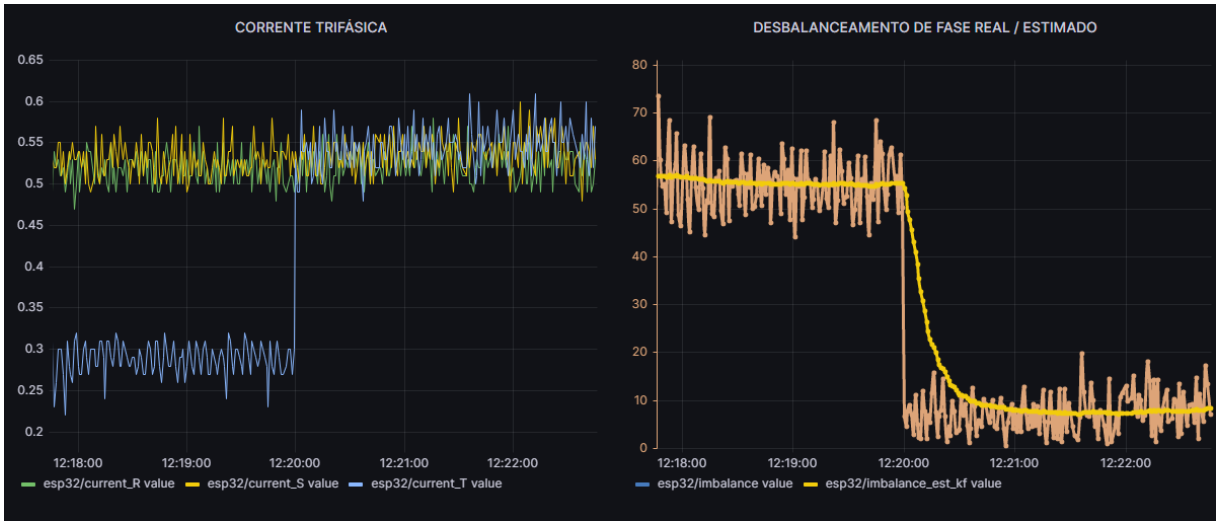
O algoritmo de desbalanceamento de fases mede a porcentagem de desvio entre fases e é um parâmetro fundamental para monitorar, pois representa riscos ao ativo, comprometendo o bom funcionamento do mesmo. Para a validação do algoritmo, a priori retirou-se o sensor de corrente da fase *T*, e avaliou-se o comportamento, conforme a Figura 56. Posteriormente, retornou-se com o SCT-013 para medição da fase *T*, conforme a Figura 57.

Figura 56 - Desbalanceamento de fase sem medição da fase *T*



Fonte: Autor (2023)

Figura 57 - Desbalanceamento de fase com retorno da medição da fase *T*



Fonte: Autor (2023)

Conforme observa-se na Figura 56, ao remover o sensor da fase *T*, o desbalanceamento de fase passa a ser superior a 40%, e ao retornar à medição da fase *T*, como ilustra a Figura 57, o desbalanceamento passa a ser menor que 20%. Além disso, pode-se observar as curvas de corrente trifásicas para fins de comparação entre as curvas.

Outro aspecto a ser observado, trata-se das curvas de desbalanceamento provenientes dos cálculos realizados de desvio entre fases, e com estimação através do filtro de Kalman. Nota-se que com a estimação de Kalman, as curvas apresentam um resultado mais próximo do real. Devido aos ruídos presentes na aquisição dos sinais de corrente, consequentemente isso afeta também os cálculos de desbalanceamento de fases, e por isso atotar uma solução como filtro de Kalman torna-se tão importante.

Do ponto de vista de intertravamento baseado no sinal de desbalanceamento de fase, ou seja, parar o funcionamento de uma planta em função de *threshold* de porcentagem de desbalanceamento configurado, um sistema ruidoso representa um grande problema na operação, pois constantemente pode-se intertravar os atuadores, no caso, o motor elétrico. Isso torna-se inseguro e ineficiente do ponto de vista operacional. Por isso, adotar o filtro de Kalman torna-se uma solução mais robusta.

4.8 Power BI

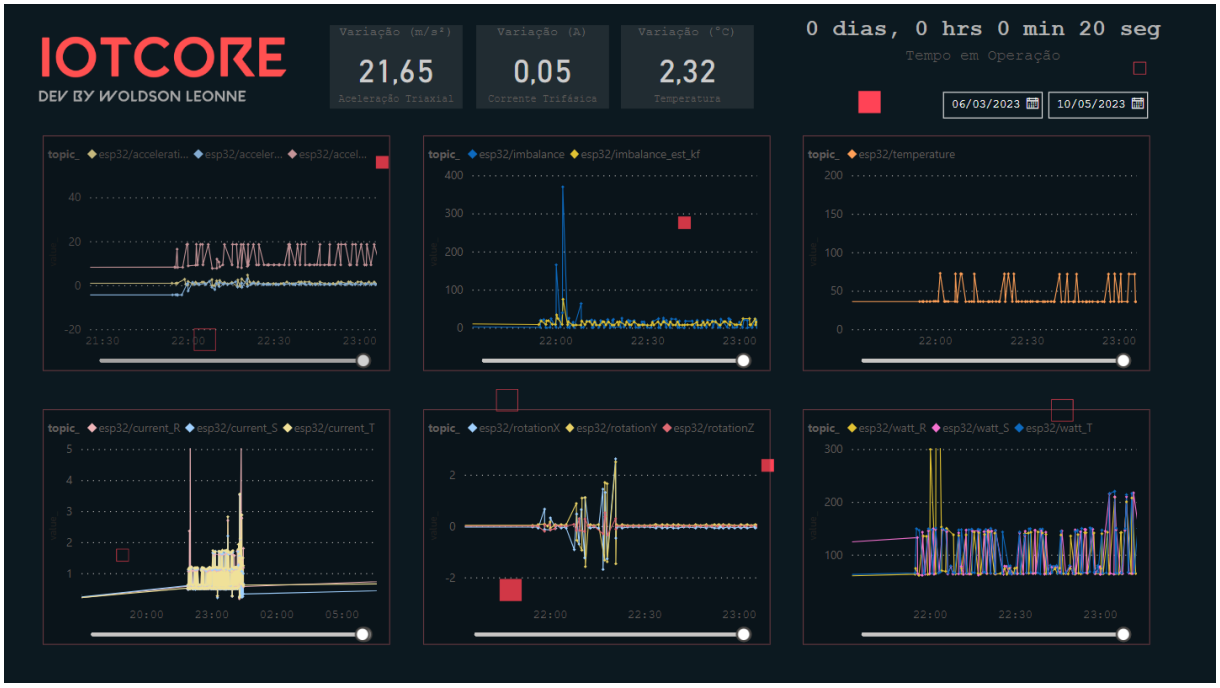
A supervisão dos dados no Power BI deve-se a uma conexão ao banco de dados SQL Server, e através de consultas SQL exibem os dados coletados em tempo real. O relatório no Power BI foi desenvolvido através do Power BI Desktop, o qual oferece várias ferramentas modernas para visualização de dados.

Apesar das consultas em SQL poderem ser realizadas em tempo real, a atualização dessa consulta não ocorre em tempo real, ou seja, os novos dados não são exibidos nos gráficos de forma automática, faz-se necessário sempre atualizar o relatório. Existem maneiras de configurar para a taxa de atualização ser de até 5 em 5 minutos, por exemplo, mas não se configurou esse método na aplicação em questão.

Além disso, notou-se que, para consultas muito extensas, ou seja, solicitar ao banco um grande volume de dados, o tempo de resposta era maior. Esse tempo de resposta para disponibilização dos dados é inerente ao tipo de banco de dados. Mas

isso não implicou em uma lentidão ou outro efeito negativo no relatório. A navegação e exibição de outros elementos como rótulo de dados iterativo não foi comprometido. A Figura 58 ilustra o relatório desenvolvido no Power BI para a visualização dos dados do sistema em questão.

Figura 58 - Supervisão dos dados com Power BI



Fonte: Autor (2023)

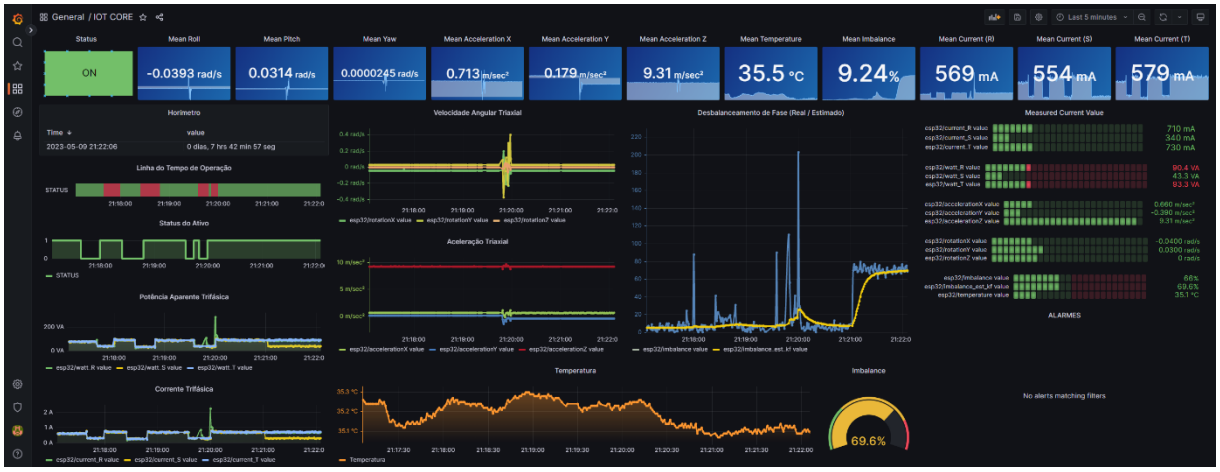
O painel no Power BI possui seis gráficos, nos quais exibem aceleração triaxial, desbalanceamento de fase normal e com filtro de Kalman, temperatura, corrente trifásica, velocidade angular trifásica, e potência trifásica. O painel também possui três cartões superiores, que mostram a variação de aceleração triaxial, variação de corrente nas três fases e variação da temperatura. Esses dados exibidos são controlados por um filtro de datas, o qual pode ser ajustado conforme a necessidade do usuário. Além disso, pode-se visualizar o tempo em operação do ativo monitorado, que exibe esse dado em detalhe no formato de dias, horas, minutos e segundos.

Com os dados disponibilizados no Power BI, pode-se facilmente realizar customizações e modos como os dados são exibidos. Pode-se, por exemplo, elaborar um *design* com elementos gráficos, condicionantes e até mesmo executar *scripts* em Python para modelar os dados dentro do Power BI.

4.9 Grafana

Diferentemente do Power BI, os dados no Grafana consomem os dados armazenados no banco de dados InfluxDB. Além disso, outro aspecto importante trata-se da atualização automática dos dados, no Grafana essa atualização automática já possui facilmente esse recurso, pois o intuito da ferramenta é visualizar dados em tempo real. O painel no Grafana pode ser visualizado na Figura 59.

Figura 59 - Supervisão dos dados com Grafana



Fonte: Autor (2023)

No painel de supervisão realizado no Grafana, pode-se observar várias semelhanças com o relatório no Power BI, porém com a vantagem de exibir os dados em tempo real, com as amostras atualizadas a cada segundo. O painel no Grafana apresenta os mesmos gráficos disponíveis no Power BI, permitindo visualizar métricas como aceleração triaxial, desbalanceamento de fase com filtro de Kalman, temperatura, corrente trifásica, velocidade angular trifásica e potência trifásica.

Além disso, o Grafana oferece recursos adicionais para aprimorar a visualização e análise dos dados. Por exemplo, o painel no Grafana possui gráficos de tempo em operação, que destacam o tempo em que o sistema está ligado e desligado. O gráfico marca visualmente em vermelho os períodos em que o sistema esteve desligado e em verde os períodos em que esteve ligado. Ao passar o mouse sobre um item no gráfico, é exibido um *tooltip* que registra o intervalo de tempo correspondente. Isso permite uma análise mais detalhada dos períodos de operação e desligamento do sistema.

Outro elemento presente no painel é o status atual do motor elétrico. Ele indica se o motor está ligado ou desligado em tempo real, fornecendo uma visão instantânea do estado do sistema.

Os cartões são utilizados para exibir informações resumidas e métricas calculadas. No caso do Grafana, os cartões mostram a média das principais variáveis do sistema, permitindo uma rápida compreensão do estado geral do sistema. Além disso, o painel também exibe as informações em tempo real do último valor registrado para cada variável, garantindo que os usuários tenham acesso aos dados mais recentes.

Em resumo, o painel de supervisão no Grafana oferece a vantagem de exibir os dados em tempo real, além de incluir recursos adicionais, como gráficos de tempo em operação, status atual do motor elétrico, cartões com médias das principais variáveis e exibição dos últimos valores registrados em tempo real. Esses recursos facilitam a monitorização contínua do sistema e permitem uma análise mais detalhada e precisa das métricas relevantes.

4.10 Genesis64

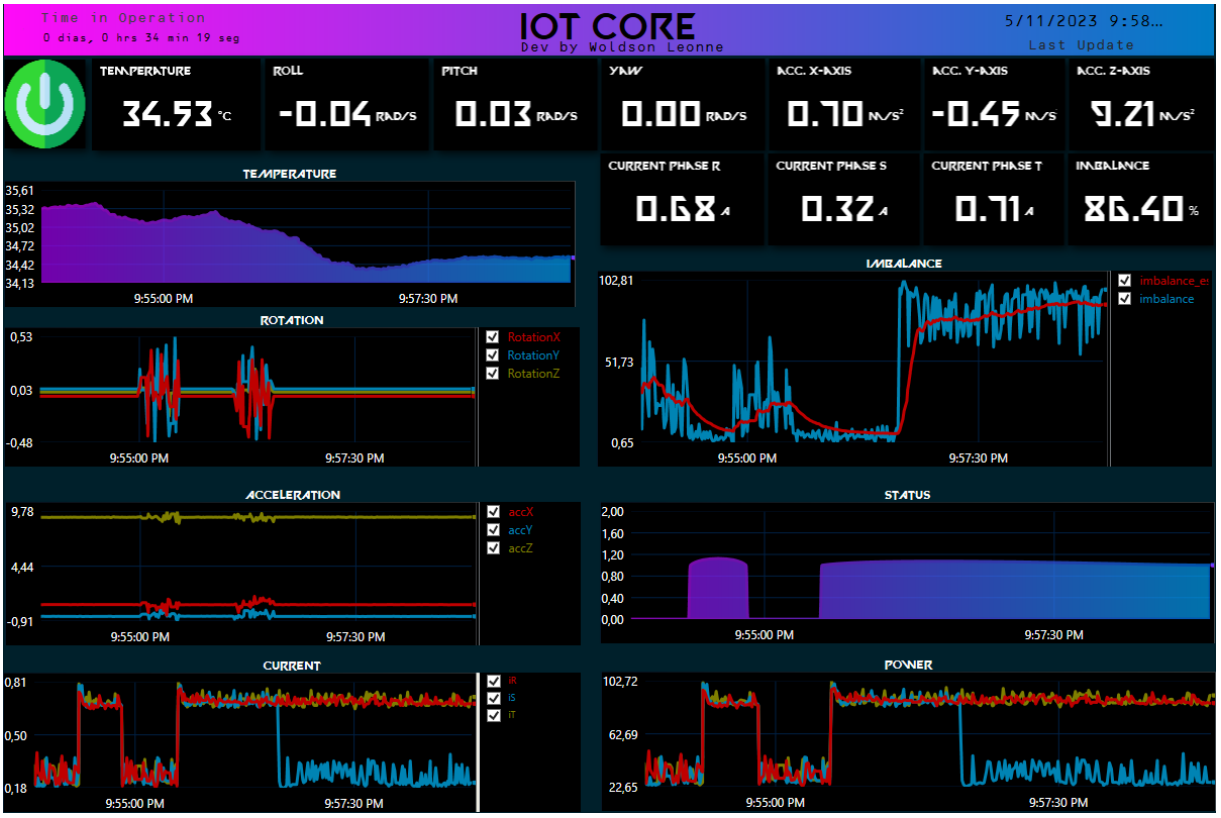
No painel de supervisão realizado no supervisório Genesis64, da ICONICS, encontramos algumas semelhanças com o relatório no Power BI e o painel no Grafana. Assim como nos outros sistemas, o painel no Genesis64 apresenta os mesmos gráficos em tempo real. No entanto, existem algumas diferenças notáveis em relação à fonte de dados e alguns recursos específicos.

