

TRABALHO FINAL DE GRADUAÇÃO – NOVEMBRO/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. 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. Em meio a tendência crescente de integrar equipamentos e objetos da indústria ao conceito de *Internet of Things*, formando a então *IIoT*, os resultados obtidos demonstram a viabilidade e eficácia da solução proposta, fornecendo uma plataforma viável para a automação industrial.

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 impõem 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 comerci-

ais 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.

Como teste prático, utiliza-se uma bancada didática da empresa B&R, presente no Laboratório de Controle de Processos Industriais Contínuos (LCPIC) na Universidade Federal de Itajubá - *Campus Itajubá* contendo um CLP, um Servo Motor e um Drive para controle do movimento.

Deste modo, o presente trabalho pretende demonstrar, como uma prova de conceito, que é possível desenvolver um sistema supervisório utilizando ferramentas livres de direitos autorais. Primeiramente, serão apresentados os equipamentos físicos da bancada de teste. Logo em seguida, serão mostradas as tecnologias e *softwares* utilizados, assim como a forma na qual estes foram estruturados e organizados. Por fim a exibição das funcionalidades da aplicação *web*, que se trata do resultado do projeto como um todo.

A implementação bem-sucedida deste projeto contribuirá para a melhoria da eficiência operacional e a modernização dos processos de controle em ambientes industriais, alinhando-se com as demandas atuais por automação avançada e conectividade na *Indústria 4.0*.

II ESTRUTURA DE HARDWARE

Nesta seção serão introduzidos os equipamentos físicos utilizados na configuração do ambiente de tes-

tes e aplicação, incluindo dispositivos para controle e composição da rede. Uma foto do conjunto de equipamentos utilizados no laboratório pode ser visualizado na Figura 1.



Fig. 1: Bancada B&R localizada no LCPIC

II.1 Controlador Lógico Programável X20CP0483

O Controlador Lógico Programável X20CP0483 (Figura 2) 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 MHz, 256 MB de RAM e possibilidade de comunicação Ethernet, USB, *POWERLINK* e RS232 [1]. Por se tratar de um CLP modular, também possui a capacidade de expansão por meio de módulos adicionais de entrada e saída digitais/analógicos, de fontes externas e até mesmo interfaces de comunicação.



Fig. 2: CLP X20CP0483 utilizado na implementação

II.2 Servo Drive ACOPOS 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 3) 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 [2]. 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 [3]. Na bancada didática utilizada, o servo drive vem equipado com três módulos:

- **8AC114.60-2** - Interface *POWERLINK* V2 - x2 Ports RJ-45[4]
- **8AC120.60-1** - EnDat 2.1 Encoder: Combinação de um Encoder absoluto e incremental. [5]
- **8AC130.60-1** - Módulo de 8 entradas/saídas digitais [6]

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 4, a qual indica 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.



Fig. 3: Servo Drive ACOPOS 1016 para controle de movimento



Fig. 4: Chave de configuração

A malha de controle mostrada na Figura 5 exibe como é a implementação do controle de movimento do motor executada pelo servo drive. Pode-se notar que 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.

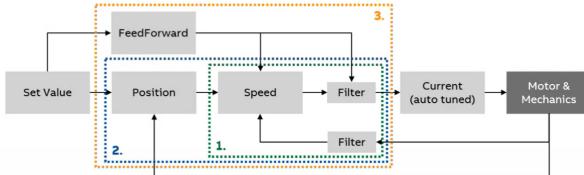


Fig. 5: Malha de controle contendo três controladores: Posição, Velocidade e Corrente (B&R)

II.3 Servo Motor

O motor que será utilizado para testes se trata do modelo **8LS35E2030.D000-0** (Figura 6), 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. 6: Servo Motor 8LS35E2030.D000-0

II.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.



Fig. 7: Switch Cisco utilizado na aplicação de controle distribuído

O Cisco Catalyst 2960 WS-C2960-24LT-L (Figura 7) é um switch que possui 24 portas Ethernet 10/100 Mbps e 2 portas Gigabit Ethernet (10/100/1000) para alta velocidade [7]. A ferramenta de gerenciamento fácil para configuração e monitoramento **Cisco Network Assistant** também será utilizada para ajustes iniciais.

II.5 Controle Distribuído

O controle distribuído ou 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 [8]. 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.

Para a aplicação desta metodologia, foi elaborada uma rede composta por 3 computadores, 1 controlador, 1 switch, proporcionando escalabilidade. Um diagrama mostrando as conexões e a atribuição dos endereços de IP é mostrada na Figura 8.

