

TRABALHO FINAL DE GRADUAÇÃO – DEZEMBRO/2024
UNIVERSIDADE FEDERAL DE ITAJUBÁ
ENGENHARIA DE CONTROLE E AUTOMAÇÃO

**DESENVOLVIMENTO DE UM SISTEMA SUPERVISÓRIO PARA
AUTOMAÇÃO DE UMA ESTAÇÃO DE CONTROLE DE MOVIMENTO
UTILIZANDO FERRAMENTAS E TECNOLOGIAS DE CÓDIGO
ABERTO**

Rafael Coelho Paes - 2019000081

Orientador: Jeremias Barbosa Machado

Resumo: Este artigo apresenta o desenvolvimento de um sistema supervisório para automação industrial, empregando exclusivamente recursos de código aberto. A motivação para esta pesquisa reside na busca por soluções acessíveis e flexíveis para a indústria, promovendo a redução de custos e a liberdade de personalização. O trabalho descreve a seleção e integração de ferramentas de código aberto para cada etapa do desenvolvimento do sistema supervisório, incluindo a aquisição de dados, visualização em tempo real, armazenamento e análise de dados, e interface de usuário. Algumas tecnologias utilizadas foram o padrão OPC UA, banco de dados MySQL e a biblioteca Python voltada para ciência de dados Streamlit para a plataforma web. Durante a realização do trabalho, foram utilizadas técnicas de programação para implementar a comunicação entre os dispositivos da rede, a lógica de controle do Servo Motor e a interface de usuário web. Como teste prático, utiliza-se uma bancada didática da empresa B&R, contendo um CLP, um Servo Motor e um drive para controle do movimento. Os resultados obtidos demonstram a viabilidade e eficácia da solução proposta, fornecendo uma plataforma robusta e flexível para a automação industrial. A implementação bem-sucedida deste projeto contribui significativamente para a melhoria da eficiência operacional e a modernização dos processos de controle de movimento em ambientes industriais, alinhando-se com as demandas atuais por automação avançada e conectividade na indústria 4.0.

Palavras-Chave: Automação Industrial, IHM, Controle de Movimento, OPC UA, Banco de dados, Aplicação Web

I INTRODUÇÃO

Na indústria moderna, a utilização de sistemas supervisórios é essencial para monitorar e controlar processos industriais de maneira eficiente e segura. Tradicionalmente, muitas empresas recorrem a sistemas supervisórios pagos, que oferecem uma gama de recursos e suporte técnico especializado. No entanto, essa abordagem não está isenta de desafios e limitações.

Os sistemas supervisórios pagos muitas vezes im-

põem altos custos de licenciamento e manutenção, além de também exigirem um treinamento complexo de todas suas funcionalidades e aplicações, o que pode representar um ônus significativo para empresas de pequeno e médio porte, bem como para iniciativas de automação em ambientes acadêmicos ou de pesquisa. Além disso, a dependência de fornecedores comerciais pode resultar em falta de flexibilidade e liberdade para personalizar e adaptar o sistema às necessidades específicas de cada aplicação industrial.

Dante desse cenário, surge a necessidade de explorar alternativas que ofereçam uma abordagem mais acessível, flexível e transparente para o desenvolvimento de sistemas supervisórios industriais. As ferramentas de código aberto surgem como uma solução promissora, permitindo que empresas e desenvolvedores tenham acesso a recursos poderosos e comunidades de colaboração, sem as restrições impostas pelos modelos de negócio proprietários.

II CONTROLE DISTRIBUÍDO

O controle descentralizado é uma abordagem de gestão e operação de sistemas de automação industrial onde o controle é distribuído entre vários dispositivos ou unidades independentes, em vez de ser centralizado em um único controlador principal. Neste sistema, cada unidade de controle local opera de forma autônoma, gerenciando uma parte específica do processo ou da máquina, mas ainda se comunica com outras unidades para coordenar atividades e compartilhar informações.

Algumas vantagens dessa metodologia incluem a **escalabilidade**, pois facilita a expansão do sistema e a **redundância**, pois a falha de um nó local não comprometerá todo o sistema, aumentando a robustez e a confiabilidade geral.

III ESTRUTURA DE SOFTWARE

III.1 Automation Studio

Uma vez que os equipamentos utilizados são da fabricante austríaca B&R (Berneker & Rainer), foi escolhido utilizar o próprio ambiente de desenvolvimento da fabricante, o *Automation Studio*, para realizar as configurações iniciais do servo motor, da rede e da conectividade com o servidor OPC UA.

Desde a configuração de hardware até a implementação de lógica de controle e visualização de processos em tempo real, o *Automation Studio* proporciona suporte para uma variedade de protocolos de comunicação, dispositivos de campo e tecnologias de controle, possibilitando uma experiência unificada.

Por se tratar de um software proprietário, sua utilização no meio industrial exige a posse de uma licença para que o desenvolvedor tenha proveito de todos os recursos do programa, tornando sua utilização inviável para empresas que não possuem condições financeiras para manter o licenciamento do programa de forma contínua.

III.2 Linguagem de Programação Python

A linguagem Python, criada por Guido van Rossum e lançada pela primeira vez em meados de 1990 [1], é uma linguagem de programação de alto nível, interpretada, dinâmica e de propósito geral, amplamente utilizada em uma variedade de domínios, desde desenvolvimento web e científico até automação de tarefas e aprendizado de máquina.