No Genesis64, os dados exibidos nos gráficos são provenientes do servidor OPC UA, ao invés de serem obtidos de um banco de dados como o InfluxDB ou o SQL Server. O uso do servidor OPC UA permite que o supervisório se conecte diretamente aos dispositivos e sistemas de controle para obter os dados em tempo real.

Assim como no Grafana, o painel no Genesis64 inclui gráficos de tempo em operação. No entanto, em vez de utilizar cores específicas, ele usa um gráfico de pulsos, onde um nível alto representa o estado "*ligado*" e um nível baixo representa o estado "*desligado*". Esse gráfico de pulsos fornece uma representação visual do estado do sistema ao longo do tempo.

Da mesma forma que no Grafana, o Genesis64 também exibe o status atual do motor elétrico, indicando se está ligado ou desligado em tempo real. Isso permite uma rápida visualização do estado do motor. A Figura 60 ilustra o supervisório no Genesis64.

Figura 60 - Supervisão dos dados com Genesis64



Fonte: Autor (2023)

O painel também utiliza cartões para exibir informações em tempo real do último valor registrado para cada variável, fornecendo acesso rápido aos dados mais recentes.

No entanto, é importante mencionar que uma limitação do Genesis64 é a visualização de dados sem recuperação. Isso significa que, ao reiniciar a aplicação, os dados anteriormente exibidos são perdidos. No entanto, o Genesis64 tem a capacidade de consumir dados provenientes do banco de dados SQL Server, embora essa não seja uma prática comum nesse tipo de supervisório. O Genesis64 normalmente utiliza dados provenientes do servidor OPC UA e oferece suporte a outros protocolos de comunicação, como MQTT, para obter dados em tempo real.

4.11 Aplicação web com Flask

No painel de supervisão realizado na aplicação web desenvolvida em Flask, encontra-se algumas semelhanças com o relatório no Power BI. Assim como no Power BI, o painel na aplicação web utiliza os mesmos gráficos para visualização dos dados.

Além dos gráficos, o painel na aplicação web também inclui o item que exibe o *status* atual do motor elétrico, indicando se está ligado ou desligado, assim como no Grafana. Também são exibidas informações em cartões, como a última atualização, a média da temperatura, a média do desbalanceamento de fase estimada e o horímetro.

No entanto, uma limitação desse sistema é a atualização automática dos dados em tempo real. Para atualizar os dados, é necessário recarregar a página da web. Isso significa que os dados não são atualizados de forma automática, como no Grafana. O usuário precisa interagir com a página para obter os dados mais recentes.

Uma das características diferenciadoras dessa aplicação em relação aos outros meios de supervisão é a exibição de gráficos no domínio da frequência por meio da transformada de Fourier, para as variáveis de vibração triaxial. Isso permite uma análise mais avançada dos dados de vibração e fornece insights adicionais sobre o comportamento do sistema.

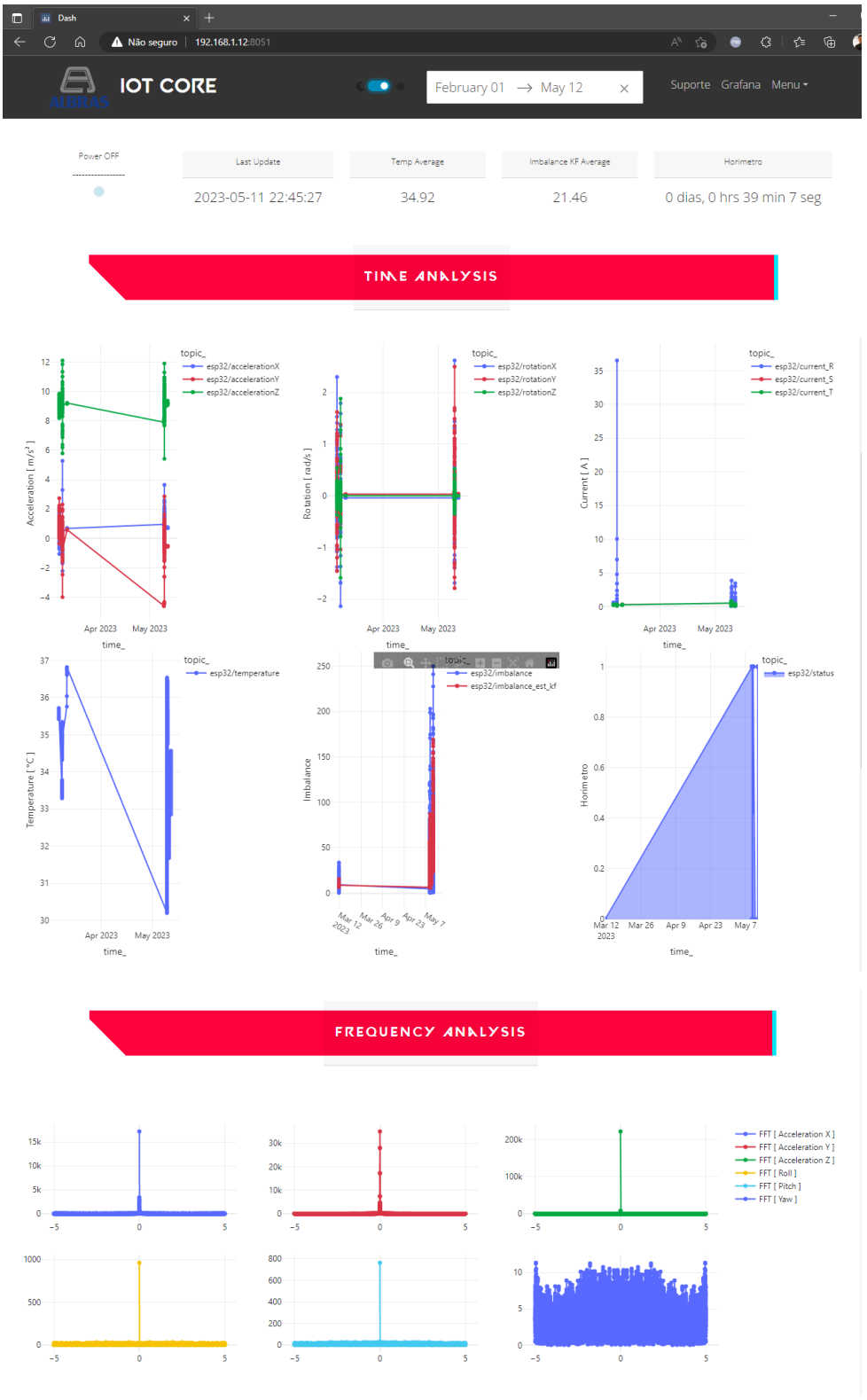
Outro diferencial é a disponibilidade de várias opções de temas para o usuário escolher. Isso garante uma melhor experiência do usuário, permitindo que ele selecione um tema que se adapte às suas preferências e torne a interface mais amigável.

Além disso, a aplicação web desenvolvida em Flask também oferece recursos adicionais, como a capacidade de realizar o *download* dos gráficos e a opção de filtrar as informações por meio de um filtro de datas. Esse filtro permite limitar a consulta dos dados no banco SQL Server com base em uma faixa de datas específica, permitindo uma análise mais precisa dos dados.

Em resumo, o painel de supervisão na aplicação web desenvolvida em Flask apresenta gráficos similares aos do Power BI, exibe os dados em tempo real provenientes do banco de dados SQL Server e oferece recursos como exibição do *status* do motor, cartões com informações em tempo real, exibição de gráficos no domínio da frequência, opções de temas personalizáveis, *download* de gráficos e filtragem de dados por meio de um filtro de datas. No entanto, a atualização automática

dos dados em tempo real requer a recarga da página web. A Figura 61 ilustra uma visão geral da aplicação web desenvolvida com Flask.

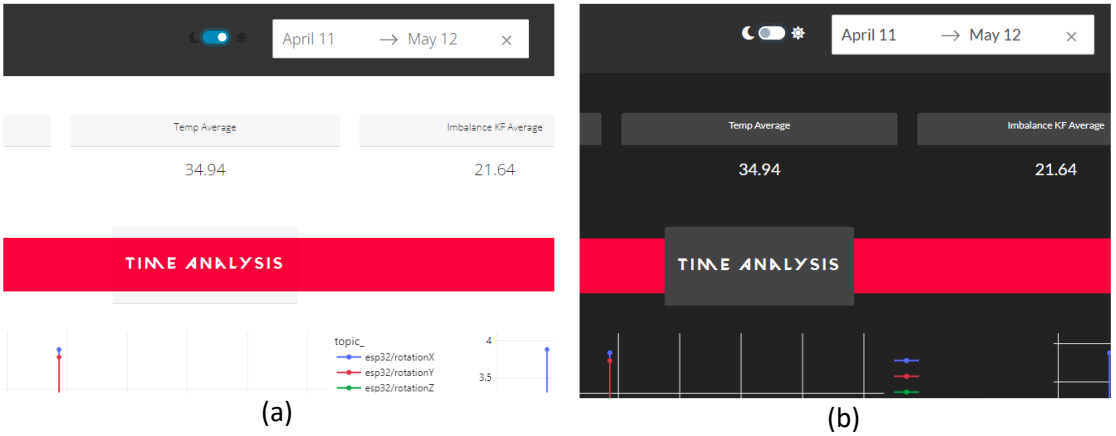
Figura 61 - Supervisão dos dados com aplicação web em Flask



Fonte: Autor (2023)

Para modificar o tema rapidamente entre um tema claro ou escuro, pode-se selecionar o botão de tema, conforme ilustrado na Figura 62, na qual (a) representa a escolha por um tema claro, e (b) representa a escolha por um tema escuro.

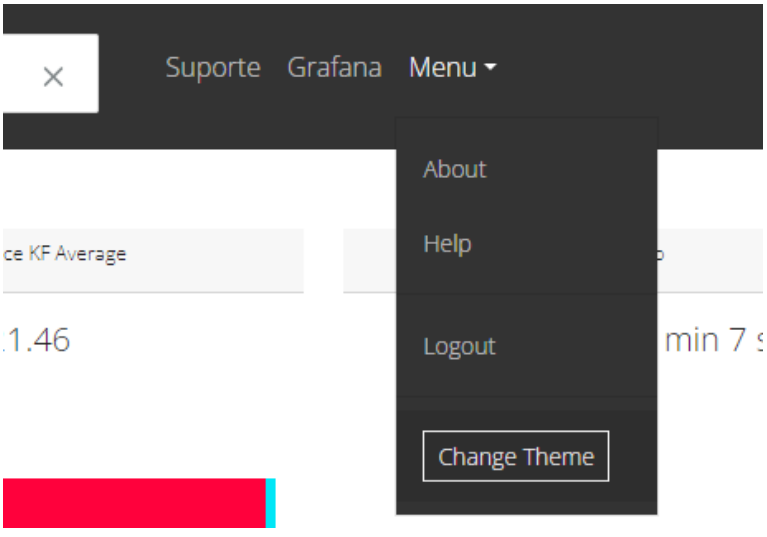
Figura 62 - Alternância rápida de temas claro/escuro



Fonte: Autor (2023)

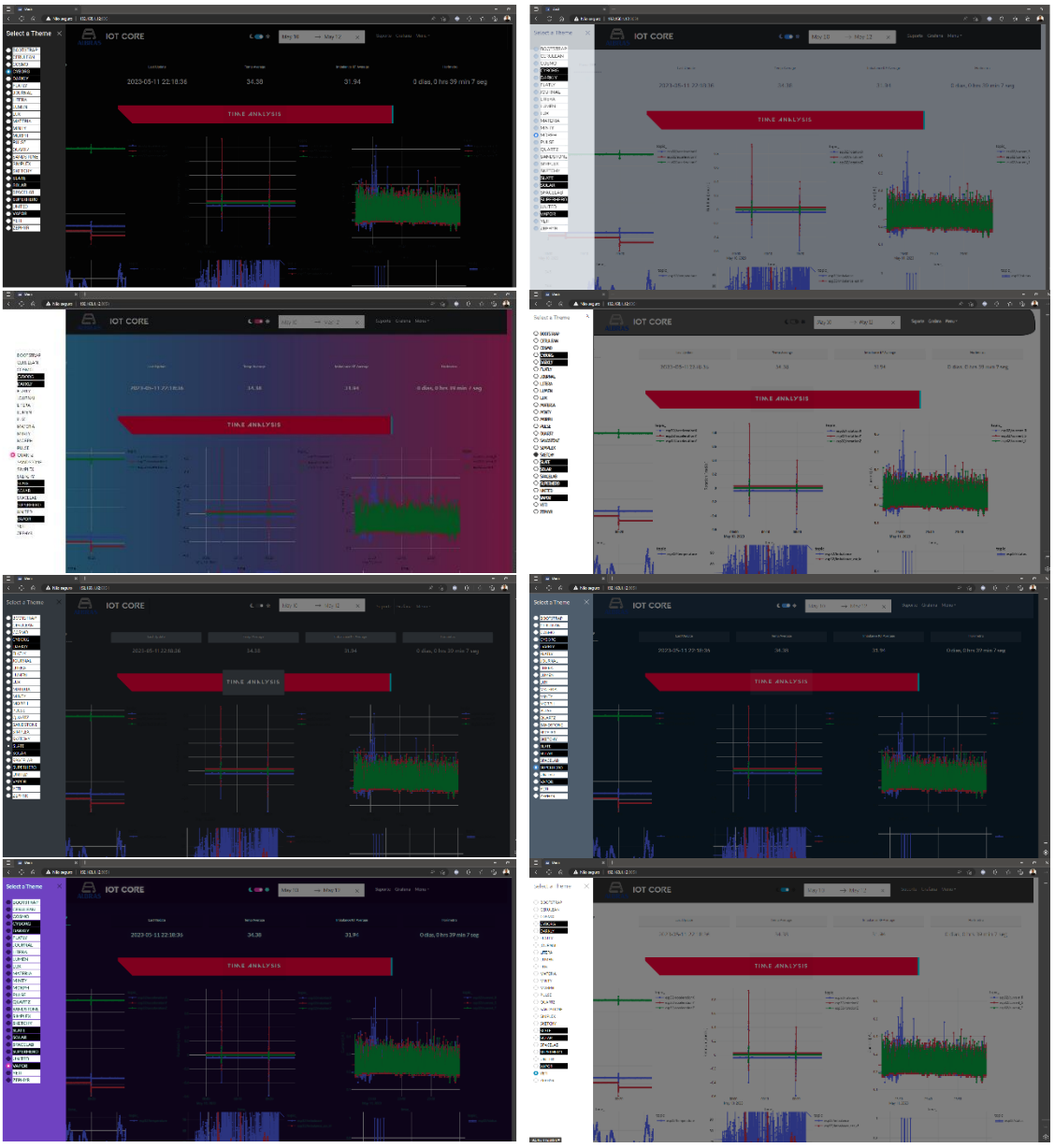
Além das suas opções de temas através do botão de seleção da Figura 69, pode-se escolher várias outras opções de temas. Ao clicar em "*Menu*" e escolher a opção "*Change Theme*", conforme a Figura 63, uma barra de navegação lateral surge, com várias opções de temas disponíveis. A Figura 64 ilustra oito das vinte e seis opções distintas de temas.

