UNIVERSIDADE FEDERAL DE ALAGOAS-UFAL
CAMPUS A.C. SIMÕES
ENGENHARIA DE COMPUTAÇÃO

GLAUBER DE ARRUDA BRAGA

INTEROPERABILIDADE ENTRE CONTROLADORES INDUSTRIAIS E FACTORY I/O ATRAVÉS DE UM SERVIDOR OPC UA

MACEIÓ
2023

Glauber de Arruda Braga

INTEROPERABILIDADE ENTRE CONTROLADORES INDUSTRIAIS E FACTORY I/O ATRAVÉS DE UM SERVIDOR OPC UA

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação da Universidade Federal de Alagoas - UFAL, Campus A.C. Simões.

Orientador: Prof. João Raphael Souza Martins

Maceió
2023

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecária: Taciana Sousa dos Santos – CRB-4 – 2062

B813i Braga, Glauber de Arruda.
Interoperabilidade entre controladores industriais e Factory I/O através de um servidor OPC UA / Glauber de Arruda Braga. – 2023.
61 f. : il. color.

Orientador: João Raphael Souza Martins.
Monografia (Trabalho de Conclusão de Curso em Engenharia da Computação) – Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2023.

Bibliografia: f. 60-61.

1. Automação industrial. 2. OPC UA. 3. CLP. 4. Factoy IO (Software). I. Título.

CDU: 004.4

Glauber de Arruda Braga

INTEROPERABILIDADE ENTRE CONTROLADORES INDUSTRIAIS E FACTORY I/O ATRAVÉS DE UM SERVIDOR OPC UA

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação da Universidade Federal de Alagoas - UFAL, Campus A.C. Simões.

Data de Aprovação: 26/10/2023

Banca Examinadora

Prof. João Raphael Souza Martins
Universidade Federal de Alagoas
Campus A.C. Simões
Orientador

Prof. Thiago D. Cordeiro
Universidade Federal de Alagoas
Campus A.C. Simões
Examinador

Prof. Tiago Alves de Almeida
Universidade Federal de Alagoas
Campus A.C. Simões
Examinador

RESUMO

O uso de computadores e softwares nas indústrias e sistemas de automação industrial teve um grande avanço na década de 1990. No entanto, a integração desses sistemas nem sempre é fácil, pois eles são compostos por componentes de diferentes fabricantes com protocolos proprietários. Para solucionar esse problema, foi criado o padrão OPC UA (Open Platform Communication Unified Architecture), que é uma tecnologia aberta e independente de plataforma para a integração de sistemas de automação industrial. Este trabalho apresenta um estudo sobre a integração de sistemas de automação industrial utilizando o padrão OPC UA. O trabalho apresenta uma metodologia para configurar uma rede através de um controlador CLP que se comunica com softwares que não possuem o mesmo protocolo de comunicação nativo, o *Factory IO* e o SIMULINK. A integração é realizada através do padrão OPC UA, realizando a troca de informações e de comandos. Os resultados mostraram que a integração de sistemas de automação industrial utilizando o padrão OPC UA é uma solução eficaz para os problemas de interoperabilidade, pois ele permite a troca de dados e comandos entre sistemas de diferentes fabricantes e protocolos, de forma segura e eficiente.

Palavras-chave: Automação industrial, OPC UA, CLP, Factory IO.

ABSTRACT

The use of computers and software in industrial automation systems has made significant advances since the 1990s. However, the integration of these systems is not always straightforward, as they are often composed of components from different manufacturers with proprietary protocols. To address this issue, the OPC UA (Open Platform Communication Unified Architecture) standard was developed. OPC UA is an open and platform-independent technology for the integration of industrial automation systems. This work presents a study on the integration of industrial automation systems using the OPC UA standard. The work presents a methodology for configuring a network through a PLC controller that communicates with software that does not have the same native communication protocol, Factory IO and SIMULINK. The integration is performed through the OPC UA standard, exchanging information and commands. The results showed that the integration of industrial automation systems using the OPC UA standard is an effective solution to interoperability problems, as it allows the exchange of data and commands between systems from different manufacturers and protocols in a secure and efficient manner.

Keywords: Industrial automation, OPC UA, PLC, Factory IO.

LISTA DE FIGURAS

Sumário

1 INTRODUÇÃO

O uso dos computadores e softwares nas indústrias e sistemas de automação industrial teve um grande avanço na década de 90, já que eles permitiram o controle e a visualização de processos de forma mais eficiente e integrada. Entretanto essa integração nem sempre funciona de forma fácil já que esses sistemas, na maioria da vezes, são compostos por componentes de diferentes fabricantes (sensores, atuadores, controladores, redes de comunicação, softwares e interfaces) e possuem protocolos proprietários que não funcionam entre si. A integração de sistemas requer uma análise cuidadosa das necessidades e características de cada processo, bem como a escolha das tecnologias mais adequadas e compatíveis.

Para tentar solucionar esse problema, em meados de 1996 surgiu um esforço para criação de um padrão de comunicação para os softwares de automação industrial, que resultou no OPC DA (Open Platform Communication Data Access) e na OPC Foundation, uma organização com objetivo de manter e desenvolver o padrão. Em 2006 surgiu o OPC UA (*Unified Architecture*), uma evolução do agora chamado OPC Classic, que usa um modelo de informação baseado em objetos e que pode ser estendido para representar dados complexos e específicos. Ele também oferece mecanismos de segurança, como criptografia e autenticação, para proteger a transmissão de dados. OPC UA é uma tecnologia aberta e independente de plataforma, que pode ser implementada em diferentes arquiteturas e protocolos de rede.