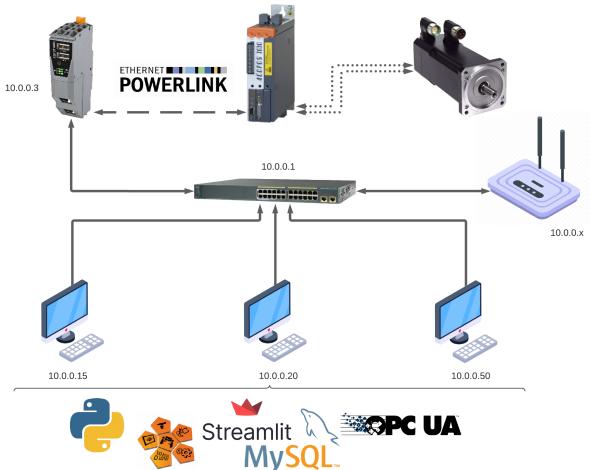


Fig. 8: Diagrama da Rede

III ESTRUTURA DE SOFTWARE

Nesta seção, será apresentada a estrutura de software empregada no desenvolvimento do sistema supervisório web, destacando as principais ferramentas e tecnologias utilizadas. A escolha permitiu garantir a eficiência, escalabilidade e flexibilidade do sistema, permitindo que o mesmo atenda aos requisitos de automação industrial. Serão detalhadas as camadas de

aplicação, a integração com o banco de dados, a comunicação com o hardware e as interfaces de usuário, enfatizando como cada elemento contribui para o desempenho do sistema.

III.1 Automation Studio

Os equipamentos utilizados no desenvolvimento deste trabalho são da fabricante austríaca B&R (Berneker & Rainer), portanto utilizou-se o próprio ambiente de desenvolvimento da fabricante, o *Automation Studio* [9], 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.

III.2 Parâmetros de movimentação do motor

As variáveis de programação utilizadas para o funcionamento do motor 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 que servem para a parametrização do motor. Variáveis numéricas (Reais, Inteiras ou outros tipos) tais como Posição, Velocidade, Aceleração, Desaceleração e Configurações de *Jog* são todas definidas por meio deste objeto. O segundo objeto, do tipo **MpAxisBasic**, é uma estrutura com duas funcionalidade: primeiro, as operações de controle, sendo possível encontrar em sua estrutura diversas variáveis *booleanas* que dão acesso aos comandos do motor, tais como Ligar/Desligar, Movimentar eixo, Parar e Reiniciar. 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 9.

Name	Type	Name	Type
Position	LREAL	MpAxisBasic_0	UINT
Distance	LREAL	Enable	BOOL
Velocity	REAL	ErrorReset	BOOL
Acceleration	REAL	Parameters	UINT
Deceleration	REAL	Update	BOOL
Direction	DINT	Power	BOOL
Homing	MpAxisHomingType	Home	BOOL
Home	MpAxisHomingPositionType	MoveVelocity	BOOL
Mode	LREAL	MoveAbsolute	BOOL
Position	MpAxisJogType	MoveAdditive	BOOL
Jog	MpAxisJogOptionsType	Stop	BOOL
Velocity	REAL	JogPositive	BOOL
Acceleration	REAL	MoveRelative	BOOL
Deceleration	REAL	LimitLoad	BOOL
LimitPosition	MpAxisJogLimPositionType	ReleaseBrake	BOOL
Jerk	REAL	Simulate	BOOL
Deceleration	REAL	AutoTune	BOOL
Jerk	REAL	Active	BOOL
StopAtPosition	MpAxisStopAtPositionType	Error	BOOL
LimitLoad	REAL	StatusID	DINT
Direction	DINT	UpdateDone	UINT
Jerk	REAL	Position	REAL
		Velocity	REAL
		CommandBusy	BOOL
		CommandAborted	BOOL
		PowerOn	BOOL
		IsEnabled	BOOL
		InVelocity	BOOL
		InPosition	BOOL
		Moving	BOOL
		MoveDone	BOOL
		Stopped	BOOL
		LimitLoadReady	BOOL
		BrakeReleased	BOOL
		Info	MpAxisBasicInfoType
		Internal	MpComInternalData

Fig. 9: Variáveis principais no programa

III.3 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 [10], o OPC UA supera as limitações de suas versões anteriores, oferecendo uma série de recursos avançados, incluindo segurança aprimorada, escalabilidade, flexibilidade e capacidade de comunicação em tempo real.