Figura 63 - Opções disponíveis em menu



Fonte: Autor (2023)

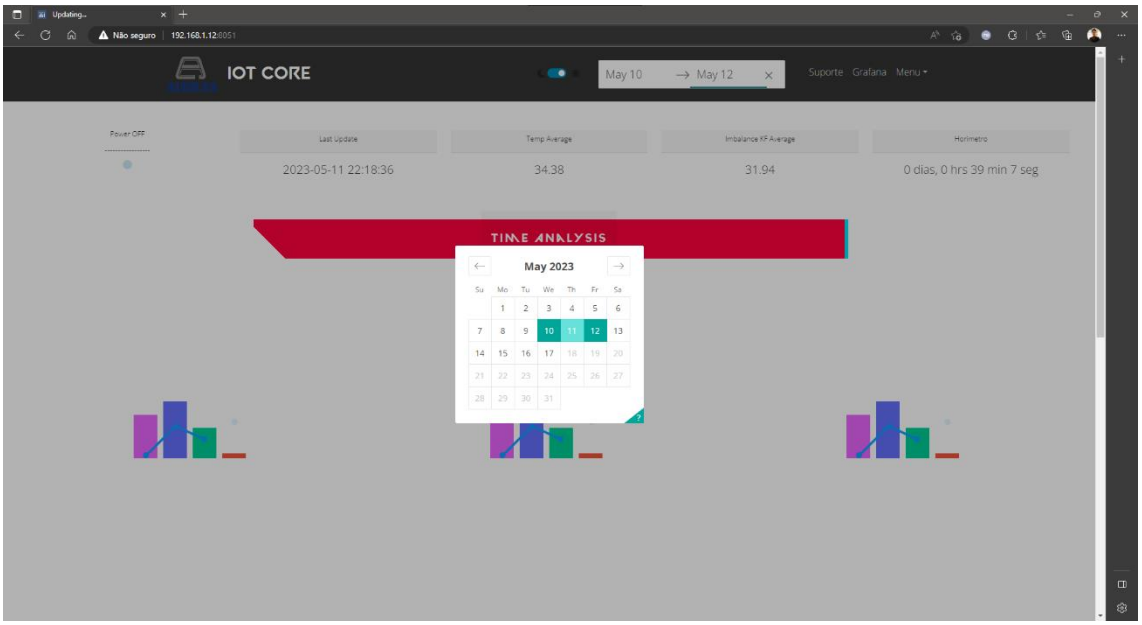
Figura 64 - Seleção de temas da aplicação web



Fonte: Autor (2023)

Além dos recursos de temas, a aplicação possui um *layout* moderno, como pode ser observado ao selecionar o range de datas para o intervalo de visualização dos dados. A Figura 65 ilustra o *pop-up* da seleção de datas, e além disso, é possível observar que a área de gráficos é modificada ao carregar os gráficos, e isso deve-se a uma configuração de animação enquanto os dados não estão disponíveis na consulta.

Figura 65 - *Pop-up* seleção de data e animação de carregamento



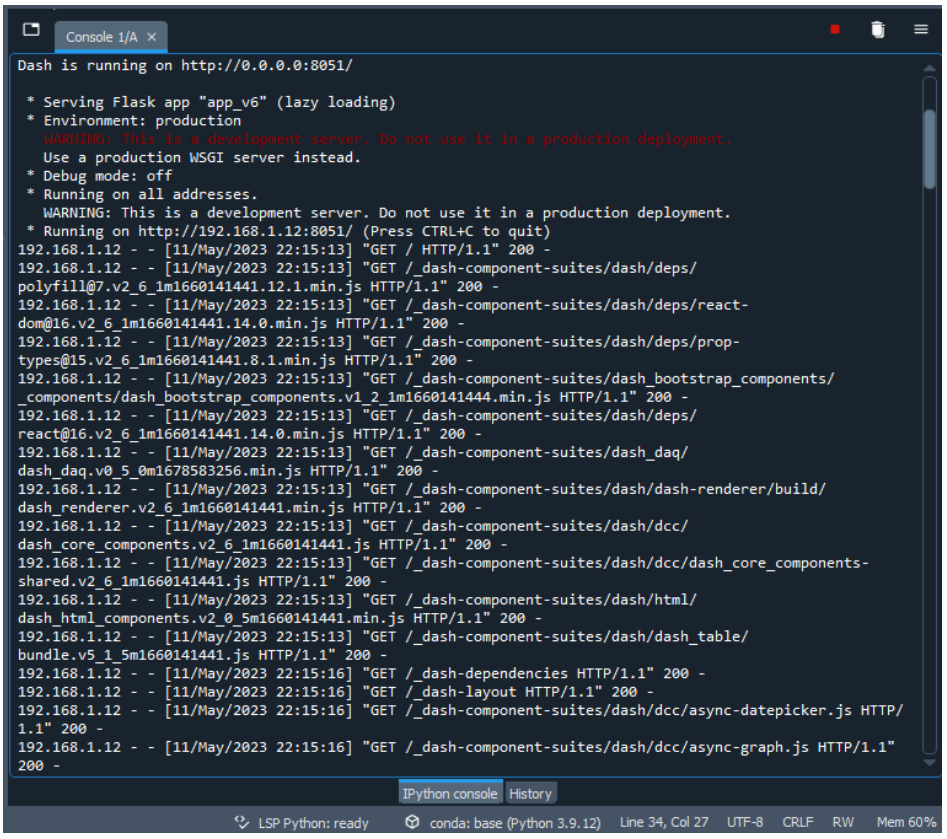
Fonte: Autor (2023)

No painel web desenvolvido em Flask, foi observado um excelente comportamento da página ao utilizar Bootstrap. Com o uso do Bootstrap, a página do painel web se adaptou de forma responsiva ao ser redimensionada. Isso significa que, ao alterar o tamanho da janela do navegador ou visualizar a página em dispositivos móveis, como smartphones ou tablets, o *layout* e os elementos da página se ajustaram automaticamente para proporcionar uma experiência de visualização otimizada.

Ao redimensionar a página, os elementos do painel, como gráficos e cartões, reorganizam-se e redimensionam-se para se adequar ao espaço disponível, mantendo a legibilidade e a usabilidade da página. Isso significa que os usuários podem acessar o painel em diferentes dispositivos e ter uma experiência consistente e confortável, independentemente do tamanho da tela. A capacidade de resposta do Bootstrap garante que o conteúdo do painel seja facilmente acessível e legível, melhorando a usabilidade e a experiência do usuário.

Além dos recursos modernos de visualização de dados, a segurança e análise do comportamento dos usuários ao interagir com a aplicação é um fator importante. Nesse sentido, todo dispositivo que interage com a aplicação tem o endereço *IP* monitorado, bem como as ações realizadas na aplicação. Ou seja, um *log* com as ações e quem realizou as ações (Endereço *IP*) são descritos em um console, o qual pode ser observado na Figura 66.

Figura 66 - Log de monitoramento de interação com a aplicação



Fonte: Autor (2023)

4.12 Aplicação mobile

Na aplicação mobile desenvolvida, o painel de supervisão possui características semelhantes aos relatórios do Power BI, Grafana, Genesis64 e aplicação web. Ele apresenta os mesmos gráficos para visualização dos dados em tempo real.

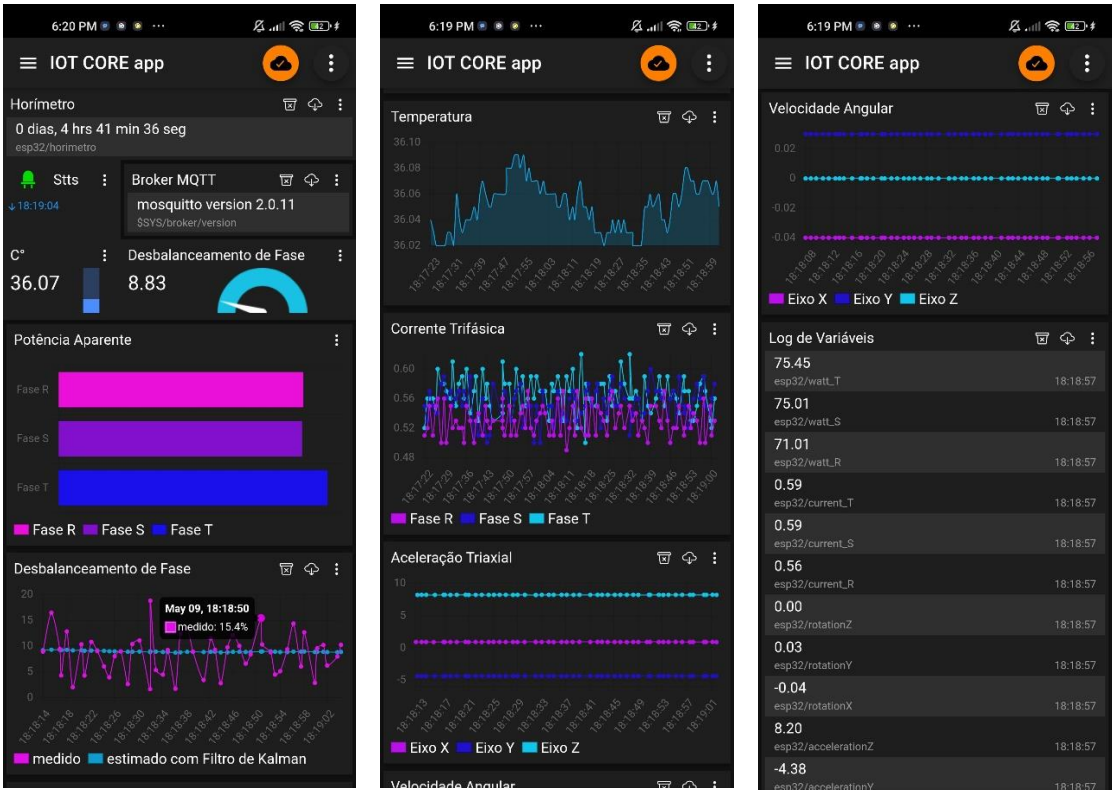
Assim como nos outros painéis, a aplicação mobile exibe o *status* atual do motor elétrico, o horímetro e um log com todas as informações transmitidas em tempo real. Isso permite que os usuários acompanhem o funcionamento do motor, registrem o tempo de operação e visualizem um histórico das informações recebidas.

Uma diferença importante da aplicação mobile em relação às outras é o dispositivo em que ela funciona. Essa aplicação é projetada exclusivamente para dispositivos móveis, como smartphones e tablets. Portanto, os usuários podem acessar e visualizar o painel de supervisão em qualquer lugar, usando dispositivos móveis.

Outra diferença é que os dados são consumidos diretamente do *broker* MQTT, eliminando a necessidade do servidor de aplicação que executa o *script BRIDGE.py*. Isso permite uma comunicação direta e em tempo real entre a aplicação *mobile* e o *broker* MQTT, facilitando a obtenção dos dados atualizados. Além disso, todas as informações do broker estão disponíveis, necessitando apenas do tópico para exibir os dados de uma variável. A exemplo, utilizou-se o tópico nativo do Mosquitto chamado "*$SYS*" para exibir a versão do *broker* MQTT.

No entanto, uma desvantagem dessa abordagem é que a aplicação mobile não possui a capacidade de armazenar e recuperar dados. Isso significa que, ao reiniciar a aplicação, todas as informações são perdidas, assim como no caso do Genesis64. Portanto, é importante garantir que os dados sejam monitorados continuamente e que os registros necessários sejam realizados em outros sistemas de armazenamento, se necessário. A Figura 67 ilustra a aplicação mobile com os painéis desenvolvidos para supervisionar o sistema.

Figura 67 - Supervisão dos dados através de aplicação mobile



Fonte: Autor (2023)

4.13 Alarmes

Para a configuração de alarmes, utilizou-se apenas uma variável para realizar a validação do algoritmo. A variável de desbalanceamento de fase é uma das mais importantes quando se trata de gestão de ativos ao realizar monitoramento de corrente trifásica. Nesse sentido, baseado na lógica de gatilho para alarme, ao detectar um desbalanceamento acima do limite configurado, o algoritmo em Python, deve enviar um e-mail de alerta e salvar esse evento no banco de dados SQL Server, em uma tabela chamada "*ALARMS*".

Para simular a geração de alarmes, retirou-se o sensor de corrente de uma das fases, para gerar um desbalanceamento de fase acima de 30%, o qual foi o valor definido como limite. Após isso, verificou-se a caixa de entrada de e-mail configurado para receber os alertas, a Figura 68 ilustra o resultado do envio dos alarmes recebidos por e-mail, conforme o esperado.

Figura 68 - Eventos de alarmes recebidos via e-mail

Fonte: Autor (2023)

Nota-se que, o intervalo entre cada e-mail recebido, é de 30 minutos. Esse fato deve-se justamente ao tempo em que os dados são armazenados em um *buffer* e posteriormente verificado se existem valores acima do limite definido.

A implementação de alarmes em sistemas de automação é uma prática comum para identificar e notificar os operadores sobre eventos indesejados ou condições anormais. No entanto, para garantir uma resposta eficaz e uma resolução rápida dos problemas, é essencial que os alarmes não se limitem a relatar apenas o evento ocorrido, mas também forneçam informações adicionais sobre como solucioná-lo.

Ao receber um alarme por e-mail, os operadores precisam ser orientados sobre quais medidas tomar para lidar com a situação. Isso pode incluir instruções

específicas, procedimentos a serem seguidos ou até mesmo um checklist de ações recomendadas. Essas diretrizes ajudam os operadores a tomar as medidas corretas de forma rápida e eficiente, minimizando o tempo de resposta e reduzindo o impacto do evento.

Além disso, é importante que os alarmes forneçam informações sobre as possíveis causas do problema. Isso permite que os operadores identifiquem a causa raiz e adotem métodos adequados para resolvê-la. Os alarmes podem indicar possíveis fontes de falha, sistemas ou componentes envolvidos, ou até mesmo fornecer dicas sobre etapas de diagnóstico a serem seguidas.

Essas informações são valiosas para acelerar o processo de resolução de problemas e evitar retrabalho desnecessário. Nesse sentido, os alarmes possuem informações sobre o evento, como data, hora e valor lido, bem como orientações para investigar as causas problema, conforme ilustra a Figura 69, o e-mail recebido de alerta de desbalanceamento de fases.