Python é conhecida por sua versatilidade e facilidade de aprendizado, permitindo aos desenvolvedores criar rapidamente protótipos de ideias e soluções, bem como desenvolver aplicações complexas em um tempo mais curto do que em muitas outras linguagens de programação. Devido à estas características, a utilização do Python no campo da Tecnologia de Informação vem crescendo exponencialmente [2]. A versão a ser utilizada para o desenvolvimento deste presente trabalho é a 3.12.2.

III.3 Banco de Dados MySQL

Desenvolvido, distribuído e suportado pela Oracle Corporation, o MySQL é um dos sistemas de gerenciamento de banco de dados relacional em código aberto mais populares e amplamente utilizados em todo o mundo [3].

Além disso, o MySQL é conhecido por sua escalabilidade, suportando desde instâncias simples em ambientes de desenvolvimento até clusters de servidores em produção, capazes de lidar com grandes volumes de dados e cargas de trabalho intensas. Para a aplicação a ser desenvolvida, essa característica é crucial, uma vez que o sistema supervisório receberá grandes quantidades de dados constantemente.

III.4 Padrão de Comunicação OPC UA

O OPC UA (Open Platform Communications Unified Architecture) é um padrão de comunicação aberto e independente de plataforma, desenvolvido para facilitar a interoperabilidade e a integração de sistemas em ambientes industriais complexos.

Criado como uma evolução do protocolo OPC clássico, em 2008 [4], o OPC UA supera as limitações de suas versões anteriores, oferecendo uma série de recursos avançados, incluindo segurança aprimorada, escalaabilidade, flexibilidade e capacidade de comunicação em tempo real.

III.5 Interface homem-máquina com a Biblioteca Python Streamlit

A linguagem de programação Python oferece diversas bibliotecas para desenvolvimento web e ciência de dados, entre elas: *Django*, *Pyramid*, *Flask*, *Tkinter*, *Pandas*, *Matplotlib*, etc. Cada uma possuindo suas vantagens e aplicações específicas. Para o desenvolvimento da plataforma do Supervisório Web, foi escolhida a biblioteca *Streamlit*, biblioteca com potencial de criar aplicativos web interativos de forma intuitiva e com economia de tempo. Tudo isso sendo uma biblioteca de código aberto, permitindo uma vasta participação da comunidade de desenvolvedores.

IV INTEGRAÇÃO DAS TECNOLOGIAS

Tendo em vista todos os recursos de software que serão utilizados, é necessário, agora, criar uma maneira para que todos estes mecanismos comuniquem entre si, possibilitando uma aplicação eficiente. O principal método será através de diversas bibliotecas Python, que, quando usadas em conjunto, fornecem poderosas possibilidades de implementação.

IV.1 Variáveis OPC UA

O servidor OPC UA é criado diretamente pelo *Automation Studio*, uma vez que as variáveis são fornecidas diretamente pelo programa. Para isso, deve-se criar, na pasta *Connectivity*, presente na aba **Configuration View**, um arquivo de configuração OPC UA (OPC UA Default View File).

Além disso, deve-se ativar a comunicação OPC UA do CLP nas configurações do mesmo, que podem ser feitas na **Physical View**, selecionando o CLP com o botão direito e escolhendo *Configuration*. A aba destinada à configuração da rede OPC UA fica localizada no final da lista de opções.

A maneira utilizada para testar a conectividade com o servidor OPC UA foi através do software **UA Expert**, que funciona como um cliente OPC UA, possibilitando a leitura e escrita das variáveis disponíveis no servidor. A Figura 1 exibe a interface do programa.

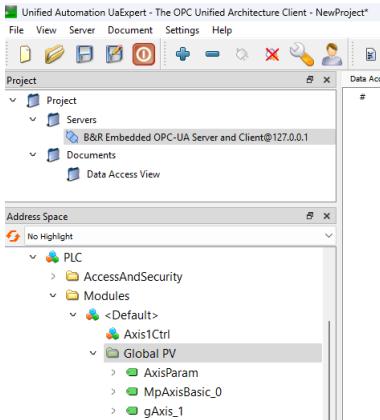


Fig. 1: Tela Inicial do Software UaExpert com a conexão estabelecida e exibindo as variáveis disponíveis no servidor

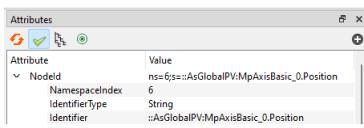


Fig. 2: Janela do Software UaExpert exibindo informações do nó de uma variável disponível no servidor

Para o acesso externo das variáveis disponíveis no servidor OPC UA, é necessário utilizar um endereço denotado como **Node** (Nó). Cada variável do servidor possui um nó único, que contém informações quanto ao seu nome, seu escopo no programa, seu tipo, valor e diversas outras informações. Na Figura 2 é possível visualizar um exemplo de nó, mostrado também pelo programa *UaExpert*, para a variável **Position**. A informação importante se trata do *NodeId*, o identificador que será usado posteriormente para leitura e/ou escrita na variável através de um *script* em Python.

Os tipos de variáveis utilizadas na aplicação foram divididos em quatro categorias diferentes:

- **Parâmetros** - Responsáveis pela parametrização do motor, possuindo variáveis para escrita como velocidade, aceleração, sentido de rotação, etc.
- **Comando** - Variáveis de controle do motor, usadas para acionar e desativar a movimentação do eixo da forma desejada. Serão utilizadas para escrita.
- **Análise** - Valores para análise gráfica da movimentação do motor, como posição e velocidade em tempo real. Usadas para leitura somente.
- **Status** - Informações cruciais para o funcionamento do motor, como mensagens de erro, estado atual (ligado/desligado), entre outras informações.