Este trabalho tem como intuito criar uma rede industrial com softwares que não possuem o mesmo protocolo de comunicação e integra-los através do OPC UA. A rede criada consiste em uma planta de controle de tanques simulada no Factory IO e seu controle PID é realizado através do Simulink com toda a comunicação feita através do protocolo OPC UA. Por fim comparamos seu desempenho com uma rede industrial onde o controle é realizado diretamente por um CLP.

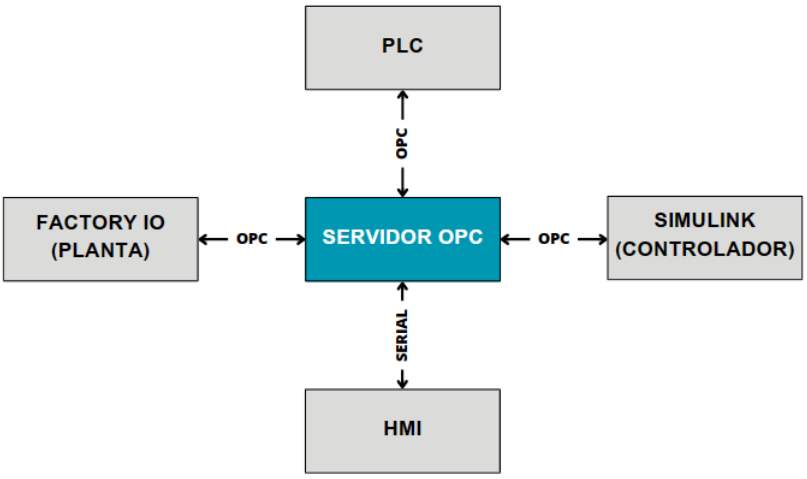
2 OBJETIVOS

Neste tópico serão discutidos os objetivos gerais e específicos do projeto, dando um panorama geral dos passos necessários para atingir o objetivo proposto.

2.1 Objetivos gerais

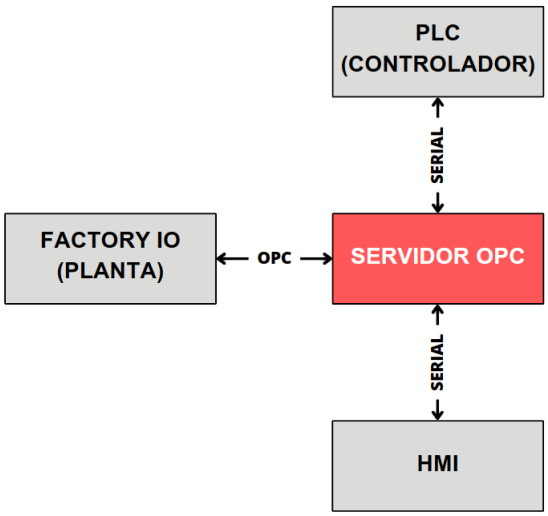
Construir uma rede industrial para realizar o controle PID através do Simulink de uma planta no Factory IO contendo um sistema de tanque, onde toda a comunicação é feita através do protocolo OPC UA (figura 2). No final, iremos comparar seu funcionamento com o controle sendo realizado diretamente pelo CLP (figura 1).

Figura 1 – Captura de tela do diagrama de funcionamento do sistema 2



Fonte: Autoria própria.

Figura 2 – Captura de tela do diagrama de funcionamento do sistema 1



Fonte: Autoria própria.

2.2 Objetivos específicos

I. Estudar as ferramentas necessárias para habilitar a comunicação entre um CLP fabricado pela Siemens da linha SIMATIC e uma planta simulada utilizando o software Factory IO realizando o controle PID desta através do MATLAB, onde toda a comunicação é feita através do protocolo OPC UA;

II. Configurar o PLCSim Advanced V3.0 da Siemens para simular o CLP de CPU S7-1516-3 PN;

III. Configurar o Factory IO para se comunicar com o servidor OPC e com o controlador;

IV. Configurar a simulação do PLC como um servidor OPC UA através do TIA Portal V16;

V. Desenvolver um script no MATLAB para realizar sua conexão com o servidor OPC;

VI. Configurar o controlador PID utilizando o TIA Portal V16 e realizar o controle da planta no Factory IO;

VII. Configurar o PID com os mesmos parâmetros utilizados anteriormente, desta vez utilizando o Simulink conectado através do uso do protocolo OPC UA para realizar o controle da planta no Factory IO;

VIII. Comparar o desempenho do controle realizado diretamente do TIA Portal V16 com o do Simulink através do protocolo OPC UA.

3 FUNDAMENTAÇÃO TEÓRICA

Como definimos anteriormente, o objetivo deste trabalho é apresentar uma proposta de configuração de uma rede que utiliza um controlador CLP para interagir com softwares que operam com protocolos de comunicação diferentes. Para isso, empregamos o padrão OPC UA. Através de testes e análises, verificaremos o desempenho e as vantagens do sistema proposto em relação a um sistema que usa um protocolo proprietário.

Com esses objetivos bem definidos, podemos proceder à análise das ferramentas que serão empregadas no desenvolvimento do trabalho. Nesta seção, descreveremos o funcionamento de cada uma delas, avaliaremos seu desempenho em cenários gerais e examinaremos sua aplicação em casos específicos.

3.1 OPC

O avanço da informática e dos softwares aplicados à automação industrial foi marcante na década de 90, com a difusão dos PCs baseados no sistema operacional Windows para o monitoramento e controle de processos. Diante desse cenário, surgiram diversas iniciativas para o estabelecimento de um padrão nos softwares de automação e na comunicação dos dados. Em meados de 1996, foi elaborada uma especificação para o OPC DA (Open Platform Communication Data Access) e fundada a OPC Foundation, uma entidade responsável por manter e desenvolver o padrão. O OPC DA é um padrão aberto e independente de plataforma, que possibilita um fluxo contínuo de informações entre dispositivos de diferentes fabricantes.