III.4 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*. Cada uma possuindo suas vantagens e aplicações específicas. Para o desenvolvimento da plataforma do Supervisório Web, foi escolhida a biblioteca *Streamlit* [11], biblioteca com potencial de criar aplicativos web interativos de forma intuitiva e com economia de tempo. Além disso, esta é uma biblioteca de código aberto, permitindo uma vasta participação da comunidade de desenvolvedores.

III.5 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.

III.6 Variáveis OPC UA

O servidor OPC UA é criado diretamente pelo *Automation Studio*, uma vez que as variáveis são fornecidas diretamente pelo programa.

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 10 exibe a interface do programa.

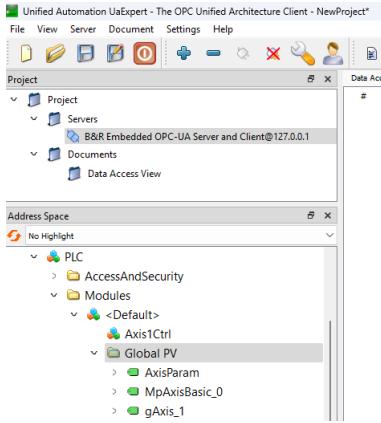


Fig. 10: Tela Inicial do Software UaExpert com a conexão estabelecida e exibindo as variáveis disponíveis no servidor

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 11 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.

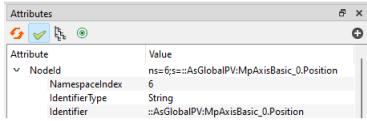


Fig. 11: Janela do Software UaExpert exibindo informações do nó de uma variável disponível no servidor

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 e sentido de rotação.
- **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, es-

tado atual (ligado/desligado), entre outras informações.

III.7 Conexão com o servidor OPC UA e acesso às variáveis

Nesta aplicação decidiu-se por primeiramente armazenar as variáveis provenientes do CLP 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 no método de programação assíncrona. Em programas de execução assíncrona, os comandos são executados simultaneamente ou em concorrência (isto é, enquanto alguma determinada tarefa está sendo executada, outras funcionalidades podem ser executadas em simultâneo), de modo a obter uma otimização de tempo e tirando maior proveito dos recursos de hardware utilizados [12].

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 no Apêndice A.

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 este comando pode continuar sendo executado em paralelo à outros comandos, proporcionando uma aquisição de dados mais fluida.

IV ESTRUTURA DO BANCO DE DADOS

Bancos de dados desempenham um papel crucial no armazenamento e gerenciamento de grandes volumes de dados operacionais e históricos. No caso, o banco de dado **SQL** criado será responsável por registrar informações como parâmetros de movimento, dados dos servomotores, alarmes e capturar dados de entrada e saída de sensores e atuadores.

IV.1 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.

O Apêndice B 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 utilizado 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.

```
1 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 e senha.

O banco de dados foi separado em diversas tabelas (Figura 12), cada uma agrupando variáveis do processo que possuem um propósito específico. A tabela de **Análise**, por exemplo, são variáveis para monitorar a *Posição* e *Velocidade* do motor. **Status** armazena variáveis de estado do motor, como erro e estado de posição. A tabela **Comando** irá possuir variáveis, em sua maioria *booleana*, para enviar comandos relativos à movimentação do eixo. **Parâmetros** são variáveis usadas para parametrizar o motor, com valores de velocidade, aceleração, e distância de rotação desejada. Por fim, **Alarmes** armazena as variáveis de alarme para a segurança do sistema, como estado de alarme, severidade e reconhecimento.

Parâmetros		Status		Comando	
Position	FLOAT	Error	BOOL	Enable	BOOL
Distance	FLOAT	StatusID	INT(255)	Power	BOOL
Velocity	FLOAT	PowerOn	BOOL	ErrorReset	BOOL
Acceleration	FLOAT	IsHomed	BOOL	Home	BOOL
Deceleration	FLOAT	MoveDone	BOOL	MoveVelocity	BOOL
Direction	TINYINT	MoveActive	BOOL	MoveAbsolute	BOOL
JogVelocity	FLOAT	Time	TIMESTAMP(3)	MoveAdditive	BOOL
JogAcceleration	FLOAT			Stop	BOOL
JogDeceleration	FLOAT			JogPositive	BOOL
Time	TIMESTAMP(3)			JogNegative	BOOL
Alarmes					
Descrição		VARCHAR(100)	Severidade	INT	Time
Reconhecimento			BOOL		TIMESTAMP(3)
Time					
Análise					
ID	integer				
Position	FLOAT				
Velocity	FLOAT				
Time	TIMESTAMP(3)				

Fig. 12: Tabelas do Banco de Dados

V ESTRUTURA DA REDE

Como visto na Figura 8, 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 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

VI RESULTADOS OBTIDOS

Nesta seção, serão apresentados os resultados obtidos durante o desenvolvimento e a implementação do sistema supervisório web com os recursos e ferramentas previamente apresentados. O foco da análise será voltado para a aplicação web, que integra todas as tecnologias utilizadas. Portanto, serão discutidos aspectos como o desempenho do sistema, a precisão no controle dos servomotores, a eficiência da interface web para monitoramento em tempo real, além dos benefícios proporcionados pela acessibilidade remota e pela facilidade de integração com outros dispositivos.

VI.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 13. 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 mostrado no Apêndice E.



Fig. 13: Página inicial para Login no sistema

Vale notar 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. Para fazer isso, basta utilizar os comandos do Apêndice D em um terminal, que retornarão a *string* a ser substituída no arquivo.

Cada usuário contido no arquivo *.yaml* possui um atributo denominado *role*: *Aluno*, *Professor* e *Professor Orientador*, que irá determinar quais páginas este poderá acessar na plataforma. Essa verificação é feita na função *main* do arquivo de cada página. Em casos em que o usuário tente acessar uma página 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 14 exibe a mensagem de erro.