IV.2 Conexão com o servidor OPC UA e acesso às variáveis

A ideia por trás do método de acesso às variáveis provenientes do CLP é primeiramente armazená-

```

1 import asyncio
2 from asyncua import *
3
4 url = "opc.tcp://192.168.0.1:4840/"
5
6 async def main():
7     async with Client(url=url) as client:
8         # Código do programa
9         print("Hello World")
10
11 if __name__ == "__main__":
12     asyncio.run(main())

```

Fig. 3: Estrutura de código para criação de programa assíncrono com conexão OPC UA

las em um banco de dados através de um *Script* em Python. Para isso, faz-se necessária uma conexão com o servidor OPC UA.

O sistema a ser utilizado se baseia em **um** programação assíncrona. Em programas de execução assíncrona, os comandos são executados simultaneamente ou em concorrência (isto é, linhas de código são parcialmente executadas), de modo a obter uma otimização de tarefas, reduzindo o tempo gasto e tirando maior proveito dos recursos de hardware utilizados [5].

A biblioteca Python *asyncio* fornece todas as ferramentas necessárias para este fim. Outra biblioteca importante se trata da *asyncua*, que contém as funções necessárias para executar a leitura e escrita nos nós do servidor OPC UA.

Para criar um programa assíncrono com conexão com o servidor OPC UA, basta construir uma estrutura como mostrada na Figura (3).

O comando necessário para efetuar a leitura dos nós OPC UA é dado por:

```
n = await client.get_node('NODE_ID').read_value()
```

A variável *n* receberá o valor do nó do servidor OPC UA. O comando *await* que precede a função *get_node* faz parte da biblioteca de programação assíncrona, e significa que a corوتina será paralisada até que função seja devidamente finalizada.

IV.3 Criação e Conexão com o Banco de Dados

A criação das tabelas que serão usadas para armazenar as variáveis do servidor pode ser feita através de um terminal *CLI* do MySQL, porém, de maneira a facilitar a criação e exclusão de tabelas, foi desenvolvido um *script* em Python que automatiza este processo.

A Figura (4) mostra as linhas de código para alcançar o objetivo descrito. A variável *mydb* é um objeto que possui atributos Nome, User, Host e Senha para conectar com o banco. Os demais comandos são basicamente linhas de comando para programação de banco de dados. Com isso, basta executar o script e todas as tabelas desejadas são criadas automaticamente.

Para realizar a conexão com o Banco de Dados MySQL criado, utiliza-se o comando *st.connection* da biblioteca *Streamlit*. Como argumentos, deve-se especificar o tipo do banco. Neste caso, o comando uti-

Fig. 4: Script Python para criação do banco de dados



Fig. 5: Bancada B&R

lizado está representado abaixo. A variável *c* irá ser usada para efetuar *Queries* (pedidos) ao banco através de comandos próprios da linguagem do MySQL. É importante checar se o banco está sendo executado pelo sistema operacional através da janela *Serviços* do Windows, por exemplo.

```
c = st.connection('mysql', type='sql')
```

De modo que esta linha de código seja devidamente executada, é necessário criar um arquivo denominado *secrets.toml* localizado na raiz da biblioteca Streamlit, geralmente no endereço "**C:/Users/[user]/.streamlit/secrets.toml**". Este arquivo possuirá as informações necessárias para a conexão com o banco, tais como endereço de IP, Port, nome do banco, senha, etc.

V ESTRUTURA FÍSICA

Nesta seção serão introduzidos os equipamentos físicos utilizados para confecção do ambiente de testes e aplicação, incluindo dispositivos para controle e para confecção da rede. Uma foto do laboratório pode ser visualizada na **Figura 5**.

V.1 Controlador Lógico Programável X20CP0483

O X20CP0483 faz parte da linha Compact-S PLC da B&R, é um controlador compacto, ideal para uma variedade de aplicações de automação industrial, oferecendo uma combinação de desempenho e economia de espaço. As especificações técnicas deste controlador incluem um processador *ARM Cortex-A9* de 500



Fig. 6: CLP X20CP0483 utilizado na implementação

MHz, 256 MB de RAM e possibilidade de comunicação Ethernet, USB, *POWERLINK* e RS232 [6]. Por se tratar de um CLP modular, também permite a capacidade de expansão por meio de módulos adicionais de entrada e saída digitais/analógicos, de fontes externas e até mesmo interfaces de comunicação.

V.2 Servo Drive ACOPPOS 1016

A família de produtos ACOPOS, também da B&R, simplifica significativamente o processo de parametrização dos componentes, pois grande parte dos parâmetros dos produtos estão incluídos em bibliotecas do software próprio da B&R, o Automation Studio.

O servo drive modelo **ACOPOS 8V1016.00-2** (Figura 7) da B&R é uma solução de controle de movimento de alta performance, projetada para atender às exigências de precisão e eficiência em aplicações industriais avançadas [7]. Este servo drive se destaca por sua capacidade de fornecer controle preciso e dinâmico de motores, contribuindo para a otimização dos processos produtivos.

Também oferece opções de personalização através de módulos plug-in [8]. Na bancada didática utilizada, o servo drive vem equipado com três módulos:

- **8AC114.60-2** - Interface POWERLINK V2 - x2 Ports RJ-45[9]
 - **8AC120.60-1** EnDat 2.1 Encoder: Combinação de um Encoder absoluto e incremental. [10]
 - **8AC130.60-1** Módulo de 8 entradas/saídas digitais [11]

O drive manterá a comunicação com o CLP por meio do seu módulo *8AC114.60-2*, utilizando a rede Ethernet Powerlink, um protocolo aberto para Ethernet padrão. A conexão será por meio das entradas RJ-45 presentes no módulo, e para comunicação ser efetuada com sucesso, é necessário se atentar com a posição das chaves presentes no módulo, mostradas na Figura 8, que indicam o nó da rede POWERLINK a ser utilizado. O valor do nó que está configurado no programa *Automation Studio* deve ser o mesmo indicado pela chave física.