Figura 3 – Logo da OPC Foundation

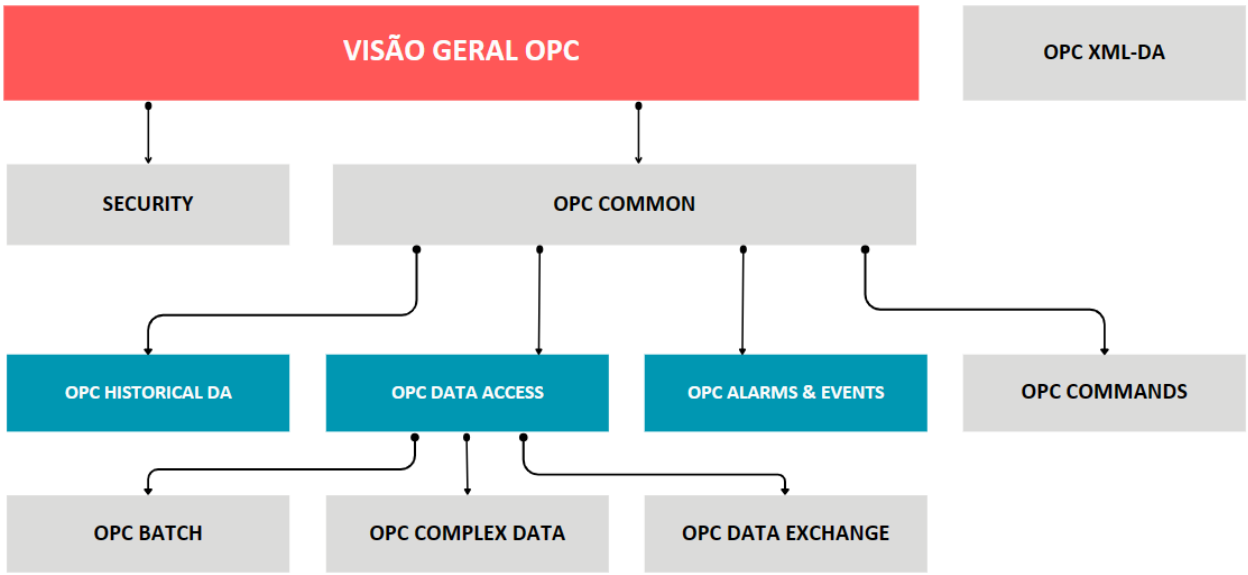
Fonte: Site OPC Foundation (<https://opcfoundation.org/>, acessado em: JULHO de 2023)

O OPC DA surgiu como uma solução para padronizar a comunicação entre os CLP e os sistemas de supervisão, independente dos protocolos específicos de cada fabricante, tais como Modbus, DeviceNet e HART. A primeira versão do OPC utilizava as tecnologias COM e DCOM, da Microsoft, para permitir a troca de dados entre componentes de software distribuídos em uma rede. Atualmente essa versão ficou conhecida como OPC Classic e tem como limitação a dependência do sistema operacional Windows.

3.1.1 OPC Classic

A especificação OPC *Classic* é dividida em 3 definições separadas: OPC DA (*Data Access*) que define a troca de dados, incluindo valores, tempo e informação de qualidade, OPC AE (*Alarms and Events*) define a troca de mensagens do tipo alarme e eventos, além de variáveis de estados, e OPC HDA (*Historical Data Access*) que define métodos de consulta e análises que podem ser aplicados a dados históricos com registro de data e hora. Na figura (4) podemos ter uma visão geral da padronização do OPC *Classic*.

Figura 4 – Padrões da interface OPC Classic



Fonte: Baseado na figura encontrada em BURKE; IWANITZ; LANGE, 2010

O OPC *Classic* utiliza a arquitetura cliente – servidor para a troca de informações, onde o servidor encapsula as informações de processo e disponibiliza na sua interface, e o cliente conectado pode acessar e consumir os dados oferecidos. As aplicações consomem e fornecem dados que podem ser tanto do cliente quanto do servidor.

Por utilizar as tecnologias proprietárias da Microsoft o software era totalmente dependente do SO Windows, além disso haviam inúmeros problemas com o uso da DCOM na comunicação remota com o OPC.

3.1.2 OPC Unified Architecture (UA)

De acordo com (BURKE;IWANITZ;LANGE,2010), o OPC Unified Architecture foi criado a partir da necessidade de encontrar um substituto para tudo que era baseado nas especificações COM, sem perder nenhuma de suas características ou seu desempenho. Alguns fatores que motivaram o desenvolvimento da nova geração foram:

- Limitações da tecnologia DCOM;

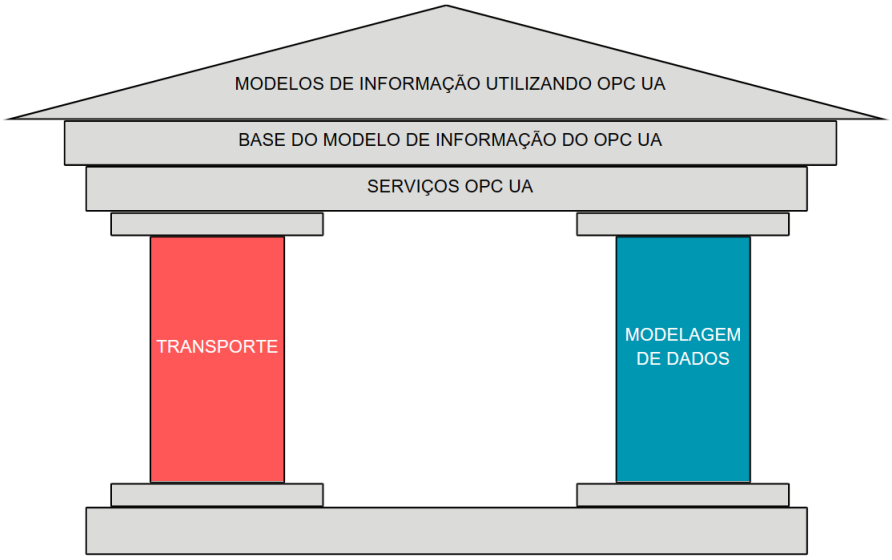
- Necessidade de descontinuar o uso de COM/DCOM;
- Acabar com a exclusividade da plataforma Windows;
- Falta de proteção contra acessos não autorizados.