```
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. 14: Erro de autenticação caso usuário esteja deslogado

A parte do código que rege a autenticação do usuário é exibido no Apêndice C. 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 levando em consideração os *cookies* do navegador, de modo a evitar o *logoff* indesejado.

VI.2 Painel Lateral

Assim que a autenticação for aprovada, o usuário terá acesso à plataforma web e todas as páginas associadas às credenciais 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 15 exibe a aparência da barra e suas opções de direcionamento.

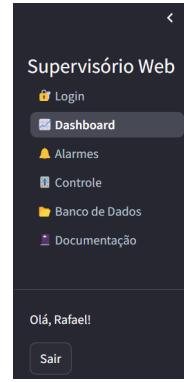


Fig. 15: Painel Lateral para navegação das páginas

VI.3 Dashboard

A página inicial no 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 16.

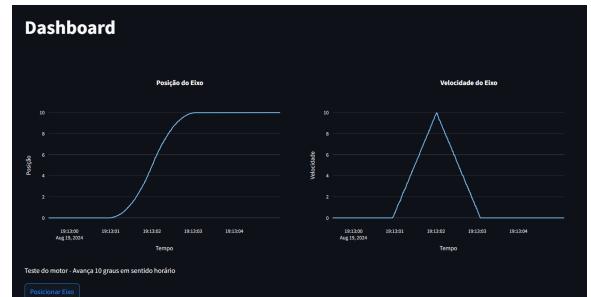


Fig. 16: Dashboard contendo informações importantes do processo

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

VI.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 futu-

ras falhas caso estejam em estados inconsistentes. A Figura 17 exibe a página e suas aplicações.

Descrição	Severidade	Reconhecido?	Time
Motor em estado de erro!	1	<input type="checkbox"/>	2024-08-14 15:48:24
Motor em estado de erro!	1	<input checked="" type="checkbox"/>	2024-08-06 19:31:39
Motor em estado de erro!	1	<input checked="" type="checkbox"/>	2024-08-06 19:25:36
Motor em estado de erro!	1	<input checked="" type="checkbox"/>	2024-08-06 19:25:22
Motor em estado de erro!	1	<input checked="" type="checkbox"/>	2024-08-06 19:21:30

Buttons:

- Resetar Estado de Erro
- Resetar todas as variáveis
- Reconhecer Alarms

Fig. 17: Sistema de Alarme

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 18 mostra como este *data type* está estruturado.

Name	Type
Alarm	STRUCT
Status	BOOL
Description	VSTRING[80]
Severity	UINT
Acknowledgement	BOOL

Fig. 18: Tipo de Variável para Alarmes

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 17 possuem apenas efeito visual, não sendo possível causar uma alteração nas variáveis do CLP. Para isso, será necessário clicar no botão "Reconhecer Alarms".

VI.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 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 19 e 20 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.