A malha de controle mostrada na Figura 9 exibe como é a implementação do controle de movimento do motor executada pelo servo drive. Pode-se notar que



Fig. 7: Servo Drive ACOPOS 1016 para controle de movimento



Fig. 10: Servo Motor 8LS35E2030.D000-0



Fig. 8: Chave de configuração

se trata de uma metodologia de *Controle em Cascata*, onde uma malha é localizada dentro de outra. Existem essencialmente três variáveis sendo controladas: Posição, Velocidade e Corrente, sendo o *tuning* desta última realizado automaticamente por padrão (auto tuned). A sintonização do controlador de posição só deverá ser feita após a sintonização do controlador interno, ou seja, do controlador da velocidade. O *Feed Forward* se trata de um controle de malha aberta, e só será necessário se a carga acoplada ao eixo possuir uma dinâmica complexa.

V.3 Servo Motor

O motor que será utilizado para testes se trata do modelo **8LS35E2030.D000-0** (Figura 10), um servo motor de três fases e quatro polos, fabricado também pela B&R. Possui uma alimentação de 24Vdc e uma potência de 0.66 kW, podendo atingir uma velocidade máxima de 12.000 RPM. Conhecer as características do motor é um fator importante para a configuração e parametrização na plataforma de automação.

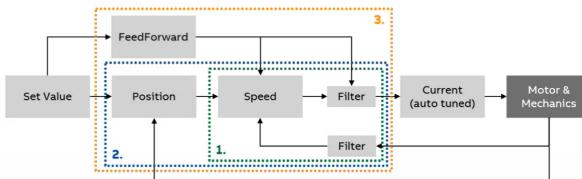


Fig. 9: Malha de controle contendo três controladores: Posição, Velocidade e Corrente (B&R)



Fig. 11: Switch Cisco utilizado na aplicação de controle distribuído

V.4 Switch Cisco Catalyst 2960 WS-C2960-24LT-L

Em uma aplicação de controle distribuído, onde diversos computadores precisam se conectar a um Controlador Lógico Programável, o uso de um switch é fundamental para estabelecer uma rede de comunicação eficiente e robusta.

Um switch é um dispositivo de rede que conecta múltiplos dispositivos, como computadores e CLPs, em uma rede local (LAN). Deste modo, o switch irá permitir que vários computadores e dispositivos de controle se conectem ao CLP simultaneamente, estabelecendo uma rede interligada que facilita a comunicação.

O Cisco Catalyst 2960 WS-C2960-24LT-L (Figura 11) é um switch que possui 24 portas Ethernet 10/100 Mbps e 2 portas Gigabit Ethernet (10/100/1000) para alta velocidade [12]. A ferramenta de gerenciamento fácil para configuração e monitoramento **Cisco Network Assistant** também será utilizada para ajustes iniciais.

Por fim, teremos uma rede composta por 3 computadores, quantidade que pode ser facilmente ajustada com a conexão das portas do Switch. Um diagrama mostrando as conexões e a atribuição dos endereços de IP é mostrada na Figura 12.

VI ENTENDENDO O FUNCIONAMENTO DO MOTOR

As variáveis de programação **utilizadas** são provenientes do sistema *mappMotion* do software *Automation Studio*, e estão relacionadas ao eixo do motor, adicionado no início da aplicação.

Para o teste de implementação da plataforma web, apenas dois objetos foram utilizados para a programação do CLP. O primeiro, do tipo **MpAxisBasic-ParType**, é uma estrutura construída a partir de diversas outras **variáveis**, estas que servem para a parametrização do motor. Variáveis numéricas (Reais, Inteiros ou outros tipos) tais como Posição, Velocidade,