Construído em cima do OPC *Classic*, o OPC UA foi lançado em 2008 com o intuito de ser uma plataforma orientada a serviços independentes que entregasse todas as funcionalidades do seu antecessor em uma estrutura extensível. Com isso, alguns objetivos e requerimentos foram descritos:

- Equivalência funcional das funções COM do OPC *Classic* devem ser mapeadas;
- Independência de plataforma, devendo funcionar em todos os tipos de dispositivos, desde microcontroladores embarcados, até grandes infraestruturas de rede;
- Segurança e limitação de acesso;
- Ser extensível, podendo ter novas funcionalidades adicionadas sem afetar o funcionamento das aplicações existentes;
- Fácil modelagem das informações para facilitar a definição de informações complexas.

Para atingir esses objetivos, o OPC UA foi construído em cima de algumas camadas (figura 5) e possui duas bases fundamentais: O componente mecanismo de transporte e o componente modelagem de dados.

Figura 5 – Camadas da interface OPC UA



Fonte: Baseado na figura encontrada em BURKE; IWANITZ; LANGE, 2010

O componente de transporte oferece vários mecanismos adequados para diferentes situações, empregando o protocolo TCP para uma comunicação intranet de alto desempenho e

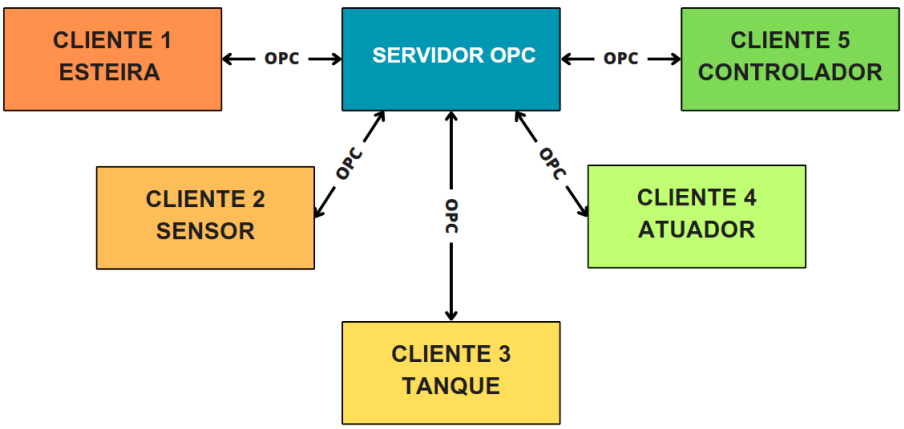
para compatibilizar outros padrões web, como o HTTP e XML. O componente modelagem de dados estabelece as normas e os elementos fundamentais para elaborar um modelo de informação com OPC UA. Esse componente também é encarregado de definir os pontos de acesso no espaço de endereçamento e os tipos básicos utilizados para compor uma hierarquia de tipos.

Uma forma de descrever os serviços UA é como uma camada intermediária entre servidores, que atuam como provedores de informações para os clientes, que são os "usuários"de informação. Eles facilitam a troca de dados entre os clientes e o servidor, permitindo que o cliente OPC UA acesse apenas a informação relevante, sem necessidade de compreender todo o modelo. As especificações do padrão OPC UA são divididas em diferentes partes, seguindo a normatização IEC, e são reconhecidas pela IEC 62541.

O padrão OPC UA é um modelo flexível e extensível, que se adapta aos requisitos da indústria 4.0. Ele é composto por 13 partes, das quais as sete primeiras definem os aspectos essenciais do protocolo, tais como o modelo de segurança, o espaço de endereçamento, os serviços, o modelo de informação dos dados e outros. As partes restantes correspondem a implementações anteriores do OPC *Classic*, abrangendo o acesso de dados, os alarmes e o histórico. Além disso, o padrão UA permite a parte expansível, que pode ser personalizada com informações relevantes para o usuário.

Uma das vantagens de se adotar o padrão OPC UA é a capacidade de integrar diferentes dispositivos e plataformas em uma rede única e segura, otimizando o acesso aos dados e a supervisão dos processos. Podemos ver um exemplo de sua expansividade na figura (6). Em muitos casos, os Controladores Lógicos Programáveis, também chamados de CLP, são configurados para atuarem como um servidor OPC UA. Vários fabricantes como a Siemens já permite o seu uso de forma nativa, possibilitando a configuração de um servidor de forma simples e direta. Desse modo, padrão OPC UA e o CLP podem ser vistos como elementos complementares na automação industrial, que contribuem para aumentar a eficiência, a flexibilidade e a confiabilidade dos sistemas.

Figura 6 – Exemplos de uma rede industrial conectada via OPC UA



Fonte: Autoria própria.

3.2 Controlador Lógico Programável (CLP)

A sigla CLP significa Controlador Lógico Programável, que corresponde a *Programmable Logic Controller* (PLC). O CLP é um dispositivo computacional que pode ser programado para simular o comportamento de um diagrama elétrico em linguagem Ladder. Ele possui maior robustez que um computador convencional e é empregado como controlador para automação industrial. Os CLPs surgiram como uma evolução da tecnologia que era empregada anteriormente, os relés conectados por fios (*hard-wired relay and timer logic control systems*).