Figura 69 - E-mail de alarme com orientações

Fonte: Autor (2023)

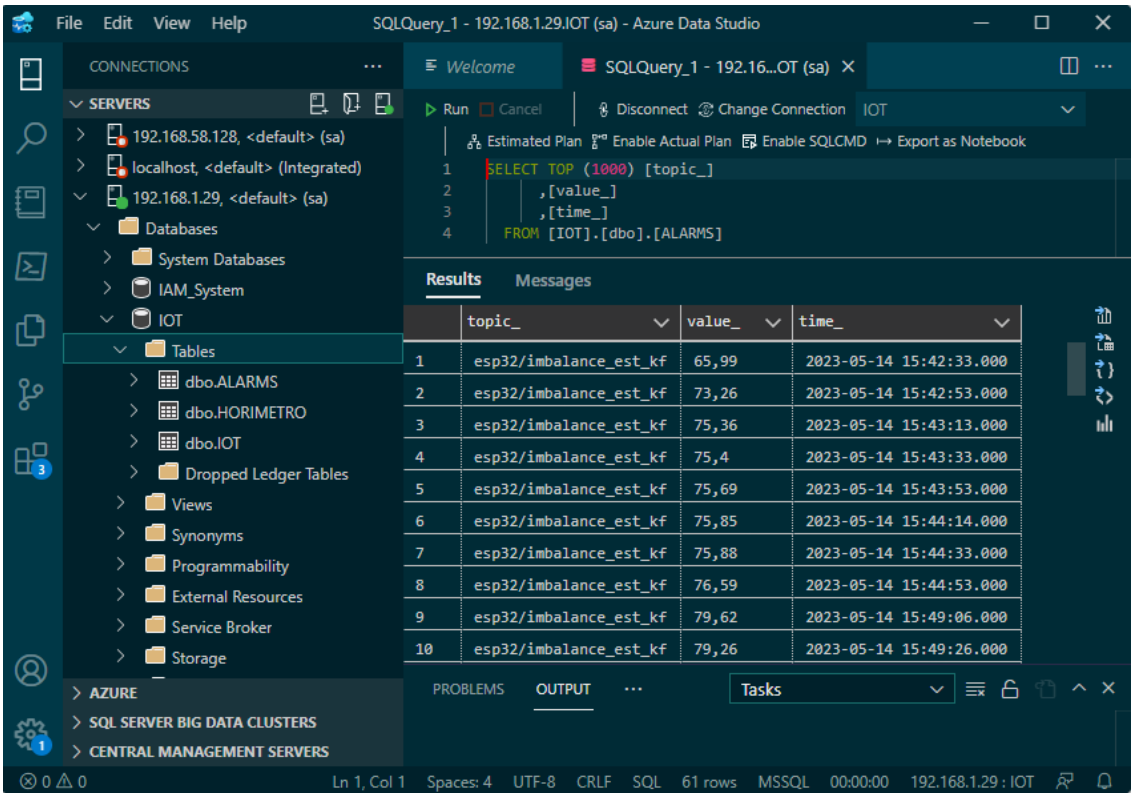
A validação do armazenamento dos eventos de alerta é de extrema importância para garantir que essas informações valiosas sejam registradas corretamente. Esses eventos não apenas são relevantes para o controle de qualidade em tempo real, mas também fornecem uma base sólida para análises futuras e a implementação de algoritmos de machine learning.

Ao armazenar os eventos de alerta, cria-se um histórico valioso que pode ser usado para análises retrospectivas e identificação de padrões. Isso permite avaliar o desempenho do sistema ao longo do tempo, identificar tendências, realizar comparações entre diferentes eventos e tomar decisões informadas para melhorar a eficiência e a confiabilidade das operações.

Além disso, o armazenamento adequado dos eventos de alerta permite a implementação de algoritmos de *machine learning* para análise preditiva. Ao correlacionar os eventos de alerta com as leituras das demais variáveis do sistema, é possível identificar padrões ocultos, prever possíveis falhas e tomar medidas proativas para evitar interrupções não planejadas. Os algoritmos de *machine learning* podem aprender com os dados históricos, aprimorar continuamente modelos e fornecer insights valiosos para o gerenciamento e a otimização do sistema.

Desse modo, para validar o armazenamento dos alarmes, realizou-se uma consulta no banco de dados SQL Server, na tabela "*ALARMS*", e pode-se verificar que os dados foram armazenados corretamente, conforme ilustra a Figura 70.

Figura 70 - Validação do armazenamento dos alarmes



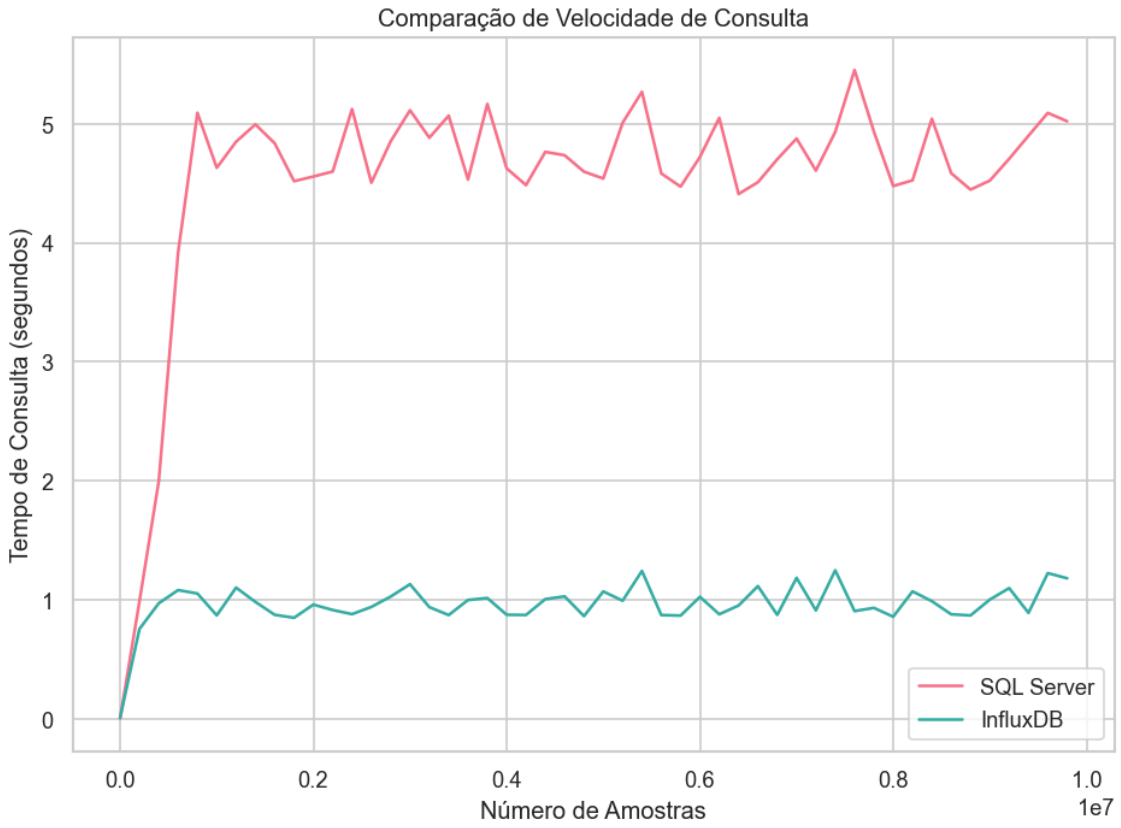
Fonte: Autor (2023)

4.14 Comparação de velocidade de consulta nos bancos

Para comparar a velocidade de consulta dos dados em cada banco em específico, utilizou-se um algoritmo desenvolvido em python, para poder realizar consulta de dados com o número de amostras variado. O algoritmo calcula o tempo de cada consulta, ou seja, o tempo para consulta no banco SQL Server, e o tempo para consulta dos dados no InfluxDB.

O algoritmo realiza várias consultas, em um range começando em dez amostras até dez milhões de amostras, com um intervalo de duzentas mil amostras. Após coletar as informações de tempo para cada consulta, utilizou-se a biblioteca matplotlib para exibir as informações coletadas, conforme observa-se na Figura 71.

Figura 71 - Comparação de velocidade de consulta entre os bancos de dados



Fonte: Autor (2023)

Conforme pode-se observar, para uma quantidade pequena de amostras consultadas, o SQL Server e InfluxDB tem praticamente o mesmo desempenho. Todavia, à medida que o número de dados consultados aumenta, o SQL Server

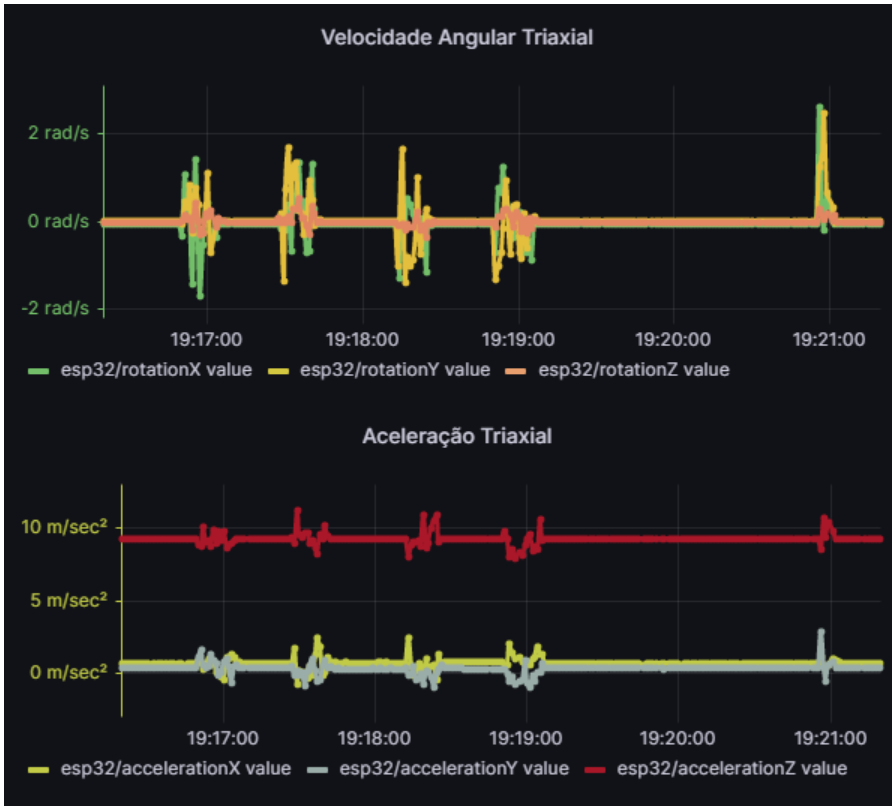
demora mais de quatro vezes o tempo de consulta em relação ao InfluxDB. Isso demonstra que, para esses tipos de dados, ou seja, dados de variáveis de processos IOT e *big data*, um banco de dados de séries temporais apresenta melhor desempenho do que um banco de dados relacional.

A estrutura de armazenamento e os algoritmos de indexação do InfluxDB são otimizados para manipular grandes volumes de dados de séries temporais de forma eficiente. Ele utiliza uma estrutura para acelerar a recuperação de dados com base no tempo. Embora o SQL Server possa lidar com dados de séries temporais, ele não possui as mesmas otimizações e recursos especializados do InfluxDB nessa área específica.

4.15 Validação do sensor de vibração

Para validar a aquisição de vibração, movimentou-se o sensor nos três eixos, em cinco momentos distintos. Ao movimentar-se, nos três eixos, nota-se a variação dos dados de aceleração e velocidade angular, conforme ilustrado na Figura 72.

Figura 72 - Simulação e coleta de dados de vibração



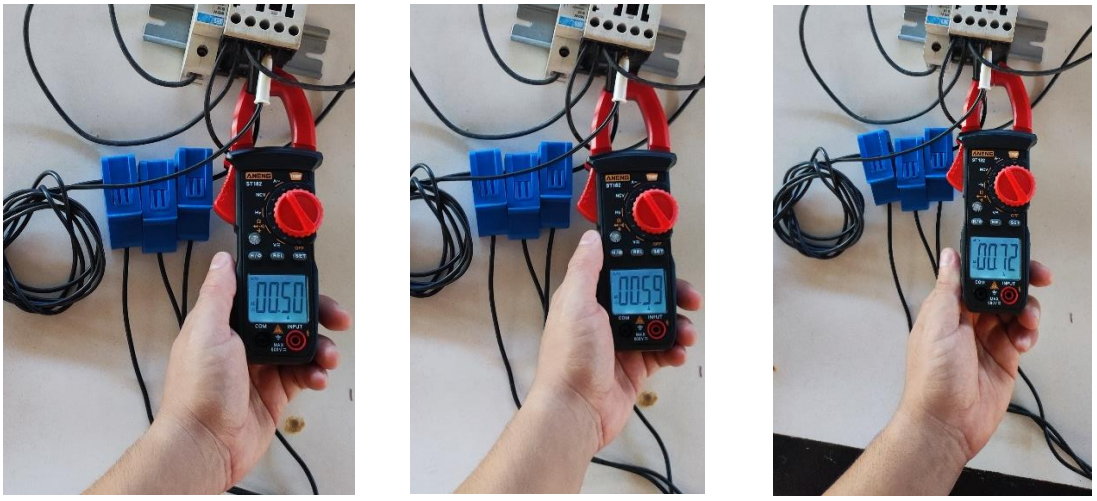
Fonte: Autor (2023)

Um aspecto importante nos resultados de simulação de vibração, trata-se do comportamento de aceleração no eixo z, correspondendo a um valor de *9,8 m/s²*. Esse valor deve-se a aceleração da gravidade, demonstrando que o sensor apresenta resultados condizentes com valores teóricos. Além disso, outras maneiras de validar a coleta, trata-se da aquisição de vibração com equipamentos certificados.

4.16 Validação do sensor de corrente

Para a validação do sensor de corrente, utilizou-se um alicate amperímetro com precisão de 2,5% para medição de corrente com três cargas diferentes, e ao mesmo verificou-se o valor lido através do sensor de corrente SCT-013. Desse modo, os três sensores de corrente SCT-013 e o alicate amperímetro foram posicionados envolta do mesmo cabo, o qual trata-se da fase da carga, conforme ilustrado na Figura 73.

Figura 73 - Medição de corrente com alicate amperímetro



(a) (b) (c)

Fonte: Autor (2023)

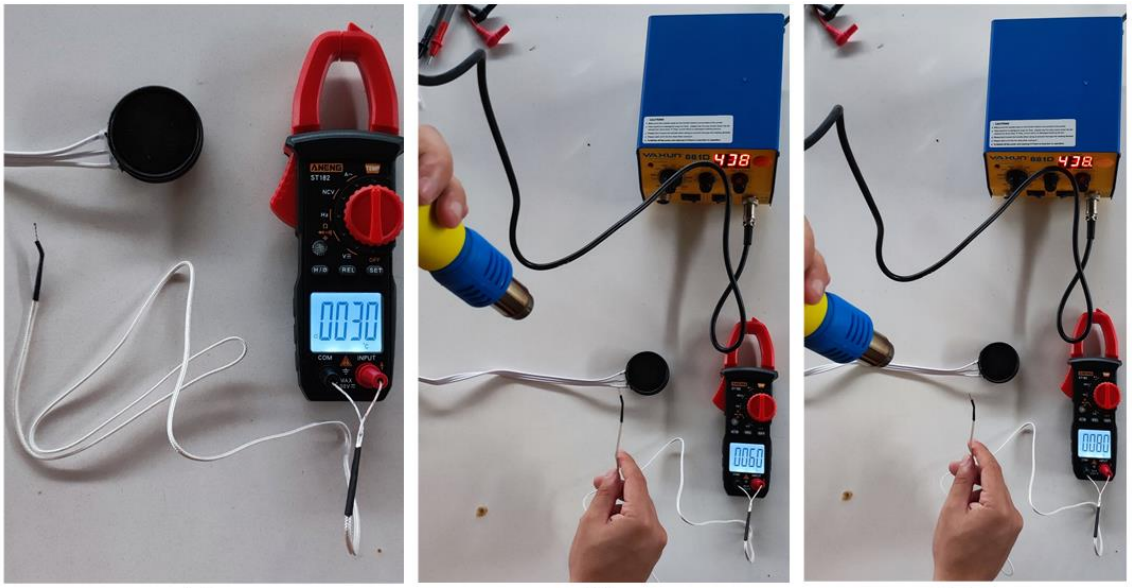
Para o primeiro caso, observado na Figura 73 (a), nota-se o valor mais baixo de corrente, sendo de 0,5A. Ao comparar esse valor com o lido pelo SCT-013, observou-se um valor de 0,53A, com certa variação proveniente de ruído. Ao aumentar a carga, a medição de corrente com o alicate amperímetro, ilustrado na Figura 73 (b), registou um valor de 0,59A, enquanto no SCT-013 o valor registrado foi de 0,60A. Por fim, ao aumentar pela última vez a carga, a corrente registrada pelo alicate amperímetro, visualizada na Figura 73 (c), foi de 0,72A, o qual foi o mesmo valor

registrado pelo SCT-013. Nota-se que, ao aumentar a carga, o valor de corrente aumenta simultaneamente com a exatidão de leitura do SCT-013. Para valores baixos de corrente, o valor lido oscila devido a componentes de ruído. Para o primeiro, segundo e terceiro caso na validação, os erros entre medições foram respectivamente de 0,03A, 0,01A e 0A.