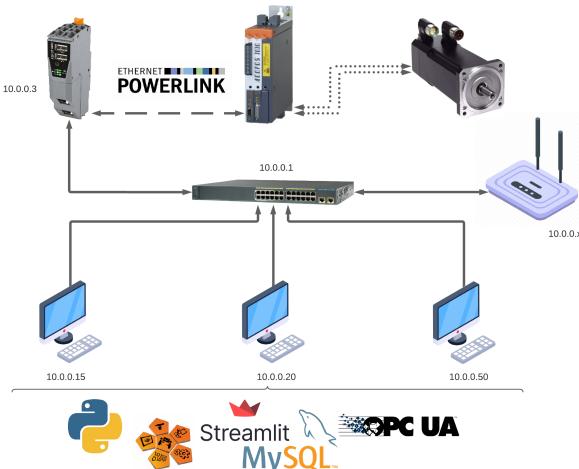


Fig. 12: Diagrama da Rede

| Name | Type | Name | Type |
|----------------|----------------------------|----------------|-------------|
| AxesParam | MpAxisBasicParType | MpAxisBasic_0 | MpAxisBasic |
| Axis | UINT | MpLink | UDINT |
| Distance | LREAL | MpAxis | BOOL |
| Velocity | REAL | ErrorReset | BOOL |
| Acceleration | REAL | Parameters | UDINT |
| Deceleration | REAL | Update | BOOL |
| Position | INT | Power | BOOL |
| Time | DINT | Home | BOOL |
| Homing | MpAxisHomingType | MoveVelocity | BOOL |
| Mode | DINT | MoveAbsolute | BOOL |
| Position | LREAL | MoveAdditive | BOOL |
| Options | MpAxisHomingOptionsType | MoveRelative | BOOL |
| Jog | MpAxisJogType | JogPositive | BOOL |
| Velocity | REAL | JogNegative | BOOL |
| Acceleration | REAL | LimitLoad | BOOL |
| Deceleration | REAL | ReleaseBrake | BOOL |
| Position | MpAxisJogLimitPositionType | Simulate | BOOL |
| Jerk | REAL | AutoTune | BOOL |
| Stop | MpAxisStopType | Active | BOOL |
| Acceleration | REAL | Error | BOOL |
| Deceleration | REAL | ErrorID | DINT |
| Jerk | REAL | UpdateDone | BOOL |
| StopAtPosition | MpAxisStopAtPositionType | Position | LREAL |
| LimitLoad | MpAxisLimitLoadType | Velocity | REAL |
| Load | REAL | CommandAborted | BOOL |
| Direction | DINT | PowerOn | BOOL |
| AutoTune | MpAxisAutoTuneType | IsHomed | BOOL |
| Jerk | REAL | IpSession | BOOL |
| Info | MpAxisBasicInfoType | MoveActive | BOOL |
| Internal | MpComInternalDataType | MoveDone | BOOL |

Fig. 13: Variáveis principais no programa

Aceleração, Desaceleração, Configurações de *Jog*, etc., são todas definidas por meio deste objeto. O segundo objeto, do tipo **MpAxisBasic**, é uma estrutura com duas funcionalidade: primeiro, as operações de controle, portanto em sua estrutura será possível encontrar diversas variáveis *booleans* que dão acesso aos comandos do motor, tais como Ligar/Desligar, Movimentar eixo, Parar, Reiniciar, etc. Nesse objeto também são encontradas algumas variáveis para análise de *Status* e diagnóstico, como por exemplo variáveis que acionam na finalização de um movimento, ou variáveis que indicam erros e seus identificadores correspondentes. Ambas estruturas são exibidas da Figura 13.

VII ESTRUTURA DO BANCO DE DADOS

[WIP] tabelas do banco (**ATUALIZAR**)

VIII ESTRUTURA DA REDE

Como visto na Figura 12, cada elemento da rede possui um endereço IP específico. Os IPs atribuídos estão todos no domínio **10.0.0.X**, de modo que a máscara de sub-rede seja **255.255.255.0**. Para encontrar

| Análise | | Status | |
|----------|-----------|--------------------|----------|
| ID | integer | StatusID | INT(255) |
| Position | FLOAT | CommunicationReady | BOOL |
| Velocity | FLOAT | ReadyToPowerOn | BOOL |
| Time | TIMESTAMP | PowerOn | BOOL |
| | | IsHomed | BOOL |

| Comando | | Parâmetros | |
|--------------|------|-----------------|-----------|
| Enable | BOOL | Position | FLOAT |
| Power | BOOL | Distance | FLOAT |
| ErrorReset | BOOL | Velocity | FLOAT |
| Home | BOOL | Acceleration | FLOAT |
| MoveVelocity | BOOL | Deceleration | FLOAT |
| MoveAbsolute | BOOL | Stop | BOOL |
| MoveAdditive | BOOL | JogPositive | BOOL |
| Stop | BOOL | JogNegative | BOOL |
| JogPositive | BOOL | Time | TIMESTAMP |
| JogNegative | BOOL | JogVelocity | FLOAT |
| | | JogAcceleration | FLOAT |
| | | JogDeceleration | FLOAT |

Fig. 14: Tabelas do Banco de Dados [desatualizado]

o endereço de IP do controlador em modo de configuração de fábrica, um dos métodos a ser adotado é utilizar o software *Wireshark*, que tem como utilidade o monitoramento e análise do tráfego de rede. Os endereços dos computadores foram atribuídos arbitrariamente. A tabela 1 contém os endereçamentos de todos os dispositivos utilizados.

| Dispositivo | Endereço IP |
|-------------------|-------------|
| Controlador (CLP) | 10.0.0.3 |
| Switch | 10.0.0.1 |
| Roteador | 10.0.0.X |
| Computador 1 | 10.0.0.15 |
| Computador 2 | 10.0.0.20 |
| Computador 3 | 10.0.0.50 |

Tabela 1: Atribuição de Endereços de IP para a rede desenvolvida

IX APLICAÇÃO WEB

IX.1 Autenticação (Login)

Ao entrar na plataforma web, o usuário irá se deparar primeiramente com a página de autenticação, onde deverá inserir o nome de usuário e senha, como demonstrado na Figura 15. O registro dos usuários e suas respectivas senhas ficam armazenados em um arquivo de extensão *.yaml*, localizado no mesmo diretório dos arquivos *python* (.py) das páginas. A estrutura deste arquivo contendo as credenciais é exibido na Figura 16.

Vale ressaltar que as senhas podem ser devidamente criptografadas por *hash*, fazendo com que elas não sejam facilmente expostas através do acesso indevido do arquivo, adicionando uma camada de proteção e prevenindo possíveis ataques de segurança.



Fig. 15: Página inicial para Login no sistema

```
pages > ! config.yaml
1 credentials:
2   usernames:
3     Rafael:
4       email: rafaelcpaes@unifei.edu.br
5       name: Rafael Coelho Paes
6       password: 'unifei1913' # Pode ser substituído por hash
7     Jeremias:
8       email: jeremias@unifei.edu.br
9       name: Jeremias Barbosa Machado
10      password: 'unifei1913' # Pode ser substituído por hash
```

Fig. 16: Estrutura do arquivo config.yaml

Em casos em que o usuário tente acessar qualquer outra página do sistema por meio da alteração da *URL* sem estar devidamente logado no sistema, uma mensagem de erro de autenticação será mostrada, de modo que o mesmo seja impedido de visualizar os conteúdos da página. A Figura 17 exibe a mensagem de erro.