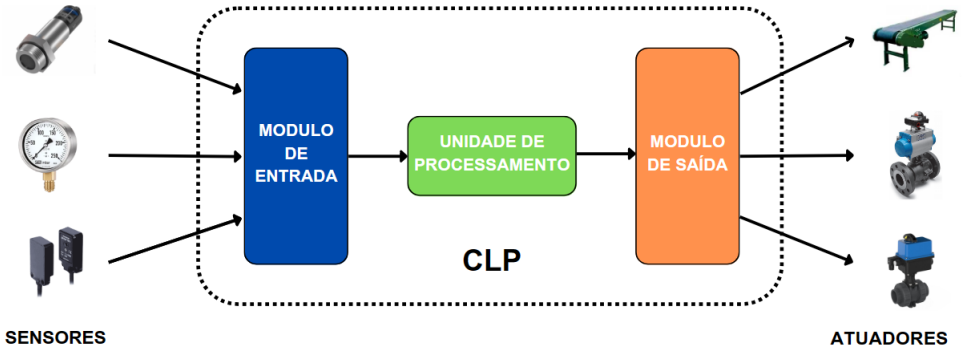
Figura 7 – Diferentes modelos de CLP

Fonte: Site Siemens (<https://www.siemens.com/>, acessado em: JULHO de 2023)

Os controladores lógicos programáveis (CLPs) são dispositivos eletrônicos que permitem a execução de programas para o controle de processos industriais. Esses dispositivos apresentam alta confiabilidade e eficiência em aplicações que envolvem controle sequencial, sincronização e temporização de eventos, sendo amplamente empregados como o elemento central de um sistema automatizado. Além disso, os CLPs reduzem a necessidade de fiação e simplificam a montagem do circuito, facilitando a manutenção e a modificação do sistema. O termo “lógico” se refere à forma de programação dos CLPs, que se baseia na implementação de funções lógicas e operações de comutação.

O Controlador Lógico Programável (CLP) é composto por três partes fundamentais: entradas, saídas e unidade de processamento. As entradas correspondem a dispositivos como chaves e sensores, que fornecem informações sobre o estado do sistema. As saídas são dispositivos como motores e bombas, que executam as ações de controle. As entradas e saídas são conectadas ao CLP e, em seguida, o controlador as monitora e as manipula de acordo com o programa definido.

Figura 8 – Divisão básica de um CLP



Fonte: Autoria própria.

Os CLPs surgiram na indústria automotiva na década de 1960, quando os equipamentos automatizados eram controlados por circuitos fixos que utilizavam relés e fios. A General Motors (GM) então elaborou as especificações para um controlador programável que pudesse substituir essas tecnologias. O aspecto fundamental dessa ideia era desenvolver uma linguagem de programação que expressasse o diagrama esquemático dos relés, com as entradas representadas por contatos de relé e as saídas por bobinas de relé. Ao usar essa linguagem de programação, a representação ficava quase idêntica ao circuito necessário para controlar a planta. Essa programação se assemelhava aos degraus de uma escada, por isso recebeu o nome de Diagrama Ladder (escada).

Em 1969 Dick Morley concebeu o primeiro controlador programável e a empresa em que trabalhava desenvolveu o primeiro CLP a ser comercializado, o MODICON 084, que foi instalado numa divisão da GM. Inicialmente ele não era muito robusto, podendo realizar apenas controle On-Off, o que limitou muito o seu uso. Com o passar dos anos, diversas inovações e avanços nas tecnologias de microcontroladores foram surgindo, o que permitiu que funcionalidades mais completas e robustas fossem adicionadas ao CLP. Hoje existem diversos fabricantes de CLP, onde podemos citar a Siemens e Rockwell como as maiores do setor.

Figura 9 – MODICON 084, primeiro CLP comercializado

Fonte: Site Blog Schneider Electric (<https://blog.se.com/br/>, acessado em: JULHO de 2023)

Para atender todas as necessidades da comunidade industrial foi formado um grupo dentro da IEC (International Electrotechnical Commission) para avaliar o projeto completo dos CLPs, criando padrões que vão desde o hardware, até a comunicação. Essa força tarefa deu origem a norma IEC 61131 que regulariza o projeto dos controladores lógicos programáveis em 8 partes:

1. **IEC 61131-1**: Informações gerais, como definição de terminologia, conceitos e definição do hardware;
2. **IEC 61131-2**: Requisitos e testes para a verificação e fabricação;
3. **IEC 61131-3**: Linguagens de programação, definindo um padrão para a estrutura de software do CLP, suas linguagens e a forma de execução dos programas;
4. **IEC 61131-4**: Guia do usuário, possuindo orientações para instalação e manutenção de CLPS;
5. **IEC 61131-5**: Definições para a comunicação com outros dispositivos;
6. **IEC 61131-6**: Definições para a comunicação com outros dispositivos via Fieldbus;
7. **IEC 61131-7**: Define a estrutura do uso da linguagem em programação difusa Fuzzy;
8. **IEC 61131-8**: Apresenta a implementação das linguagens definidas na norma IEC 61131-3.

3.2.1 CLP VS COMPUTADOR PESSOAL

Um CLP moderno pode ser definido como um dispositivo computacional desenvolvido para controlar um processo. Ele processa em tempo real informações vindas de sensores que

monitoram o status do processo e envia dados para os atuadores que podem alterar o processo. A maior diferença entre o CLP e um computador pessoal está em sua arquitetura: enquanto computadores pessoais possuem chips robustos capazes de realizar várias tarefas e programas ao mesmo tempo, os processadores de CLP são mais simples e realizam apenas uma tarefa de forma sequencial. Eles possuem um microprocessador ligado diretamente com a memória e chips I/O através de endereços paralelos. Outra diferença é que não possui telas/*displays* convencionais igual os computadores pessoais mas sim uma HMI (*Human Machine Interface*), que em português traduzimos para IHM (Interface Homem Máquina), um pequeno *display* que é responsável por mostrar os processos em tempo real. Também são equipados com terminais para entrada e saída de dispositivos, e algumas portas de comunicação. Por serem utilizados em fábricas e ambientes parecidos, os CLP devem ser capazes de aguentar condições extremas como altas temperaturas e altas umidades. São de instalação fácil e possuem uma manutenção simplificada pois possuem indicadores de erro que são mostrados na IHM. As entradas e saídas são acessíveis e podem ser facilmente conectadas.