Fig. 19: Página para controle de Posição

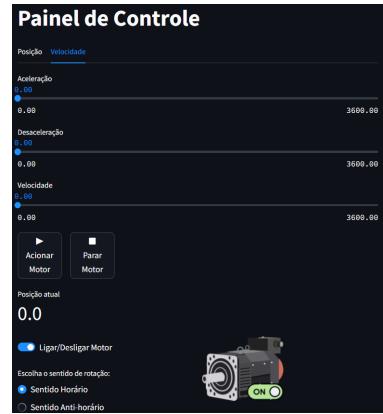


Fig. 20: Página para controle de Velocidade

VI.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 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 21.



Fig. 21: Página de consulta ao Banco de Dados

VI.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.

VI.8 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 por celular ou por algum outro dispositivo móvel industrial, que vêm sendo muito presente com o crescimento da *Indústria 4.0* e a *Industrial Internet of Things*, 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 22.

A Figura 23 exibe uma captura de tela realizada em um dispositivo móvel, onde o acesso da plataforma é realizado através de um navegador.

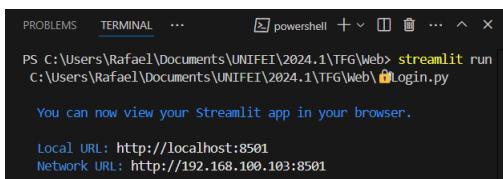


Fig. 22: Endereço de acesso à plataforma

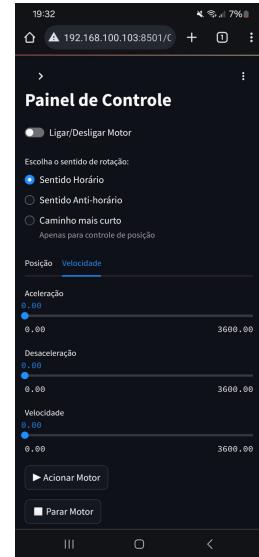


Fig. 23: Acesso mobile da página de controle

VII CONCLUSÃO

Neste trabalho foi apresentado o desenvolvimento de um sistema supervisório utilizando ferramentas e 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) e armazenamento (banco de dados).

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 estes, 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 razoavelmente 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.

Durante o desenvolvimento do projeto muitos desafios foram encontrados, como por exemplo a necessidade de um aprendizado contínuo sobre as ferramentas aplicadas e suas devidas integrações com diferentes componentes de software e hardware. Assim, 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* 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).

Este projeto pode ser considerado relevante no cenário atual de crescimento da Indústria 4.0 e da Internet Industrial das Coisas (IIoT). À medida que as empresas se tornam mais conectadas e inteligentes, a demanda por soluções de monitoramento e controle em tempo real, acessíveis de qualquer lugar, aumenta significativamente, sendo este presente trabalho uma solução viável para este fim.

REFERÊNCIAS

- [1] 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/>.
- [2] 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/>.
- [3] 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/>.
- [4] 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/>.
- [5] 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/>.
- [6] 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/>.
- [7] 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.
- [8] Cristina Toshie Motohashi Matsusaki. Modelagem de sistemas de controle distribuídos e colaborativos de sistemas produtivos. *Escola Politécnica, Universidade de São Paulo*, 2004. Disponível em: <https://doi.org/10.11606/T.3.2004.tde-20122004-112454>.
- [9] Bernecker & Rainer. Industrial automation: Br - intelligent automation for adaptive manufacturing. Disponível em: <https://www.br-automation.com/en/products/software/>.
- [10] OPC Foundation. Unified architecture. Acesso em 16 de Abril de 2024. Disponível em: <https://opcfoundation.org/about/opc-technologies/opc-ua/>.
- [11] Streamlit. A faster way to build and share data apps. Disponível em: <https://streamlit.io/>.
- [12] 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.

BIOGRAFIA



Rafael Coelho Paes - Nasceu em Rio de Janeiro (RJ), em 2000. Viveu e estudou em Itajubá (MG), ingressando no curso de Engenharia de Controle e Automação na UNIFEI em 2019, onde atuou como monitor de *Automação e Supervisórios* durante o ano de 2022. Em 2023, foi bolsista pelo programa *Global Korea Scholarship* e atualmente realiza estágio no *Centro Nacional de Pesquisa em Energia e Materiais*.