A parte do código que rege a autenticação do usuário segue na Figura 18. Todas páginas que exigem a autenticação para serem acessadas deverão conter estas linhas, de modo a proteger contra acesso inadequado. O código basicamente consulta as credenciais disponibilizadas no arquivo *config.yaml* e realiza a autenticação de acordo, também levando em consideração os *cookies* do navegador, de modo a evitar o *log off* indesejado.

IX.2 Painel Lateral

Assim que a autenticação for aprovada, o usuário terá acesso à plataforma web e todas as páginas associadas ao cargo do usuário. Como forma de melhorar a navegação, foi implementada uma barra lateral que disponibiliza todas as páginas disponíveis para o acesso, além de contar um botão para o usuário efetuar um *logout*, caso seja necessário sair do sistema. A Figura 19 exibe a aparência da barra e suas opções de direcionamento.

```
KeyError: 'st.session_state has no key "authentication_status". Did you forget
to initialize it? More info: https://docs.streamlit.io/library/advanced-
features/session-state#initialization'
```

Fig. 17: Erro de autenticação caso usuário esteja deslogado

```
# Autenticação de Usuário
with open('config.yaml') as file:
    config = yaml.load(file, Loader=SafeLoader)

authenticator = stauth Authenticate(
    config['credentials'],
    config['cookie']['name'],
    config['cookie']['key'],
    config['cookie']['expiry_days'],
    config['preauthorized']
)
```

Fig. 18: Código de verificação de autenticação

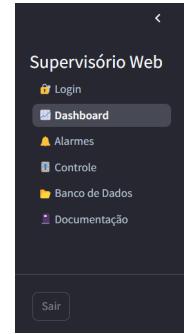


Fig. 19: Painel Lateral para navegação das páginas

IX.3 Dashboard

A página inicial ao qual o usuário será redirecionado ao concluir o *login* será a página de *Dashboard*, onde é possível se ter uma visão geral do processo. No caso da planta estudada, em que se utiliza apenas um motor, é exibido gráficos de posição e velocidade do eixo. Também é possível, através de um botão, testar o funcionamento do motor, fazendo-o avançar 10 graus em sentido horário. A página e sua funcionalidade pode ser vista na Figura 20.

Pelo fato da plataforma web *Streamlit* ser direcionada à ciência de dados, a utilização da mesma se torna uma ferramenta poderosa. A página inicial pode também, para processos de manufatura mais complexos, mostrar informações como número de peças montadas, descartadas, categorizar diferentes produções, exibir informações de diferentes máquinas e partes do processo, etc.

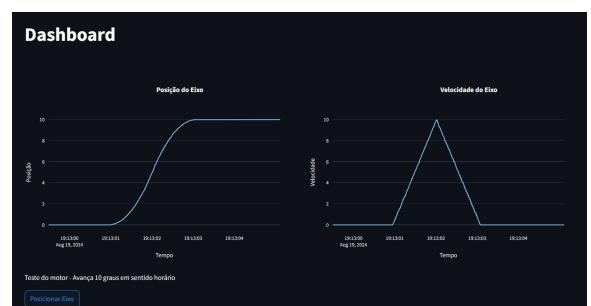


Fig. 20: Dashboard contendo informações importantes do processo

Fig. 21: Sistema de Alarme

Fig. 22: Tipo de Variável para Alarmes

IX.4 Alarmes

É crucial, em todo sistema supervisório, a presença de uma página de alarmes que indiquem ao operador qualquer caso de falha que possa prejudicar o processo. A página de alarmes desenvolvida para a plataforma web tem como função não apenas o reconhecimento dos alarmes, mas também o *reset* do estado de erro e de algumas variáveis que podem contribuir para futuras falhas caso estejam em estados inconsistentes. A Figura 21 exibe a página e suas aplicações.

Para o desenvolvimento desse sistema, foi criada uma *struct* no *Automation Studio*, isto é, um tipo de variável que será usado por todas as variáveis de alarme no programa. A Figura 22 mostra como este *data type* está estruturado.

Basicamente, a estrutura possui quatro outras variáveis: **Status**(Bool) que indicará se o alarme está ativo, **Description** (String) que apresentará uma breve descrição sobre o alarme, **Severity** (Inteiro) que trata da severidade do alarme e **Acknowledgement** (Bool) que indicará o reconhecimento do alarme. Vale dizer que as *Checkboxes* exibidas na tabela da Figura 21 possuem apenas efeito visual, não sendo possível causar uma alteração nas variáveis do CLP. Para isso, será necessário clicar no botão "Reconhecer Alarmes".

IX.5 Controle

A página de Controle pode ser considerada aquela de maior importância para a aplicação. Nesta página, é possível realizar a movimentação do eixo do motor utilizando uma interface gráfica intuitiva.