4.17 Validação do sensor de temperatura

A validação dos dados do sensor de temperatura foi realizada de forma similar a validação do sensor de corrente. Para isso, utilizou-se um multímetro com a função de leitura de temperatura, com 2% de precisão. Para simular a variação de temperatura, utilizou-se um soprador de uma estação de solda, o qual soprou ar quente nos dois sensores a serem comparados. Simulou-se a variação de temperatura para três valores diferentes variando-se a distância de exposição do ar quente, conforme observa-se na Figura 74.

Figura 74 - Medição de temperatura com multímetro



Fonte: Autor (2023)

As simulações, conforme a Figura 74, foram realizadas para três temperaturas distintas, na Figura 74 (a) simulou-se para uma temperatura ambiente, sem a utilização do soprador, para esse caso o multímetro registrou um valor de 30ºC,

enquanto o sensor do DAQ-IOT registrou um valor de 31,5ºC. Para o segundo caso, ilustrado na Figura 74 (b), utilizou-se o soprador para simular um aumento de temperatura, ao registrar 60ºC no multímetro, o DAQ-IOT registrou um valor de 58,7ºC. Por fim, no último caso da simulação, observado na Figura 74 (c), o multímetro registrou um valor de 80ºC, e o sensor de temperatura do DAQ-IOT registrou um valor de 83ºC. Os erros entre as medições comparados para cada um dos três casos, foram respectivamente de 1,5ºC, 1,3ºC e 3,0ºC.

Além disso, observou-se que ao retirar a ação do soprador e retornar a temperatura ambiente, o DAQ-IOT demorou mais para retornar à temperatura ambiente, comparado ao sensor do multímetro. Esse fato deve-se ao case de metal no qual o sensor está submetido, pois notou-se que o material metálico permaneceu quente por um certo tempo até o gradiente de temperatura entrar em equilíbrio.

4.18 Gestão de backup

A gestão de backup é essencial para manter a confiabilidade do sistema e a integridade das configurações. Nesse sentido, foram realizados backups do sistema que hospeda o broker MQTT e do sistema que hospeda o servidor.

Para o backup do sistema que hospeda o broker MQTT, utilizou-se um cartão de memória de 16GB e realizou-se uma cópia completa do sistema. Esse processo foi realizado com o auxílio de um computador desktop auxiliar.

Para o *backup* do servidor, que contém banco de dados e aplicações, foi utilizada uma clonagem do sistema. Essa clonagem foi feita por meio de uma ferramenta específica do VMWare, que permitiu criar uma réplica idêntica do servidor.

Manter cópias atualizadas dos sistemas é de extrema importância para garantir a integridade dos dados e possibilitar uma rápida recuperação em caso de perda ou corrupção de informações. Essa prática é especialmente crucial em ambientes industriais, nos quais a automação é fundamental. Softwares como o Acronis são comumente utilizados nesses ambientes para gerenciar de forma automatizada todo o processo de *backup* dos sistemas.

Ao implementar uma gestão de *backup* eficiente, é possível minimizar os riscos de perda de dados e assegurar a continuidade das operações, oferecendo tranquilidade e segurança para o sistema como um todo.

5 DISCUSSÃO

Com base nos resultados obtidos, observa-se que a escolha da linguagem Python tornou possível o desenvolvimento do sistema. Ao desenvolver cada *script*, pôde-se constatar as várias vantagens dessa linguagem, como flexibilidade, legibilidade, entre outras. A facilidade de uso de bibliotecas, a simplicidade na elaboração dos códigos, o amplo suporte em fóruns de discussão e a disponibilidade de uma rica documentação foram fundamentais para solucionar rapidamente problemas (*bugs*) e otimizar a aplicação.

Os resultados obtidos na elaboração do sistema DAQ-IOT demonstraram que o sistema se comportou de maneira eficiente, mas ainda apresenta possibilidades de melhoria no design final, como redução de espaço físico dos componentes e melhor layout. Por exemplo, a conexão de entrada dos sensores utilizando conectores *jst* interligados a terminais de entrada de conectores *jack P2* fêmea permitiu uma rápida instalação. No entanto, essa conexão poderia ser substituída por conectores fêmea soldados diretamente na placa de aquisição, o que reduziria o tamanho do circuito e a concentração de temperatura. Além disso, o *design* da placa poderia ser remodelado e a confecção poderia ser realizada com uma máquina de impressão de PCB industrial. No que diz respeito ao case, a confecção atual, feita com ferramentas metálicas em altas temperaturas, consumiu muito tempo de mão de obra. Para confecção em grandes unidades, seria mais viável modelar o case usando software como o Fusion 360 e realizar a impressão em 3D ou contratar o serviço de confecção do case, conhecido como "*hardware enclosure*".

Os resultados obtidos com o circuito de comando auxiliar foram satisfatórios. O circuito foi capaz de acionar corretamente a carga ao pressionar o botão de acionamento na cor verde e interromper a alimentação ao pressionar o botão de parada na cor vermelha, incluindo o correto contato de selo. Testes de acionamento via relé, utilizando um microcontrolador, foram realizados com sucesso. No entanto, por questões de segurança, esse recurso foi mantido em espera para futuros trabalhos, visando aprimorar ainda mais o sistema.

Os resultados obtidos nos protocolos de comunicação MQTT e OPC UA indicaram que ambos são adequados e possuem alto desempenho para a comunicação de dados, sem dependerem necessariamente de internet. Essa

constatação demonstra que ambos os protocolos são viáveis para aplicação no sistema IOTCORE.

Durante a configuração do broker MQTT, foi observado que o protocolo MQTT é amplamente utilizado em aplicações de IoT. No entanto, foi identificado que alguns supervisórios ainda estão em transição para suportar totalmente esse tipo de comunicação. Isso indica que, embora o MQTT seja um protocolo amplamente adotado, ainda há uma necessidade de melhor integração em alguns sistemas supervisórios específicos.

Por outro lado, o protocolo OPC UA demonstrou possuir uma ampla variedade de alternativas de integração em diversos tipos de supervisórios industriais. Essa flexibilidade e compatibilidade fazem com que o OPC UA seja uma escolha sólida para a comunicação em sistemas de supervisão industrial que possui supervisórios específicos e licenciados.

Ao utilizar o GENESIS64, optou-se pelo protocolo OPC UA. No entanto, houve uma tentativa de incorporar também o protocolo MQTT, mas sem sucesso, no qual os dados apresentaram-se unificados em um único tópico. Essa configuração talvez possa ser configurada com uma outra abordagem, para tornar possível utilizar o protocolo MQTT com o supervisório GENESIS64.

Os resultados obtidos na implementação dos bancos de dados evidenciaram a capacidade do sistema de armazenar os valores de todos os sensores com precisão e eficiência. Isso demonstra que o sistema é capaz de gerenciar adequadamente grandes quantidades de dados provenientes dos sensores.

Durante a implementação, constatou-se que o InfluxDB apresentou vantagens significativas em relação ao SQL Server. O InfluxDB foi capaz de consultar grandes quantidades de dados de forma mais rápida, em comparação ao SQL Server. Além disso, o InfluxDB oferece uma interface amigável semelhante a um sistema de supervisão. Essa interface permite ao usuário criar *dashboards* e analisar gráficos em tempo real, proporcionando uma experiência mais intuitiva e interativa, similar ao Grafana. Em contrapartida, o SQL Server fica restrito a consultas específicas de dados por meio de queries, principalmente devido à sua natureza como um banco de dados relacional e não temporal.

Embora o InfluxDB tenha se destacado em termos de capacidade de consulta e interface amigável, é importante ressaltar que as consultas dos dados via SQL

Server possuem uma estrutura mais amigável. A linguagem de consulta SQL é amplamente utilizada e conhecida, tornando o SQL Server uma opção popular para consultas de dados. Essa familiaridade pode facilitar o trabalho dos desenvolvedores e usuários que já estão familiarizados com a linguagem SQL.

Os resultados obtidos para o cálculo de desbalanceamento de fase foram aplicados corretamente, de acordo com a teoria. Isso indica que a metodologia utilizada para determinar o desbalanceamento de fase foi precisa e consistente com os conceitos teóricos estabelecidos.

No entanto, os resultados também revelaram a presença de grandes oscilações no valor de desbalanceamento de fase. Essas oscilações foram atribuídas aos ruídos presentes nas medições de corrente trifásica. Essa interferência pode afetar a precisão dos resultados e a estabilidade das medições.

Para mitigar o problema das oscilações causadas pelos ruídos nas medições, optou-se por utilizar o filtro de Kalman. Esse filtro foi aplicado para estimar o desbalanceamento de fase, levando em consideração a incerteza de medição decorrente dos ruídos nas medições de corrente. A filtragem de Kalman mostrou-se eficaz ao fornecer uma curva mais próxima da realidade, reduzindo as oscilações indesejadas.

A aplicação bem-sucedida do filtro de Kalman para estimar o desbalanceamento de fase abre portas para explorar seu potencial em outras variáveis coletadas. Por exemplo, a filtragem de Kalman pode ser utilizada para estimar a corrente trifásica ou mesmo realizar a fusão sensorial dos dados de vibração provenientes do sensor MPU6050. Essa técnica avançada de controle pode oferecer resultados mais precisos e confiáveis em diferentes aspectos do sistema.

Os resultados obtidos para os supervisórios demonstraram a flexibilidade do sistema em estruturar a visualização dos dados em diferentes sistemas e aplicações, proporcionando uma alta disponibilidade dos recursos e alta escalabilidade dos negócios. Isso indica que o sistema é capaz de atender a diversas necessidades e se adaptar a diferentes contextos de aplicação.

É importante destacar as peculiaridades específicas de cada supervisório utilizado. A aplicação no Power BI e a aplicação web em Flask apresentaram resultados similares, mas ambas não possuem atualização automática dos dados, exigindo configurações complexas para implementar essa funcionalidade. Ambas as

ferramentas oferecem uma ampla gama de recursos, sendo que a aplicação em Flask permite maior customização.

A supervisão utilizando o Genesis64 e a aplicação *mobile* mostraram-se promissoras para uma supervisão em tempo real. Ambas as aplicações possuem recursos nativos que permitem a atualização automática dos dados. No entanto, é importante ressaltar que essas aplicações consomem dados diretamente do *broker* MQTT e servidor OPC UA, o que implica que esses dados não têm a capacidade de recuperar informações ao reiniciar o serviço.

A supervisão dos dados com o Grafana demonstrou ser a mais promissora entre as ferramentas utilizadas. O Grafana é adequado para visualizar dados de séries temporais e oferece uma variedade de recursos para customização e atualização automática dos dados. Além disso, a aplicação consome os dados provenientes do banco de dados InfluxDB, permitindo a recuperação dos dados e uma visualização em tempo real.

Os resultados obtidos para a configuração de alarmes via e-mail demonstraram a grande importância das possibilidades da linguagem Python. Além de ler e fornecer dados, Python também pode ser utilizado para reportar dados, como no caso do envio de alarmes via e-mail. Isso mostra a versatilidade e a eficiência da linguagem na realização de diversas tarefas relacionadas ao sistema.

Os resultados obtidos na configuração de alarmes via e-mail mostraram a eficiência do serviço. O envio rápido de e-mails com alertas permite uma resposta ágil aos eventos e situações de alarme, contribuindo para a eficácia do sistema de monitoramento.

A definição do escopo do e-mail com orientações de como solucionar o problema demonstra uma boa prática para otimizar a velocidade de manutenção. Ao fornecer informações relevantes e direcionadas sobre como resolver o problema, o processo de manutenção torna-se mais ágil e eficiente.

Os testes realizados para a validação dos dados coletados de vibração mostraram resultados consistentes. Os dados de vibração correspondiam aos estímulos aplicados, e o valor da aceleração no eixo z foi próximo ao valor da aceleração da gravidade. Isso indica que os dados de vibração foram coletados de forma precisa e confiável, fornecendo informações relevantes sobre o comportamento do sistema.

Os testes de validação dos dados de temperatura e corrente demonstraram uma boa correspondência entre os dados coletados pelo DAQ-IOT e os valores lidos por um multímetro com leitura de corrente e temperatura. Os erros decorrentes das medições são pequenos, geralmente causados pela não linearidade dos componentes. No entanto, esses erros não comprometem a veracidade e autenticidade dos dados, permitindo uma análise precisa das variáveis de temperatura e corrente.

Essas discussões ressaltam a validação bem-sucedida dos dados coletados de vibração, temperatura e corrente. Os resultados dos testes indicam a consistência e precisão dos dados obtidos pelo sistema DAQ-IOT, proporcionando informações confiáveis para a análise e monitoramento do sistema.

Com base nas informações fornecidas no tópico "Motivação", é possível fazer uma comparação com os resultados obtidos no presente estudo, destacando as vantagens e desvantagens em relação aos estudos mencionados. Para isso, alguns aspectos foram levados em consideração, como tipo de alimentação, natureza da tecnologia, tipos de protocolo, tipo de armazenamento, e grandezas monitoradas. A Tabela 3 ilustra essa comparação.

Tabela 3 - Comparação do estudo com outros trabalhos relacionados

Estudo	Alimentação	Supervisão	Protocolo	Armazenamento	Variáveis diretas
IOTCORE (2023)	Externa (5Vdc)	Grafana, Power BI, Genesis64, Aplicação web e mobile própria	MQTT e OPC UA	SQL e Séries Temporais	Corrente trifásica, vibração e temperatura
Souza et al. (2022)	Interna (bateria)	Aplicação web SEMEQ	Não revelado	Não revelado	Vibração e temperatura
Onílio et al. (2021)	Interna (bateria)	Plataforma Ubidots	MQTT	Séries Temporais	Corrente monofásica, vibração e temperatura
Rodríguez (2020)	Externa	Grafana	MQTT	Séries Temporais	Corrente trifásica, vibração e temperatura

Fonte: Autor (2023)

6 CONCLUSÃO

Neste trabalho, objetivou-se desenvolver um sistema de automação baseado em IoT para coleta, transmissão, armazenamento e exibição de dados de temperatura, vibração e corrente trifásica em motores elétricos. Ao longo dos capítulos anteriores, discutiu-se a importância da automação na indústria, os desafios enfrentados na transição para a Indústria 4.0 e os objetivos específicos estabelecidos para alcançar os resultados almejados.