3.2.2 Hardware do CLP

Todos os hardwares CLP possuem características similares definidas pelo padrão IEC 61131-1:

- **Rack de montagem:** estrutura responsável por "juntar"todos os outros blocos do CLP;
- **Fonte DC:** a fonte fornece corrente contínua paras os outros blocos no *rack*. Em sistemas maiores, corrente AC ou DC é fornecida para os dispositivos na planta. Em alguns sistemas de micro capacidade, a fonte DC alimenta toda a planta;
- **Dispositivo programador:** é usado para programar a CPU e seu tipo depende da fabricante do CLP ou da preferência do consumidor. Entretanto, a maioria das empresas utiliza os PC como dispositivos programadores, e faz a conexão com o CLP a partir de protocolos como TCP/IP ou USB;
- **Entradas e saídas:** esse bloco é responsável pela conexão de todos os dispositivos da planta e serve como a interface entre a CPU e eles. Existem CLPs com o bloco de entradas/saídas fixas e outros com blocos externos. O tipo fixo é muito utilizado em microssistemas, mas por conta disso não é expansível;
- **Unidade Central de Processamento (CPU)**: Coordena e controla toda a operação do CLP. O módulo processador contém um circuito integrado que possui um ou mais micro-controladores, chips de memória e alguns circuitos que permitem armazenamento e leitura de dados. É o módulo mais importante do CLP, pois sua função principal é a interpretação dos códigos.

Figura 10 – Diferentes módulos de um CLP

Fonte: Site Siemens (<https://www.siemens.com/>, acessado em: JULHO de 2023)

3.2.3 Programando o CLP

Como foi mostrado anteriormente, a norma IEC 61131-3 define alguns padrões para as linguagens de programação encontradas nos CLPs. Existem 5 tipos básicos de linguagens padronizadas, divididas em linguagens textuais (texto estruturado, lista de instruções) e linguagens gráficas (Diagrama Ladder, Diagrama de Blocos Funcionais e Grafcet).

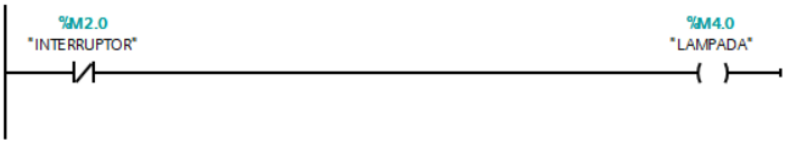
- **Texto Estruturado:** é uma linguagem textual que usa instruções em forma de blocos lógicos, como IF, THEN, ELSE, FOR, WHILE, etc. É uma linguagem mais flexível e poderosa que a Ladder, mas requer mais conhecimento de lógica de programação.
- **Lista de Instruções**: é uma linguagem textual que usa instruções em forma de códigos mnemônicos, como LD, ST, AND, OR, etc. É uma linguagem de baixo nível, que permite um controle mais direto do hardware do CLP, mas é mais difícil de ler e entender.
- **Ladder:** é uma linguagem gráfica que simula um diagrama elétrico, usando símbolos de contatos, bobinas, temporizadores, contadores, etc. É uma linguagem fácil de aprender e de visualizar o funcionamento do programa.
- **Diagrama de Blocos Funcionais:** é uma linguagem gráfica que usa blocos pré-definidos que representam funções matemáticas, lógicas ou específicas do CLP. É uma linguagem que facilita a modularização e a reutilização do código, mas pode ser complexa de depurar e testar.
- **Grafcet:** é uma linguagem gráfica que usa símbolos de etapas, transições e ações para representar o fluxo de um processo. É uma linguagem que permite modelar o comportamento sequencial e paralelo de um sistema, mas pode ser difícil de integrar com outras linguagens.

Neste trabalho iremos utilizar o diagrama Ladder para realizar a programação do CLP, sendo detalhado na próxima seção. A programação dos CLPs da Siemens é realizada através do software TIA Portal.

3.2.3.1 Diagrama Ladder

A linguagem Ladder é uma ferramenta gráfica usada para desenvolver softwares para CLPs e é baseada na representação gráfica da lógica de relés representada por contatos (chaves) e bobinas. É responsável pela lógica de controle, indicando para o controlador qual ação deve ser realizada a partir dos valores de entrada e atualizando as saídas ou atuadores que interagem com os processos industriais. Como foi visto anteriormente, ela tenta representar de forma idêntica o circuito necessário no controle. Também é conhecida como lógica de diagrama de contatos, pois se assemelha à tradicional notação de diagramas elétricos e dos painéis de controle de relés. Foi a primeira linguagem de programação desenvolvida para os CLPs e é amplamente utilizada para programar circuitos que compõem processos industriais, como o acionamento de motores elétricos e cilindros hidráulicos. É uma das cinco linguagens propostas pela norma IEC 61131-3 e é de fácil compreensão por aqueles que têm algum conhecimento sobre circuitos elétricos.