Basicamente, a plataforma apresenta dois modos de controle de movimento: um para controlar a posição (colocar o eixo em um ângulo específico) e outro para velocidade (fazer o eixo rotacionar indefinidamente em uma dada velocidade), e cada modo está separado em

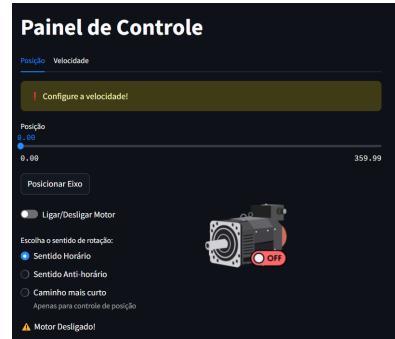


Fig. 23: Página para controle de Posição



Fig. 24: Página para controle de Velocidade

uma *aba* da página, sendo as opções para o sentido da rotação disponíveis logo abaixo. Para utilizar ambos tipos de controle, é necessário primeiramente configurar os parâmetros de *Velocidade*, *Aceleração* e *Desaceleração*, e caso o usuário se esqueça de realizar alguma configuração (ou caso o motor estiver desligado), uma mensagem de aviso será mostrada. Os *sliders* para seleção dos valores estão limitados aos valores máximos dos parâmetros configurados pelo CLP.

As figuras 23 e 24 mostram a interface homem-máquina disponível no Painel de Controle para o controle de posição e velocidade, respectivamente. Ao começar um controle de posição, um novo indicador é exibido mostrando a posição atual do motor. Como outro *feedback* visual, a página apresenta uma figura dinâmica que indica se o motor está propriamente ligado, desligado ou em estado de erro.

IX.6 Banco de Dados

Esta página é projetada para facilitar a consulta ao banco de dados, permitindo o acesso a todas as informações armazenadas de maneira eficiente e organizada. Assim, é possível fornecer uma ferramenta robusta para usuários que precisam verificar dados históricos, seja para análise de problemas ocorridos anteriormente ou para realizar auditorias e monitoramentos contínuos.

Para facilitar a localização de dados específicos, a

| Consulta ao Banco de Dados | | | |
|---|-------|-------|---------------------|
| Selecione o Banco de Dados a ser analisado | | | |
| Análise | | | |
| Data | | | |
| 2024/08/19 | | | |
| Escolha o período dos dados que deseja analisar | | | |
| 08:00 | 18:15 | 23:59 | |
| id Position Velocity Time | | | |
| 595,528 | 10 | 0 | 2024-08-19 19:13:52 |
| 595,529 | 10 | 0 | 2024-08-19 19:13:52 |
| 595,530 | 10 | 0 | 2024-08-19 19:13:52 |
| 595,531 | 10 | 0 | 2024-08-19 19:13:52 |
| 595,532 | 10 | 0 | 2024-08-19 19:13:52 |
| 595,533 | 10 | 0 | 2024-08-19 19:13:52 |
| 595,534 | 10 | 0 | 2024-08-19 19:13:52 |
| 595,535 | 10 | 0 | 2024-08-19 19:13:52 |
| 595,536 | 10 | 0 | 2024-08-19 19:13:52 |
| 595,537 | 10 | 0 | 2024-08-19 19:13:52 |

Fig. 25: Página de consulta ao Banco de Dados

página inclui filtros que permitem aos usuários selecionar a tabela do banco de dados e o período exato em que os dados foram gerados. Os usuários podem definir tanto a data quanto o horário de início e término da consulta. Um exemplo de consulta por tabela e por data é exibido na Figura 25.

IX.7 Documentação

Na área de documentação, será disponibilizado, além de uma breve descrição dos produtos utilizados, o *download* dos manuais de todos os componentes e dispositivos usados na aplicação, de modo que o usuário possa consultar a documentação dos equipamentos e conhecer mais sobre a planta a ser operada.

X ACESSO MOBILE

Devido ao fato da biblioteca *Streamlit* criar aplicativos web *responsivos*, isto é, as páginas de aplicação podem ser renderizadas em uma variedade de dispositivos com tamanhos de janela variados, a utilização de um dispositivo móvel para o acesso da plataforma web se torna possível.

Portanto, é viável controlar atuadores e executar a leitura de sensores diretamente pelo celular, por exemplo. Para isso, basta conectar na mesma rede dos demais equipamentos através do WiFi, que pode ser gerado por um roteador conectado ao *Switch*. Deste modo, torna-se possível acessar pelo navegador o site através do endereço de IP gerado pelo script do *Streamlit*. Este endereço é exibido no terminal assim que o comando de execução é realizado e o aplicativo é devidamente compilado, como exibido na Figura 26.

A Figura 27 exibe uma captura de tela realizada em um dispositivo móvel, onde o acesso da plataforma é realizado através de um navegador.

XI CONCLUSÃO

Neste trabalho foi apresentado o desenvolvimento de um sistema supervisório utilizando ferramentas e

```
PROBLEMS TERMINAL ... powershell + ~ ⌂ ... ^ x
PS C:\Users\Rafael\Documents\UNIFEI\2024.1\TFGWeb> streamlit run
C:\Users\Rafael\Documents\UNIFEI\2024.1\TFG\Login.py
You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.100.103:8501
```

Fig. 26: Endereço de acesso à plataforma



Fig. 27: Acesso mobile da página de controle

recursos de código aberto, que pode ser aplicado com objetivo de redução de custos, eliminando a necessidade de licenças custosas e permitindo a alocação de recursos financeiros para outras áreas críticas do projeto. O sistema proposto se mostrou eficaz, sendo capaz de suprir necessidades fundamentais para um sistema supervisório: visualização limpa (gráficos e visores dinâmicos), segurança (autenticação), armazenamento (banco de dados), etc. No entanto devido à alguns aspectos sua implementação ainda se faz uma solução difícil para o ambiente da automação.

Certos módulos e bibliotecas da linguagem Python exigem um estudo preciso de suas documentações, que muitas vezes podem estar incompletas ou desatualizadas. Outro fator envolvendo o uso de módulos externos é a interoperabilidade entre os mesmos, que muitas vezes podem causar bugs devido à incompatibilidades das bibliotecas.