Durante a execução deste trabalho, cada um dos objetivos estabelecidos foi abordado de forma detalhada e sistemática. Os algoritmos de decisão multicritério nos permitiram selecionar a linguagem de programação mais adequadas para o desenvolvimento do sistema de automação, considerando aspectos como escalabilidade, flexibilidade e disponibilidade de recursos. Estabeleceu-se as configurações do servidor principal e virtualizou-se um servidor de *backup*, garantindo a eficiência, a disponibilidade e a integridade dos dados. Desenvolveu-se um protótipo capaz de coletar informações de temperatura, vibração e corrente elétrica, transmitindo os dados por meio do protocolo MQTT e OPC UA. Além disso, desenvolveu-se uma aplicação web para supervisionar o sistema, assim como em outras aplicações, como através de telas de supervisório no ICONICS Genesis64, Power BI, Grafana e aplicativo mobile. Também se implementou algoritmos de alertas e notificações via e-mail para identificar anomalias nos dados coletados.

Durante a análise dos resultados e discussões realizadas, observou-se que as tecnologias fechadas proporcionam uma rápida implementação quando o engenheiro possui domínio da ferramenta específica. No entanto, também identificou-se que o uso de tecnologias abertas e a aquisição de conhecimento próprio podem gerar centralização de conhecimento, demandar um longo período de implementação e tornar o sistema susceptível a bugs, além de oferecer uma baixa cadeia de ajuda devido à natureza específica da utilização criada. Essa constatação ressalta a importância de considerar cuidadosamente as escolhas tecnológicas e buscar soluções que ofereçam flexibilidade, interoperabilidade e suporte adequado.

Os estudos realizados neste trabalho trouxeram benefícios significativos para a indústria e para o campo da automação. Um dos principais benefícios é a redução da exposição dos operadores a ambientes potencialmente perigosos, uma vez que o

sistema automatizado coleta e monitora os dados de temperatura, vibração e corrente elétrica, permitindo uma detecção precoce de possíveis anomalias e ações corretivas mais rápidas. Além disso, a transição de medição pontual para medição online possibilita um acompanhamento contínuo e em tempo real do desempenho dos motores elétricos, proporcionando maior eficiência operacional e redução de custos associados a falhas e paradas não programadas.

A medição online também oferece a vantagem de fornecer dados mais precisos e atualizados em comparação com a medição pontual, permitindo uma análise mais detalhada do desempenho dos motores elétricos e a identificação de possíveis melhorias nos processos industriais.

Além disso, a implementação do sistema de automação baseado em IoT promove uma maior integração e interoperabilidade dos sistemas industriais, possibilitando a troca de dados entre diferentes plataformas e sistemas. Isso contribui para uma maior eficiência e produtividade, uma vez que os dados coletados podem ser facilmente compartilhados e analisados por diferentes setores da empresa, facilitando a tomada de decisões informadas e estratégicas.

Este trabalho abre caminho para várias oportunidades de pesquisa e desenvolvimento futuro. Alguns possíveis trabalhos futuros incluem:

- Realizar uma análise de vulnerabilidade do sistema de automação desenvolvido, identificando possíveis brechas de segurança e propondo medidas de proteção adequadas;
- Explorar a integração do sistema de automação com outros supervisórios, ampliando as opções de supervisão e controle do processo industrial;
- Desenvolver uma arquitetura de sistema redundante, garantindo maior disponibilidade e confiabilidade do sistema de automação;
- Investigar a viabilidade e os benefícios de fabricar as placas impressas em um ambiente de produção em massa, visando reduzir custos e aumentar a eficiência do processo de fabricação;
- Explorar a utilização de impressoras 3D para projetar e produzir cases personalizados para os componentes do sistema de automação, proporcionando maior flexibilidade e adaptabilidade ao ambiente de instalação;

- Realizar estudos e testes para avaliar a robustez do sistema de automação em relação a interferências eletromagnéticas, garantindo seu bom funcionamento mesmo em ambientes com altos níveis de campo magnético;
- Implementar algoritmos de predição de falhas com algoritmos de *Machine Learning*, visando um aprendizado profundo e diferentes técnicas aplicadas.

Essas são apenas algumas sugestões de possíveis trabalhos futuros que podem contribuir para o aprimoramento e expansão do sistema de automação baseado em IoT desenvolvido neste trabalho.

Os resultados obtidos no presente trabalho são relevantes não apenas para a indústria em geral, mas também para o campo da automação e tecnologia. A implementação de soluções baseadas em IoT e a adoção de tecnologias abertas e flexíveis têm o potencial de impulsionar a transformação digital e a Indústria 4.0, permitindo melhorias substanciais na eficiência, segurança, produtividade e tomada de decisões.

Portanto, os estudos realizados demonstraram a viabilidade e eficácia do sistema de automação proposto, destacando seus benefícios e apontando para várias oportunidades de desenvolvimento e aprimoramento no futuro. A adoção de tecnologias abertas, o uso de IoT e a análise de dados em tempo real têm o potencial de revolucionar a indústria, tornando-a mais eficiente, segura e competitiva em um cenário cada vez mais digital.

REFERÊNCIAS

ABHISHEK, A. S.; BHASKER, M.; PONRAJ, A. S. **IoT Based Control System for Home Automation**. 2021 IEEE 2nd International Conference on Technology, Engineering, Management for Societal impact using Marketing, Entrepreneurship and Talent (TEMSMET). **Anais**... Em: 2021 IEEE 2ND INTERNATIONAL CONFERENCE ON TECHNOLOGY, ENGINEERING, MANAGEMENT FOR SOCIETAL IMPACT USING MARKETING, ENTREPRENEURSHIP AND TALENT (TEMSMET). Pune, India: IEEE, 2 dez. 2021. Disponível em: <https://ieeexplore.ieee.org/document/9768686/>. Acesso em: 22 abr. 2023

AGRAWAL, A. **Enterprise Automation with Python: Automate Excel, Web, Documents, Emails, and Various Workloads with Easy-to-code Python Scripts**. Place of publication not identified: BPB Publications, 2022.

ALKHAFAJEE, A. R. et al. **Security and Performance Analysis of MQTT Protocol with TLS in IoT Networks**. 2021 4th International Iraqi Conference on Engineering Technology and Their Applications (IICETA). **Anais**... Em: 2021 4TH INTERNATIONAL IRAQI CONFERENCE ON ENGINEERING TECHNOLOGY AND THEIR APPLICATIONS (IICETA). Najaf, Iraq: IEEE, 21 set. 2021. Disponível em: <https://ieeexplore.ieee.org/document/9717495/>. Acesso em: 12 abr. 2023

ANANDAN, R. et al. **The Industrial Internet of Things (IIoT): Intelligent Analytics for Predictive Maintenance**. 1. ed. [s.l.] Wiley-Scrivener, 2022.

ARMENISE, M. N. et al. **Advances in gyroscope technologies**. Berlin: Springer, 2010.

ARNOLD, J. **Learning Microsoft Power BI transforming data into insights**. S.l.: O'REILLY MEDIA, 2023.

ARONOWITZ, A. **Flask Framework Cookbook: Building Web Applications with Flask**. 2nd. ed. [s.l.] Amazon Digital Services LLC - KDP Print US, 2021, 2021.

ASHOURI, M.; DAVIDSSON, P.; SPALAZZESE, R. Quality attributes in edge computing for the Internet of Things: A systematic mapping study. **Internet of Things**, v. 13, p. 100346, mar. 2021.

ASPIN, A. **Pro Power BI desktop**. Staffordshire [England]: Apress, 2016.

ATILGAN, E.; OZCELIK, I.; YOLACAN, E. N. **MQTT Security at a Glance**. 2021 International Conference on Information Security and Cryptology (ISCTURKEY). **Anais**... Em: 2021 INTERNATIONAL CONFERENCE ON INFORMATION SECURITY AND CRYPTOLOGY (ISCTURKEY). Ankara, Turkey: IEEE, 2 dez. 2021. Disponível em: <https://ieeexplore.ieee.org/document/9654337/>. Acesso em: 12 abr. 2023

BALZER, G.; SCHORN, C. **Asset management for infrastructure systems: energy and water**. Second edition ed. Cham: Springer, 2022.

BASTOS, J.; ARAÚJO, P. **Hands-On infrastructure monitoring with Prometheus: implement and scale queries, dashboards, and alerting across machines and containers**. Birmingham: Packt Publishing, Limited, 2019.

BELABED, T.; JEMMALI, S.; SOUANI, C. **FFT implementation and optimization on FPGA**. 2018 4th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP). **Anais**... Em: 2018 4TH INTERNATIONAL CONFERENCE ON ADVANCED TECHNOLOGIES FOR SIGNAL AND IMAGE PROCESSING (ATSIP). Sousse: IEEE, mar. 2018. Disponível em: <https://ieeexplore.ieee.org/document/8364454/>. Acesso em: 23 abr. 2023

BENICIO, L. P. DA S. **Manutenção, sua Conceituação e Contexto Histórico: Aplicação de Algumas Técnicas em Motores Elétricos**. Monografia—Campina Grande: Universidade Federal de Campina Grande, 2017.

BIN UZAYR, S. **Mastering Python for web: a beginner's guide**. First edition ed. Boca Raton: CRC Press, Taylor & Francis Group, 2022.

BIN UZAYR, S. (ED.). **Mastering Bootstrap: a beginner's guide**. First edition ed. Boca Raton: CRC Press, 2023.

BLOKDYK, G. **InfluxDB**. 3th. ed. [s.l.] 5STARCooks, 2018.

BRUNELLI, M. **Introduction to the Analytic Hierarchy Process**. 1st ed. 2015 ed. Cham: Springer International Publishing : Imprint: Springer, 2015.

C. HILLAR, G. **Hands-On MQTT Programming with Python: work with the lightweight iot protocol in python**. Place of publication not identified: PACKT Publishing Limited, 2018.

CAMERON, N. **Electronics projects with the ESP8266 and ESP32: building web pages, applications, and Wifi enabled devices**. Berkeley, CA: Apress L.P., 2021.

CANDEL, J. M. O. **Implementing DevSecOps with Docker and Kubernetes: an experiential guide to operate in the devops... environment for securing and monitoring container**. S.l.: BPB PUBLICATIONS, 2022.

CAO, H. **Dual-Mass Linear Vibration Silicon-Based MEMS Gyroscope**. S.l.: SPRINGER VERLAG, SINGAPOR, 2023.

CARREIRO, F. B.; NASCIMENTO, J. F.; SOUSA, G. M. L. Modelagem e Desenvolvimento de uma Solução de IoT e Gêmeo Digital para Medição de Consumo de Energia Residencial. **XXIV Congresso Brasileiro de Automática**, n. 24º, 2022.

CAVALIERI, S.; CUTULI, G. **Performance evaluation of OPC UA**. 2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010). **Anais**... Em: FACTORY AUTOMATION (ETFA 2010). Bilbao: IEEE, set. 2010. Disponível em: <http://ieeexplore.ieee.org/document/5641184/>. Acesso em: 30 abr. 2023

CERBULESCU, C. C. et al. **Optimize Critical Data Pattern Detection In Systems With Real Time Decisions**. 2022 23rd International Carpathian Control Conference

(ICCC). **Anais**... Em: 2022 23RD INTERNATIONAL CARPATHIAN CONTROL CONFERENCE (ICCC). Sinaia, Romania: IEEE, 29 maio 2022. Disponível em: <https://ieeexplore.ieee.org/document/9805932/>. Acesso em: 9 abr. 2023

CHEBEL-MORELLO, B. **From prognostics and health systems management to predictive maintenance 2**. Hoboken, NJ: ISTE Ltd/John Wiley and Sons Inc, 2017.

CONTI, G. et al. A Multi-Port Hardware Energy Meter System for Data Centers and Server Farms Monitoring. **Sensors**, v. 23, n. 1, p. 119, 23 dez. 2022.

CUNHA, J. M. **Modelo de Gerenciamento Integrado para Ambientes de Redes Industriais**. Dissertação de Mestrado—Florianópolis: Universidade Federal de Santa Catarina, 2004.

DABBAS, E. **Interactive dashboards and data apps with Plotly and Dash: harness the power of a fully fledged frontend web framework in Python - no JavaScript required**. Birmingham: Packt, 2021.

D'ANTONI, J. **SQL Server on Azure Virtual Machines**. Place of publication not identified: Packt Publishing, 2020.

DAVIDSON, L. **Pro SQL Server 2012 Relational Database Design and Implementation**. Sixth edition ed. New York: Apress, 2021.

DE SOUZA, V. C. et al. Utilização das tecnologias da indústria 4.0 na manutenção preditiva através do monitoramento de equipamentos e instalações / Use of industry 4.0 technologies in predictive maintenance through monitoring equipment and facilities. **Brazilian Journal of Development**, v. 8, n. 1, p. 7063–7083, 26 jan. 2022.

DECKLER, G. **Learn power BI: a comprehensive, step-by-step guide for beginners to learn real-world business intelligence**. Second edition ed. Birmingham: Packt Publishing, 2021.

DEY, N.; TAMANE, S. (EDS.). **Big Data Analytics for Smart and Connected Cities:** [s.l.] IGI Global, 2019.

DHAVAL, N.; ASHWIN, R. Study on Security Issues and Threats for MQTT with IoT Paradigm. **International Conference on Science, Engineering and Technology (ICSET 2022)**, p. 637–646, 2022.

DI MARTINO, S. et al. **Industrial Internet of Things: Persistence for Time Series with NoSQL Databases**. 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE). **Anais**... Em: 2019 IEEE 28TH INTERNATIONAL CONFERENCE ON ENABLING TECHNOLOGIES: INFRASTRUCTURE FOR COLLABORATIVE ENTERPRISES (WETICE). Napoli, Italy: IEEE, jun. 2019. Disponível em: <https://ieeexplore.ieee.org/document/8795406/>. Acesso em: 9 abr. 2023

DORDEVIC, B. et al. **File system performance for type-1 hypervisors on the Xen and VMware ESXi**. 2022 21st International Symposium INFOTEH-JAHORINA (INFOTEH). **Anais**... Em: 2022 21ST INTERNATIONAL SYMPOSIUM INFOTEH-

JAHORINA (INFOTEH). East Sarajevo, Bosnia and Herzegovina: IEEE, 16 mar. 2022. Disponível em: <https://ieeexplore.ieee.org/document/9751288/>. Acesso em: 2 abr. 2023

ESPRESSIF SYSTEMS. **ESP32**. Disponível em: <https://www.espressif.com/en/products/socs/esp32>. Acesso em: 2 abr. 2023.

FARITOVICH, F. A. **Usando o pacote Genesis64 Scada para construir sistemas de controle de supervisão, aquisição de dados e controle**. Trabalho de Qualificação de Graduação (Tese de Bacharelado) — Kazan: Ministério da Educação e Ciência da Federação Russa. Universidade Federal de Kazan (Privolzhsky), 2016.

FOUDA, E. **A complete guide to Docker for operations and development: test-prep for the Docker Certified Associate (DCA) exam**. Berkeley: Apress, 2022.

GARAPATI, Y. et al. **An Analytical Hierarchy Process Investigation on High Speed Data Implementations Using Big Data**. 2022 International Conference on Computer Communication and Informatics (ICCCI). **Anais**... Em: 2022 INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATION AND INFORMATICS (ICCCI). Coimbatore, India: IEEE, 25 jan. 2022. Disponível em: <https://ieeexplore.ieee.org/document/9741015/>. Acesso em: 22 abr. 2023

GKATZIOURAS, E. **A Developer's Essential Guide to Docker Compose Simplify the Development and Orchestration of Multi-Container Applications**. Birmingham: Packt Publishing, Limited, 2022.

GOMES, W. L. P. G.; SERUFFO, M. C. D. R.; SILVEIRA, A. Proposal of a multi-criteria decision-making model for the implementation of a supervisory system for industry 4.0. **International Journal of Management and Decision Making**, v. 1, n. 1, p. 1, 2024.