Figura 11 – Exemplo de código escrito em Ladder com um interruptor controlando uma lâmpada

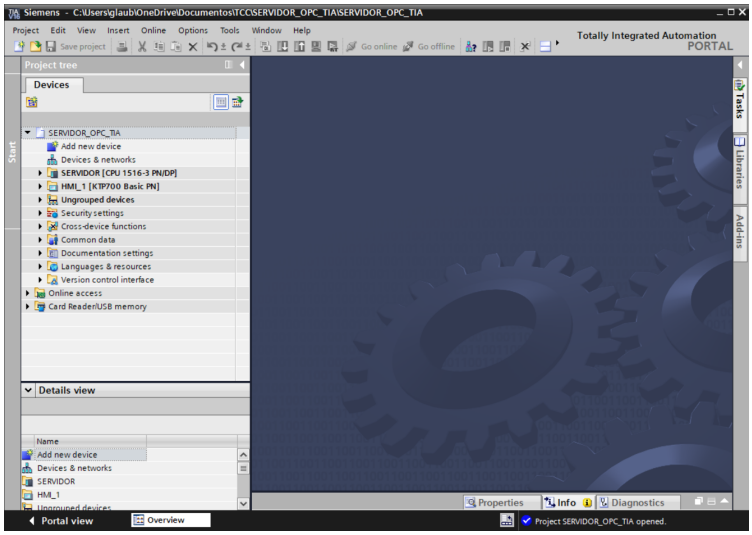


Fonte: Autoria própria utilizando o software TIA Portal.

3.3 TIA Portal

O TIA Portal (*Totallly Integraded Automation Portal*) é um software de automação da Siemens criado para ser utilizado de forma fácil e intuitiva. Lançado em 2010, ele oferece um conceito operacional mais padronizado para o uso de controladores e IHM (interface homem-máquina) da empresa, garantindo estabilidade e consistência de dados durante seu uso, além de fornecer acesso a uma robusta biblioteca para todos os casos de uso. A versão V16 do software, utilizada neste trabalho, engloba uma série de produtos como o SIMATIC STEP 7, SIMATIC WinCC, entre outros. Aqui aprofundaremos apenas no primeiro citado.

Figura 12 – Captura de tela da pagina inicial do TIA Portal

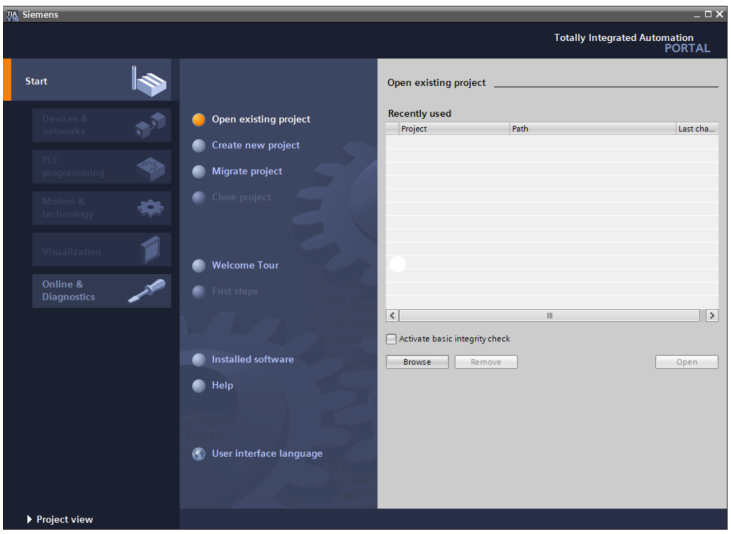


Fonte: Captura de tela de autoria própria.

3.3.1 SIMATIC STEP 7

O SIMATIC STEP 7 é uma ferramenta de configuração de controladores industriais da linha Siemens SIMATIC. A variante que foi utilizada neste trabalho é a Profissional V16, que é destinada para a configuração de controladores da família SIMATIC S7-1500, S7-1200, S7-300 e S7-400. O software possui diversas funções chave para a programação do CLP, como o controle e gerenciamento de projetos, configuração e definição de parâmetros do hardware e comunicação, criação de projetos, carregamento de programas no controlador e teste de sistemas de automação. Sua interface é simples e permite o fácil entendimento do usuário.

Figura 13 – Captura de tela da pagina inicial do SIMATIC STEP 7



Fonte: Captura de tela de autoria própria.

Quando uma solução de automação é criada no SIMATIC STEP 7, uma série de passos

básicos devem ser seguidos. É possível seguir dois processos alternativos: primeiro configurar o hardware e depois construir o diagrama de blocos, ou, construir o diagrama de blocos sem configurar o hardware (essa abordagem é muito utilizada quando é necessário fazer uma manutenção externa, ou quando se quer programar os blocos separadamente e depois integrar com um projeto existente. Podemos analisar os passos no diagrama abaixo, logo após uma descrição detalhada de cada passo.