A biblioteca de interface Web também possui certas limitações para este tipo de aplicação. As funcionalidades de reconhecimento de alarmes, por exemplo, não puderam ser implementadas de maneira ideal, pois a interação do usuário com a tabela não causava o retorno de nenhuma ação no back-end do programa, de modo a impossibilitar o reconhecimento de alarmes individuais.

No quesito de aquisição de dados, os scripts de leitura e escrita no banco de dados se mostraram razavelmente eficazes, de modo a adquirir os valores

das variáveis em intervalos de até 100 milissegundos. Acredita-se que este período possa ser reduzido caso o script Python de aquisição esteja sendo executado em uma máquina dedicada. O fato de utilizar um mesmo computador para efetuar a tarefa de obtenção e de hospedagem da plataforma web resulta em *delays* e *stuttering* na geração de gráficos e disponibilização dos valores na interface.

Como durante o desenvolvimento muitos desafios foram encontrados, e da necessidade de um aprendizado contínuo sobre as ferramentas de código aberto e a integração de diferentes componentes de software e hardware, foram criados dois guias para auxiliar os alunos de automação na criação de um supervisório web e no uso da bancada contendo o controlador, servo-drive e o servo-motor. Todos os códigos e arquivos utilizados (incluindo o guia) estão disponíveis no repositório do *GitHub* no link: https://github.com/Rafael-CP/TFG_Web_Supervisory

Por fim, como sugestão para trabalhos futuros recomenda-se explorar ainda mais recursos da linguagem Python para expandir as possibilidades de criação de sistemas supervisórios, uma vez que a mesma é uma ferramenta rica e em constante desenvolvimento. Por exemplo, bibliotecas e *Frameworks* tais como *Django*, *Flask*, *Pyramid*, etc. são poderosos recursos a serem explorados.

Outras linguagens de desenvolvimento web, com ainda mais recursos, que também são ótimas opções de aplicação. A mais comum e popular é a linguagem *JavaScript*, que possui integração fácil com tecnologias como *NodeJS* e *React*, com principal foco em desenvolvimento de aplicações web (front-end).

Uma sugestão seria a criação de um servidor OPC UA próprio para a comunicação das variáveis da planta de automação, uma vez que neste projeto o servidor foi feito com auxílio de um software proprietário. Desta forma, toda a pirâmide da automação irá englobar tecnologias independentes e livres de licenças proprietárias, de modo à ampliar e democratizar o acesso à área.

REFERÊNCIAS

- [1] Python Software Foundation. History and license, 2024. Acesso em 15 de Abril de 2024. Disponível em: <https://docs.python.org/3/license.html>.
- [2] David Robinson. The incredible growth of python. *The Stack Overflow Blog*, 2017. Acesso em 15 de Abril de 2024. Disponível em: <https://stackoverflow.blog/2017/09/06/incredible-growth-python/>.
- [3] Oracle. Mysql 8.0 reference manual - what is mysql? Acesso em 16 de Abril de 2024. Disponível em: <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>.
- [4] OPC Foundation. Unified architecture. Acesso em 16 de Abril de 2024. Disponível em: <https://opcfoundation.org/about/opc-technologies/opc-ua/>.
- [5] Diogo João Cardoso. Asyncrfj: uma abordagem assíncrona à programação orientada a objeto reativa. *Universidade Federal de Santa Maria Centro de Tecnologia*, 2018. Acesso em 30 de Abril de 2024. Disponível em: https://repositorio.ufsm.br/bitstream/handle/1/16315/DIS_PPGCC_2018_CARDOSO_DIOGO.pdf?sequence=1&isAllowed=y.
- [6] Bernecker & Rainer. *X20 System User's manual*. Acesso em 16 de Novembro de 2023. Disponível em: <https://www.br-automation.com/pt-br/produtos/sistemas-de-controle/sistema-x20/compact-s-plc/x20cp0483/>.
- [7] Bernecker & Rainer. *ACOPOS User's Manual*. Acesso em 5 de Outubro de 2023. Disponível em: <https://www.br-automation.com/pt-br/produtos/motion-control/acopos/servo-drivers/8v101600-2/>.
- [8] Bernecker & Rainer. *8V1016.00-2 Datasheet v1.7*. Acesso em 5 de Outubro de 2023. Disponível em: <https://www.br-automation.com/pt-br/produtos/motion-control/acopos/servo-drivers/8v101600-2/>.
- [9] Bernecker & Rainer. *8AC114.60-1 Datasheet v1.7*. Acesso em 8 de Outubro de 2023. Disponível em: <https://www.br-automation.com/pt-br/produtos/motion-control/acopos/modulos-plug-in/8ac11460-2/>.
- [10] Bernecker & Rainer. *8AC120.60-1 Datasheet v1.6*. Acesso em 8 de Outubro de 2023. Disponível em: <https://www.br-automation.com/pt-br/produtos/motion-control/acopos/modulos-plug-in/8ac12060-1/>.
- [11] Bernecker & Rainer. *8AC130.60-1 Datasheet v1.5*. Acesso em 8 de Outubro de 2023. Disponível em: <https://www.br-automation.com/pt-br/produtos/motion-control/acopos/modulos-plug-in/8ac13060-1/>.
- [12] Cisco Systems Inc. *Switches Cisco Catalyst 2960 Series Documentation*. Acesso em 27 de Março de 2024. Disponível em: https://www.cisco.com/c/pt_br/support/switches/catalyst-2960-series-switches/series.html.