GORMAN, K. **Introducing Microsoft SQL Server 2019: reliability, scalability, and security both on premises and in the cloud**. Birmingham, UK: Packt Publishing, 2020.

HAMDANI, S.; SBEYTI, H. **A Comparative study of COAP and MQTT communication protocols**. 2019 7th International Symposium on Digital Forensics and Security (ISDFS). **Anais**... Em: 2019 7TH INTERNATIONAL SYMPOSIUM ON DIGITAL FORENSICS AND SECURITY (ISDFS). Barcelos, Portugal: IEEE, jun. 2019. Disponível em: <https://ieeexplore.ieee.org/document/8757486/>. Acesso em: 12 abr. 2023

HASSAN, A. et al. **Statistical scheme for fault detection using Arduino and MPU 6050**. 2019 Prognostics and System Health Management Conference (PHM-Qingdao). **Anais**... Em: 2019 PROGNOSTICS AND SYSTEM HEALTH MANAGEMENT CONFERENCE (PHM-QINGDAO). Qingdao, China: IEEE, out. 2019. Disponível em: <https://ieeexplore.ieee.org/document/8942922/>. Acesso em: 21 abr. 2023

HATTERSLEY, L. **The Official Raspberry Pi Handbook**. Cambridge: Raspberry Pi (Trading) Ltd, 2021.

HERNANDEZ, R. J. **Building IoT visualizations using Grafana: power up your IoT projects with Grafana**. Birmingham: Packt Publishing, 2022.

HIGHAM, A.; CHALLENDER, J.; WATTS, G. **Introduction to built asset management**. Hoboken, NJ: John Wiley & Sons, 2022.

HILLAR, G. C. **MQTT essentials: a lightweight IoT protocol: the preferred IoT publish-subscribe lightweight messaging protocol**. Birmingham, UK: Packt Publishing, 2017.

HOLTON, T. **Digital signal processing: principles and applications**. Cambridge, United Kingdom ; New York, NY, USA: Cambridge University Press, 2020.

HOSSAIN, N. et al. Cyber security risk assessment method for SCADA system. **Information Security Journal: A Global Perspective**, v. 31, n. 5, p. 499–510, 3 set. 2022.

HUGHES, A.; DRURY, B. **Electric motors and drives: fundamentals, types and applications**. Fifth edition ed. Oxford [England] ; Cambridge, MA: Newnes, 2019.

HYMAN, J. **Microsoft Power Bi For Dummies**. 1st. ed. Indianapolis: John Wiley and Sons, 2022.

ICONICS. **Genesis64 Overview**. Disponível em: <https://iconics.com/products/genesis64>. Acesso em: 16 abr. 2023.

INFLUXDATA. **InfluxDB: Purpose-Built Open Source Time Series Database**. Disponível em: <https://www.influxdata.com/products/influxdb-overview/>. Acesso em: 9 abr. 2023.

INVENSENSE INC. **MPU-6000/MPU-6050 Product Specification**. , 2013. Disponível em: <https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf>. Acesso em: 21 abr. 2023

IQBAL, A.; PATTINSON, C.; KOR, A.-L. **Performance monitoring of Virtual Machines (VMs) of type I and II hypervisors with SNMPv3**. 2015 World Congress on Sustainable Technologies (WCST). **Anais**... Em: 2015 WORLD CONGRESS ON SUSTAINABLE TECHNOLOGIES (WCST). London, United Kingdom: IEEE, dez. 2015. Disponível em: <http://ieeexplore.ieee.org/document/7415127/>. Acesso em: 2 abr. 2023

JAPON, B. R. **LEARN IOT PROGRAMMING USING NODE-RED begin to code full stack iot apps and edge devices with... raspberry pi, nodejs, and grafana**. S.l.: BPB PUBLICATIONS, 2022.

JAPÓN, B. R. **Learn IoT Programming Using Node-RED: Begin to Code Full Stack IoT Apps and Edge Devices with Raspberry Pi, NodeJS and Grafana**. S.l.: BPB PUBLICATIONS, 2022.

JAZDI, N. **Cyber physical systems in the context of Industry 4.0**. 2014 IEEE International Conference on Automation, Quality and Testing, Robotics. **Anais**... Em:

2014 IEEE INTERNATIONAL CONFERENCE ON AUTOMATION, QUALITY AND TESTING, ROBOTICS (AQTR). Cluj-Napoca, Romania: IEEE, maio 2014. Disponível em: <http://ieeexplore.ieee.org/document/6857843/>. Acesso em: 25 mar. 2023

JUNIOR, E. G. **Introdução a Sistemas de Supervisão, Controle e Aquisição de Dados**. 1. ed. Rio de Janeiro: Alta Books, 2019.

KANE, S. P.; MATTHIAS, K. **Docker: up & running**. Third edition ed. Sebastopol, California: O'Reilly Media, Inc., 2023.

KARADUMAN, B.; CHALLENGER, M. **Model-driven Development for ESP-based IoT Systems**. 2021 IEEE/ACM 3rd International Workshop on Software Engineering Research and Practices for the IoT (SERP4IoT). **Anais**... Em: 2021 IEEE/ACM 3RD INTERNATIONAL WORKSHOP ON SOFTWARE ENGINEERING RESEARCH AND PRACTICES FOR THE IOT (SERP4IOT). Madrid, Spain: IEEE, jun. 2021. Disponível em: <https://ieeexplore.ieee.org/document/9516819/>. Acesso em: 2 abr. 2023

KATTI, B. et al. **SA-OPC-UA: Introducing Semantics to OPC-UA Application Methods**. 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE). **Anais**... Em: 2018 IEEE 14TH INTERNATIONAL CONFERENCE ON AUTOMATION SCIENCE AND ENGINEERING (CASE). Munich, Germany: IEEE, ago. 2018. Disponível em: <https://ieeexplore.ieee.org/document/8560467/>. Acesso em: 30 abr. 2023

KHWANRIT, R. et al. **Accuracy Comparison of Present Low-cost Current Sensors for Building Energy Monitoring**. 2018 International Conference on Embedded Systems and Intelligent Technology & International Conference on Information and Communication Technology for Embedded Systems (ICESIT-ICICTES). **Anais**... Em: 2018 11TH INTERNATIONAL CONFERENCE ON EMBEDDED SYSTEMS AND INTELLIGENT TECHNOLOGY & 9TH INTERNATIONAL CONFERENCE ON INFORMATION AND COMMUNICATION TECHNOLOGY FOR EMBEDDED SYSTEMS (ICESIT-ICICTES). Khon Kaen: IEEE, maio 2018. Disponível em: <https://ieeexplore.ieee.org/document/8442066/>. Acesso em: 22 abr. 2023

KOROTKEVITCH, D. **SQL server advanced troubleshooting and performance tuning: best practices and techniques**. First edition ed. Beijing Boston Farnham Sebastopol Tokyo: O'reilly, 2022.

LACHEV, T. **Applied Microsoft Power BI: Bring your data to life!** 7th. ed. [s.l.] Prologika Press, 2022.

LAROCK, T.; LAAR, E. VAN DE. **Pro SQL Server 2022 wait statistics: a practical guide to analyzing performance in SQL Server and Azure SQL Database**. 3rd ed ed. New York, NY: Apress, 2023.

LATHKAR, M. **Building web apps with Python and Flask learn to develop and deploy responsive RESTful web applications using Flask framework**. First edition ed. New Delhi, India: BPB Publications, 2021.

LEE, C.; KIM, N.; HONG, S. Toward Industrial IoT: Integrated Architecture of an OPC UA Synergy Platform. **IEEE Access**, v. 9, p. 164720–164731, 2021.

LESZKO, R. **Continuous delivery with Docker and Jenkins: create secure applications by building complete CI/CD pipelines**. Third edition ed. Birmingham, UK: Packt Publishing, 2022.

LEVCHENKO, R.; CARDOSO, E. A. **System Center 2016 virtual machine manager cookbook: design, configure, and manage an efficient virtual infrastructure with VMM in System Center 2016**. Third edition ed. Birmingham, UK: Packt Publishing, 2018.

LEWIS, F. L.; XIE, L.; POPA, D. **Optimal and robust estimation: with an introduction to stochastic control theory**. 2nd ed ed. Boca Raton: CRC Press, 2008.

LI, Q. et al. **Kalman Filter and Its Application**. 2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS). **Anais**... Em: 2015 8TH INTERNATIONAL CONFERENCE ON INTELLIGENT NETWORKS AND INTELLIGENT SYSTEMS (ICINIS). Tianjin, China: IEEE, nov. 2015. Disponível em: <http://ieeexplore.ieee.org/document/7528889/>. Acesso em: 23 abr. 2023

LOTH, A.; VOGEL, P. **Datenvisualisierung mit Power BI**. 1. Auflage ed. Frechen: mitp, 2022.

MAHNKE, W.; LEITNER, S.-H.; DAMM, M. **OPC unified architecture**. Berlin: Springer, 2009.

MCCOLLAM, R. **Getting started with Grafana: real-time dashboards for IT and business operations**. New York: Apress, 2022.

MERCHANT, B. **MICROSOFT POWER BI PERFORMANCE BEST PRACTICES: a comprehensive guide to building fast power bi... solutions that guarantee robust and consistent per**. S.l.: PACKT PUBLISHING LIMITED, 2022.

MICROSOFT. **Preços do Power BI: Análise para toda organização**. Disponível em: <https://powerbi.microsoft.com/pt-br/pricing/>. Acesso em: 19 abr. 2023.

MIKHAYLOVNA, V. O.; ALEKSANDROVNA, K. A. Organization Methodology of Automated Measurement System Using the Program SCADA GENESIS64. **Collection of scientific papers of the International Scientific and Technical Symposium, International Kosygin Forum**, v. 3th, p. 13–20, 2019.

MIRON-ALEXE, V. **Comparative study regarding measurements of different AC current sensors**. 2016 International Symposium on Fundamentals of Electrical Engineering (ISFEE). **Anais**... Em: 2016 INTERNATIONAL SYMPOSIUM ON FUNDAMENTALS OF ELECTRICAL ENGINEERING (ISFEE). Bucharest, Romania: IEEE, jun. 2016. Disponível em: <http://ieeexplore.ieee.org/document/7803152/>. Acesso em: 22 abr. 2023

MISHRA, B.; KERTESZ, A. The Use of MQTT in M2M and IoT Systems: A Survey. **IEEE Access**, v. 8, p. 201071–201086, 2020.

MONK, S. **Raspberry Pi cookbook: software and hardware problems and solutions**. Fourth edition ed. Beijing Boston Farnham Sebastopol Tokyo: O'Reilly, 2023.

MORATO, A. et al. Assessment of Different OPC UA Implementations for Industrial IoT-Based Measurement Applications. **IEEE Transactions on Instrumentation and Measurement**, v. 70, p. 1–11, 2021.

MU, E.; PEREYRA-ROJAS, M. **Practical Decision Making using Super Decisions v3: An Introduction to the Analytic Hierarchy Process**. 1st ed. 2018 ed. Cham: Springer International Publishing : Imprint: Springer, 2018.

MUHLBAUER, N. et al. **Open-Source OPC UA Security and Scalability**. 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). **Anais**... Em: 2020 25TH IEEE INTERNATIONAL CONFERENCE ON EMERGING TECHNOLOGIES AND FACTORY AUTOMATION (ETFA). Vienna, Austria: IEEE, set. 2020. Disponível em: <https://ieeexplore.ieee.org/document/9212091/>. Acesso em: 30 abr. 2023

NAYYAR, A.; NAVED, M.; RAMESHWAR, R. (EDS.). **New horizons for Industry 4.0 in modern business**. Cham, Switzerland: Springer, 2023.

NIRANJANA, R. et al. **Effectual Home Automation using ESP32 NodeMCU**. 2022 International Conference on Automation, Computing and Renewable Systems (ICACRS). **Anais**... Em: 2022 INTERNATIONAL CONFERENCE ON AUTOMATION, COMPUTING AND RENEWABLE SYSTEMS (ICACRS). Pudukkottai, India: IEEE, 13 dez. 2022. Disponível em: <https://ieeexplore.ieee.org/document/10028992/>. Acesso em: 2 abr. 2023

ONER, V. O. **Developing IoT projects with ESP32: automate your home or business with inexpensive Wi-Fi devices**. First published ed. Birmingham Mumbai: Packt, 2021.

ONÍLIO, A. S. et al. **IoT em Motores de Indução Trifásicos**. Monografia—São Paulo: Universidade São Judas Tadeu, 2021.

OSAMA, A. **Professional SQL Server high availability and disaster recovery**. Birmingham, UK: Packt Publishing, 2019.

PARIKH, D. **Raspberry Pi and MQTT essentials: the complete guide to help you build innovative full-scale prototype projects**. Birmingham: Packt Publishing, 2022.

PATIL, S. M.; VIJAYALASHMI, M.; TAPASKAR, R. **IoT based solar energy monitoring system**. 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS). **Anais**... Em: 2017 INTERNATIONAL CONFERENCE ON ENERGY, COMMUNICATION, DATA ANALYTICS AND SOFT COMPUTING (ICECDS). Chennai: IEEE, ago. 2017. Disponível em: <https://ieeexplore.ieee.org/document/8389711/>. Acesso em: 22 abr. 2023

PAULO, A. B.; OLIVEIRA, S. C. NodeI4.0:integrando sistemas legados à indústria 4.0. **Revista de Engenharia e Pesquisa Aplicada**, v. 2, n. 4, 30 dez. 2017.

PENA-CABRERA, M.; LOMAS, V.; LEFRANC, G. **Fourth industrial revolution and its impact on society**. 2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON). **Anais**... Em: 2019 IEEE CHILEAN CONFERENCE ON ELECTRICAL, ELECTRONICS ENGINEERING, INFORMATION AND COMMUNICATION TECHNOLOGIES (CHILECON). Valparaiso, Chile: IEEE, nov. 2019. Disponível em: <https://ieeexplore.ieee.org/document/8988083/>. Acesso em: 25 mar. 2023

PENG, K. **Equipment management in the post-maintenance era: advancing in the era of smart machines**. Second Edition ed. Boca Raton: Taylor and Francis, 2020.

PERALTA, J. H. **Microservice APIs: using Python, Flask, FastAPI, OpenAPI and more**. Shelter Island, NY: Manning Publications Co, 2023.

PETKOVIĆ, D. **Microsoft SQL Server 2019: a beginner's guide**. Seventh edition ed. New York [NY]: McGraw Hill, 2020.

POLLACK, E.; STRATE, J. **Expert performance indexing in Azure SQL and SQL Server 2022: toward faster results and lower maintenance both on premises and in the Cloud**. 4th ed ed. New York, NY: Apress, 2023.

PRATAMA, B. G.; YULIANI, O. **Monitoring and Controlling Thermal Comfort in Air Conditioner Using YoloV4 And Predicted Mean Vote**. 2021 7th International Conference on Electrical, Electronics and Information Engineering (ICEEIE). **Anais**... Em: 2021 7TH INTERNATIONAL CONFERENCE ON ELECTRICAL, ELECTRONICS AND INFORMATION ENGINEERING (ICEEIE). Malang, Indonesia: IEEE, 2 out. 2021. Disponível em: <https://ieeexplore.ieee.org/document/9616849/>. Acesso em: 12 abr. 2023

QASIM, I. et al. A Model-Driven Mobile HMI Framework (MMHF) for Industrial Control Systems. **IEEE Access**, v. 8, p. 10827–10846, 2020.

RADOVICI, A.; CULIC, I. **Getting started with secure embedded systems: developing IoT systems for micro:bit and Raspberry Pi Pico using Rust and TockOS**. New York, NY: Apress, 2022.