Apêndice

A CÓDIGO PARA CRIAÇÃO DE UM PROGRAMA ASSÍNCRONO

```
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())
```

B CÓDIGO PARA CRIAÇÃO DO BANCO DE DADOS SQL E TABELAS

```
1 import mysql.connector
2
3 mydb = mysql.connector.connect(
4     host="localhost",
5     user="root",
6     password="12345",
7     database="opc_ua_tfg" # 'opc_ua_tfg' na CLI
8 )
9
10 mycursor = mydb.cursor()
11
12 mycursor.execute(\"CREATE DATABASE OPC_UA_TFG\"
13     ) # Se banco ainda não estiver criado
14
15 # Cria as tabelas para as variáveis desejadas
16 mycursor.execute("CREATE TABLE Parametros (
17     Position FLOAT, Distance FLOAT, Velocity
18     FLOAT, Acceleration FLOAT, Deceleration
19     FLOAT, Direction TINYINT, JogVelocity
20     FLOAT, JogAcceleration FLOAT,
21     JogDeceleration FLOAT, Time TIMESTAMP(3))
22 ")
23
24 mycursor.execute("CREATE TABLE Comando (Enable
25     BOOL, Power BOOL, ErrorReset BOOL, Home
26     BOOL, MoveVelocity BOOL, MoveAbsolute BOOL
27     , MoveAdditive BOOL, Stop BOOL,
28     JogPositive BOOL, JogNegative BOOL, Time
29     TIMESTAMP(3))")
30
31 mycursor.execute("CREATE TABLE Analise (id INT
32     AUTO_INCREMENT PRIMARY KEY, Position
33     FLOAT(3), Velocity Float(3), Time DATETIME
34     (3))")
35
36 mycursor.execute("CREATE TABLE Status (Error
37     BOOL, StatusID INT, PowerOn BOOL, IsHomed
38     BOOL, MoveDone BOOL, MoveActive BOOL, Time
39     TIMESTAMP(3))")
40
41 mycursor.execute("CREATE TABLE Alarmes (
42     Descricao VARCHAR(100), Severidade INT,
43     Reconhecimento BOOL, Time TIMESTAMP(3))")
44
45 #mycursor.execute("DROP TABLE Parametros,
46     Comando, Analise, Status, Alarmes") # Caso
47     # necessário dropar todas as tabelas
```

C CÓDIGO PARA AUTENTICAÇÃO DE USUÁRIO

```
1 import streamlit_authenticator as stauth
2 import yaml
3 from yaml.loader import SafeLoader
4
5 if st.session_state["authentication_status"]
6     is None:
7     st.switch_page("Login.py")
8
9 with open('config.yaml') as file:
10     config = yaml.load(file, Loader=SafeLoader
11 )
12
13 authenticator = stauth Authenticate(
14     config['credentials'],
15     config['cookie']['name'],
16     config['cookie']['key'],
17     config['cookie']['expiry_days'],
18     config['preauthorized']
19 )
```

D COMANDOS PARA CRIPTOGRAFAR SENHA EM HASH

```
1 >>> from streamlit_authenticator.utilities.
2     hasher import Hasher
3 >>> hashed_password = Hasher(['123']).
4     generate()
5 >>> print(hashed_password)
6 ['$2b$12$YASdy0n4t31wC6BJCd5KCulHT.
7 EtjlqitLoxpRSH6oq.G8DGM3uiy']
```

E CÓDIGO PARA CREDENCIAIS DE USUÁRIO

```
1 credentials:
2     usernames:
3         Rafael:
4             email: rafaelcpaes@unifei.edu.br
5             name: Rafael Coelho Paes
6             password: '$2b$12$YASdy0n4t31wC6BJCd5KCulHT.
7 EtjlqitLoxpRSH6oq.G8DGM3uiy'
8             role: 'Aluno'
9             # Senha Segura! :)
```

```
10        Jeremias:
11             email: jeremias@unifei.edu.br
12             name: Jeremias Barbosa Machado
13             password: '123' # Senha exposta! :0
14             role: 'Professor Orientador'
```

```
15        Lenarth:
16             email: lenarth@unifei.edu.br
17             name: Luiz Lenarth Gabriel Vermaas
18             password: '123' # Senha exposta! :0
19             role: 'Professor'
```

```
20        Waldecir:
21             email: waldecir.souza@unifei.edu.br
22             name: Carlos Waldecir de Souza
23             password: '123' # Senha exposta! :0
24             role: 'Professor'
```

```
25        cookie:
26             expiry_days: 0
27             key: random_signature_key # Must be string
28             name: random_cookie_name
29
30        preauthorized:
31             emails:
32             - test@gmail.com
```