ROBINSON, T. V. **Building virtual machine labs: a hands-on guide**. Place of publication not identified: T. Robinson, 2017.

RODRÍGUEZ, C. G. **Cloud Monitoring System for Industrial Engines Maintenance**. Master's degree—Barcelona: Universitat Politècnica de Catalunya, 2020.

SALITURO, E. **Learn Grafana 7.0**. 1st edition ed. Erscheinungsort nicht ermittelbar: Packt Publishing, 2020.

SAMETRIYA, D.; VASAVADA, N. **Cracking Containers with Docker and Kubernetes: the definitive guide to docker, kubernetes, and... the container ecosystem across cloud and on-premis**. S.l.: BPB PUBLICATIONS, 2022.

SCHROEDER, A.; MAYER, C.; WARD, A. M. **The book of Dash: build Dashboards with Python and Plotly**. San Francisco: No Starch Press, Inc, 2022.

SCHWARZINGER, A. **Digital Signal Processing in Modern Communication Systems**. 2nd. ed. Lake Mary, Florida: AbeBooks Seller, 2022.

SELVAM, J. **LEARN MICROPYTHON WITH ESP32: Python Programming, Raspberry Pi, Micro-python Modules, Bme280 Environment Sensor, Max7219 8x8 Matrix Display, Micro-python Projects And More**. [s.l.] Kindle Edition, 2022.

SHI, H.; NIU, L.; SUN, J. **Construction of Industrial Internet of Things Based on MQTT and OPC UA Protocols**. 2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA). **Anais**... Em: 2020 IEEE INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE AND COMPUTER APPLICATIONS (ICAICA). Dalian, China: IEEE, jun. 2020. Disponível em: <https://ieeexplore.ieee.org/document/9182598/>. Acesso em: 12 abr. 2023

SHOVIC, J. C. **Raspberry Pi IoT projects: prototyping experiments for makers**. Second edition ed. New York, NY: Apress, 2021.

SILVA, E. M. F. O. DA. **A Multi-Criteria Framework to Assist on the Design of Internet-of-Things Systems**. Dissertação—Lisbon, Portugal: Universidade Nova de Lisboa, jan. 2020.

SILVA, F. C. DE S. Ainda em busca da convergência. **Revista Controle & Instrumentação**, n. 243, 2021.

SINGH, P. **Deploy machine learning models to production: with Flask, Streamlit, Docker, and Kubernetes on Google Cloud Platform**. Berkeley, CA: Apress L.P., 2021.

SMART, G. **Practical Python Programming for IoT: Build advanced IoT projects using a Raspberry Pi 4, MQTT, RESTful APIs, WebSockets, and Python 3**. 1st. ed. [s.l.] Packt Publishing Ltd., 2020.

SOUSA, R.; TAIRA, G. R.; PARK, S. W. Integração do sistema ciber-físico para sistema de programação, intertravamento e controle de um reator batelada. **The Journal of Engineering and Exact Sciences**, v. 5, n. 5, p. 0424–0432, 20 dez. 2019.

STAPLE, D. **Robotics at Home with Raspberry Pi Pico Build autonomous robots with the versatile low-cost Raspberry Pi Pico controller and Python**. Birmingham: Packt Publishing Limited, 2023.

STEPHENS, M. P. **Productivity and reliability-based maintenance management**. Second edition ed. West Lafayette, Indiana: Purdue University Press, 2022.

SUJATHA, R.; PRAKASH, G.; JHANJHI, N. Z. (EDS.). **Cyber security applications for industry 4.0**. First edition ed. Boca Raton: Chapman & Hall/CRC Press, 2023.

TAMAS, L. et al. **State estimation based on Kalman filtering techniques in navigation**. 2008 IEEE International Conference on Automation, Quality and Testing, Robotics. **Anais**... Em: 2008 IEEE INTERNATIONAL CONFERENCE ON AUTOMATION, QUALITY AND TESTING, ROBOTICS. Cluj-Napoca, Romania: IEEE,

2008. Disponível em: <http://ieeexplore.ieee.org/document/4588811/>. Acesso em: 23 abr. 2023

THE RASPBERRY PI FOUNDATION. **Home**. Disponível em: <https://www.raspberrypi.org/>. Acesso em: 2 abr. 2023.

THOMSON, W. T. **Vibration monitoring of induction motors: practical diagnosis of faults via industrial case studies**. First edition ed. Cambridge ; New York, NY: Cambridge University Press, 2020.

VEICHTLBAUER, A.; ORTMAYER, M.; HEISTRACHER, T. **OPC UA integration for field devices**. 2017 IEEE 15th International Conference on Industrial Informatics (INDIN). **Anais**... Em: 2017 IEEE 15TH INTERNATIONAL CONFERENCE ON INDUSTRIAL INFORMATICS (INDIN). Emden: IEEE, jul. 2017. Disponível em: <http://ieeexplore.ieee.org/document/8104808/>. Acesso em: 30 abr. 2023

VMWARE INC. **Delivering a Digital Foundation for Businesses**. Disponível em: <https://www.vmware.com/>. Acesso em: 2 abr. 2023.

WAHLBRINCK, A. J. **Monitoramento da Rede Elétrica por meio de Aplicação Web e IoT**. Monografia—Lajeado: Universidade do Vale do Taquari, 2018.

WORLD INTELLECTUAL PROPERTY ORGANIZATION. **Índice Global de Inovação: executive summary /**. [s.l.] Unknown, 2022.

WU, E.; MASLOV, D. **Raspberry Pi retail applications: transform your business with a low-cost single-board computer**. United States: Apress, 2022.

WUKKADADA, B. et al. **Comparison with HTTP and MQTT In Internet of Things (IoT)**. 2018 International Conference on Inventive Research in Computing Applications (ICIRCA). **Anais**... Em: 2018 INTERNATIONAL CONFERENCE ON INVENTIVE RESEARCH IN COMPUTING APPLICATIONS (ICIRCA). Coimbatore: IEEE, jul. 2018. Disponível em: <https://ieeexplore.ieee.org/document/8597401/>. Acesso em: 12 abr. 2023

YILDIZ, S.; BURUNKAYA, M. **Web Based Smart Meter for General Purpose Smart Home Systems with ESP8266**. 2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT). **Anais**... Em: 2019 3RD INTERNATIONAL SYMPOSIUM ON MULTIDISCIPLINARY STUDIES AND INNOVATIVE TECHNOLOGIES (ISMSIT). Ankara, Turkey: IEEE, out. 2019. Disponível em: <https://ieeexplore.ieee.org/document/8932931/>. Acesso em: 22 abr. 2023

YUKHIMETS, D.; YUDINKOV, E. **The Implementation Method of Application Program Interface for Manipulator Mitsubishi RV-2FB**. 2019 International Science and Technology Conference “EastConf”. **Anais**... Em: 2019 INTERNATIONAL SCIENCE AND TECHNOLOGY CONFERENCE “EASTCONF”. Vladivostok, Russia: IEEE, mar. 2019. Disponível em: <https://ieeexplore.ieee.org/document/8725388/>. Acesso em: 22 abr. 2023

ZURAWSKI, R. (ED.). **Industrial communication technology handbook**. Second edition ed. Boca Raton London New York: CRC Press, Taylor & Francis Group, 2015.

APÊNDICE A – Instalação Sistema Operacional no Raspberry Pi

Instalação do Raspberry Pi Imager

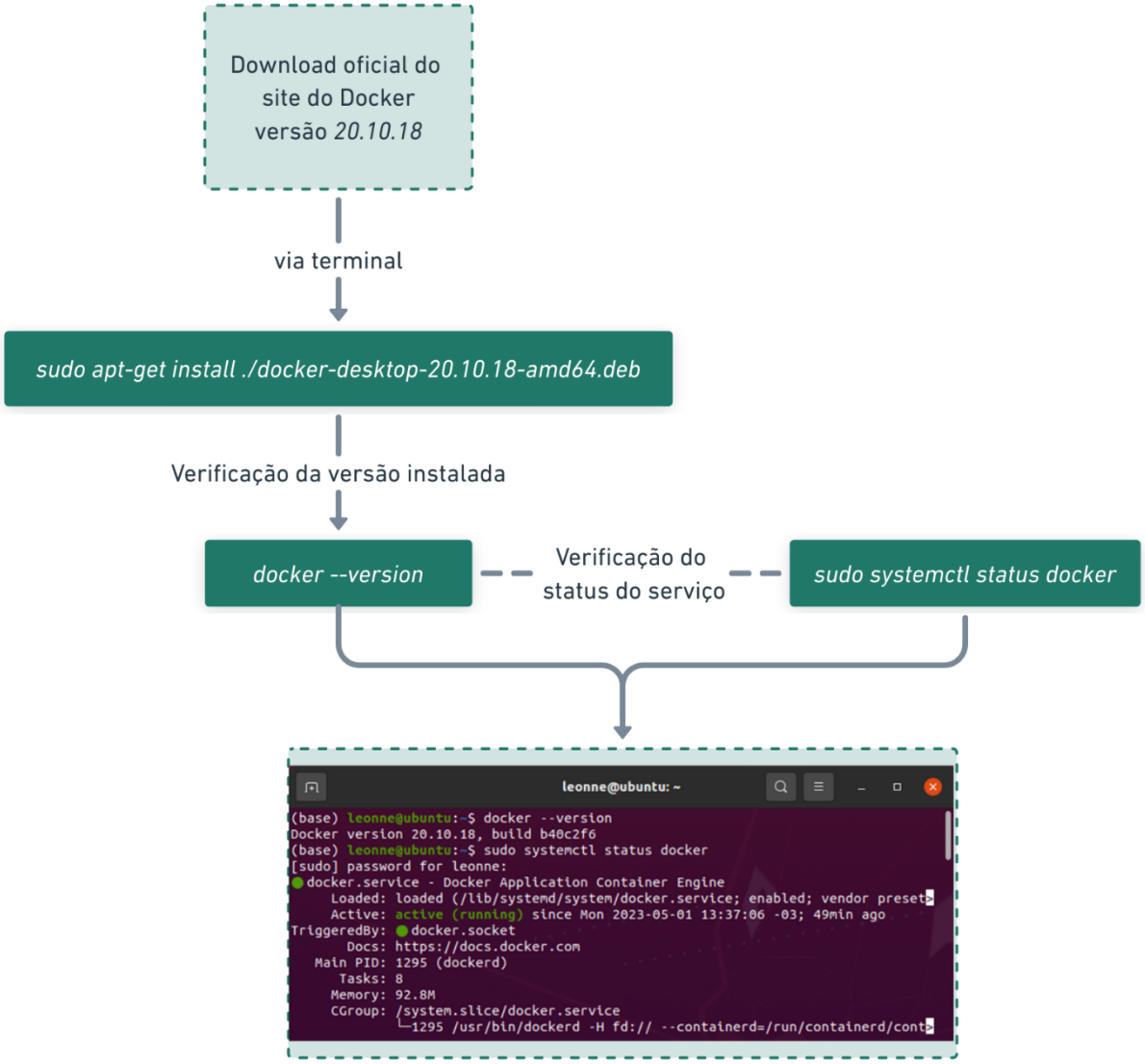
Download Raspberry Pi Imager

Excute Raspberry Pi Imager

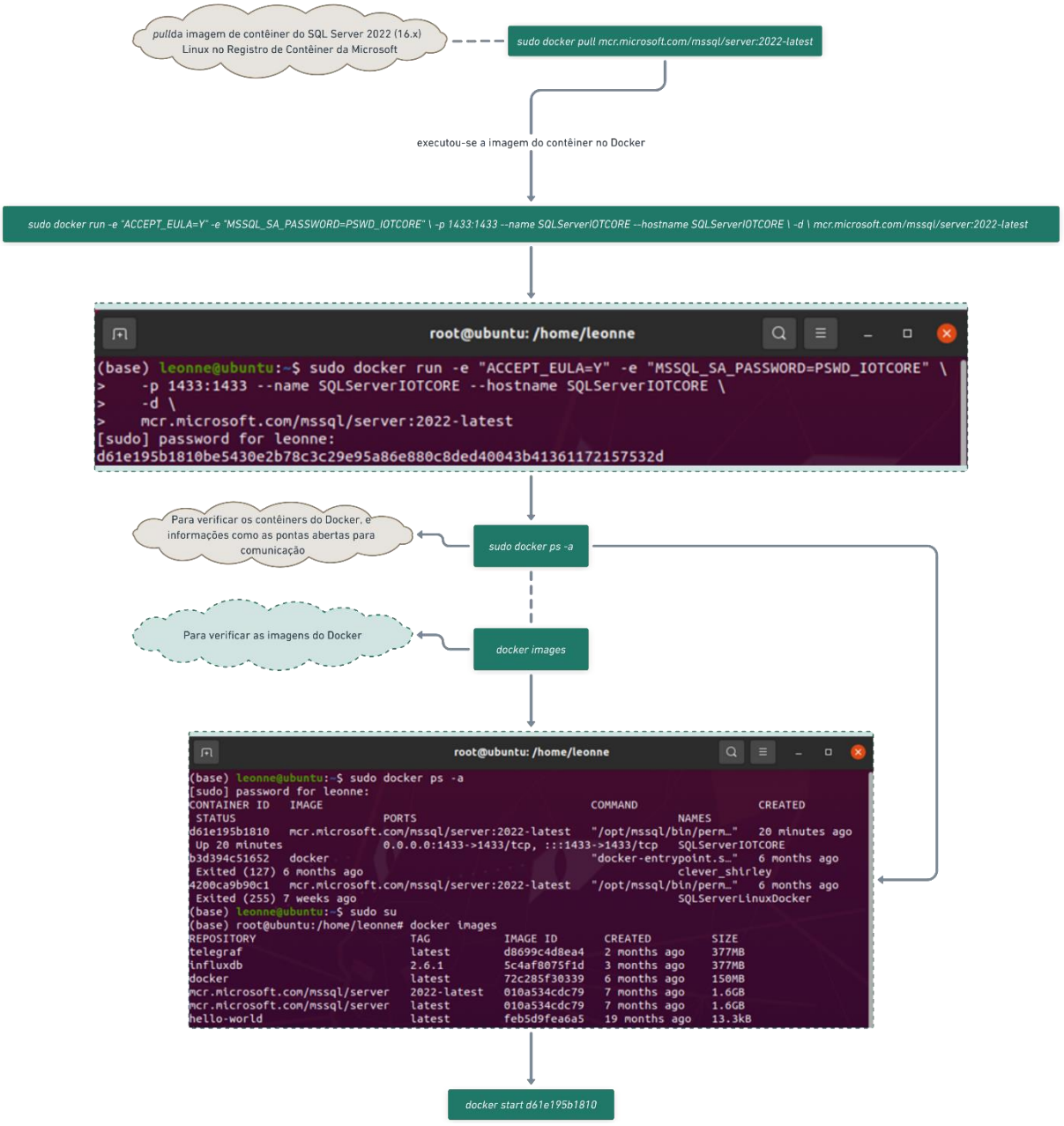
Na interface do assistente, deve-se escolher o sistema operacional Raspberry Pi OS (32-bit), escolher o dispositivo de armazenamento externo, no caso o catão de memória de 16GB, e então clicar em "*WRITE*" para realizar o download do SO e instalar no cartão de memória

Após a conclusão, deve-se inserir o cartão de memória no Raspberry Pi e ligar o mesmo, o SO então carregará.

APÊNDICE B – Instalação Docker



APÊNDICE C – Instalação da Imagem do SQL Server via Docker



APÊNDICE D – Inicialização do ambiente Anaconda e Spyder