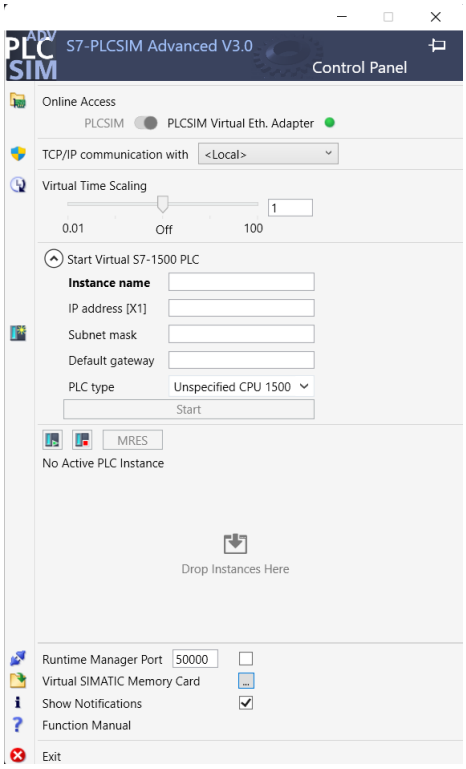
1. Instalar o SIMATIC STEP 7 e instalar as licenças necessárias;
2. Planejar o controlador: antes de acessar o STEP 7, é necessária uma etapa de planejamento, analisando a viabilidade e verificando as necessidades e as tarefas necessárias do controlador em si;
3. Desenhar a estrutura do programa: transformar o planejamento do passo anterior no programa estruturado utilizando os blocos disponíveis no STEP 7;
4. Iniciar o STEP 7: Abrir a interface de usuário do software;
5. Configurar a estrutura do projeto: criar o projeto definindo a sua estrutura hierárquica;
6. Configurar a estação de trabalho: especificamos qual o controlador programável utilizaremos;
7. Configurar o Hardware: Quando configuramos o hardware, estamos especificando numa tabela de configuração quais são os módulos que usaremos para a automação e quais os endereços serão utilizados para fazer a comunicação com eles;
8. Configurar a rede e conexões de comunicação: é necessário criar as sub-redes necessárias para a rede de automação e definir as propriedades da conexão;
9. Definir os símbolos: Pode se definir símbolos locais ou compartilhados, que podem ter nomes mais descritivos ou usar uma tabela pré-definida;
10. Criar o programa: nesse passo que vamos criar o código que rodará no CPL utilizando uma das linguagens disponíveis;
11. Carregar o programa: Depois de todas as configurações realizadas, é necessário carregar o programa criado no CLP. Pode-se carregar o programa por inteiro de uma vez ou carregar individualmente por blocos;
12. Teste do programa: Após concluir o carregamento você pode exibir os valores das variáveis na CPU, analisando o resultado encontrado.
13. Monitoramento da operação: caso ocorra algum tipo de falha no hardware, é possível verificar os logs do buffer de erro;
14. Documentar a planta: Logo após o término de todo esse processo, o próximo passo é realizar a documentação da planta, levando em consideração o programa feito.

3.4 PLCSIM Advanced

O PLCSIM Advanced (V3.0) é um software de simulação de CLP também desenvolvido pela Siemens, onde sua principal função é simular um controlador de forma digital e o seu uso não necessita de um controlador real. A partir dessa simulação é possível observar a programação lógica e analisar os efeitos do sistema, verificando as entradas e saídas para adaptar o programa.

A versão 3.0 do software é capaz de simular as CPUs da família S7-1500, como por exemplo a 1511-1 PN e a 1515-2 PN. Além da comunicação via Softbus, o PLCSIM Advanced também permite a comunicação via Ethernet utilizando o protocolo TCP/IP. Na simulação temos alguns diferenciais que não são possíveis quando usamos um CLP real, como detecção mais rápida de erros, diminuição no tempo de resposta em testes e facilidade no treinamento de funcionários.

Figura 14 – Captura de tela da pagina inicial do PLCSIM Advanced V3.0



Fonte: Captura de tela de autoria própria.

Apesar dos benefícios do uso do PLCSim Advanced temos que levar em conta alguns pontos quando simulando um programa, pois existem diferenças entre a simulação e uma CPU real. O simulador não consegue simular uma CPU nos mínimos detalhes, mesmo se o programa funcionar sem erros no CLP não é garantido que ele vá funcionar sem problemas no simulador. Como o software roda em cima do sistema operacional Windows o tempo de ciclo do simulador é diferente da CPU do CLP, pois além do PLCSim Advanced dividir processamento com outros programas, também divide com próprio Windows. Para garantir o melhor resultado possível é necessário pelo menos um núcleo físico do processador do PC disponível por instância simulada.

Também temos alguns problemas de segurança: por se tratar de um software que permite a simulação em mais de um computador simultaneamente, a comunicação entre eles não é criptografada. Na interface do usuário não existe opções para autenticação ou autorização de usuários, deixando o software "aberto" a qualquer pessoa que tiver acesso ao computador. Se não bastasse os problemas citados, encontramos um ainda maior: como a comunicação com o TIA Portal pode ser feita através do protocolo TCP/IP, qualquer pessoa com acesso a rede local tem acesso a simulação. A Siemens recomenda o uso de uma rede local privada para realizar simulações.

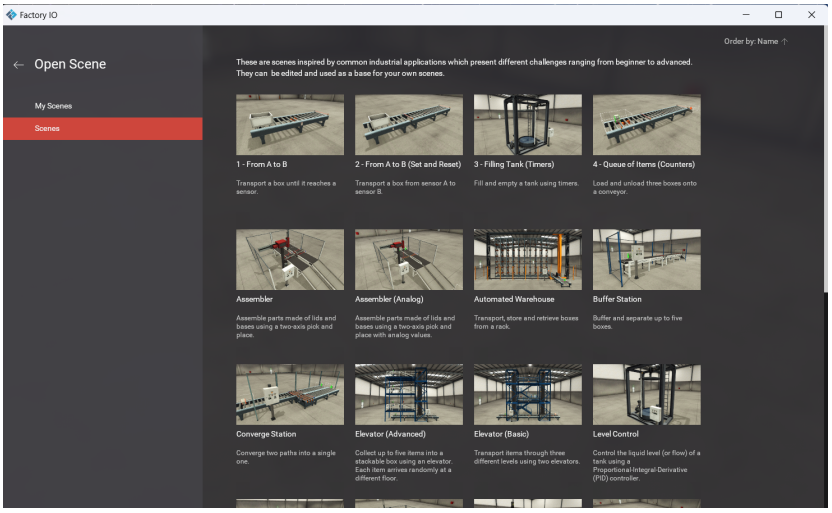
A comunicação é feita através do PLCSIM *Virtual Ethernet Adapter*, uma interface virtual que simula uma interface de rede real. O adaptador virtual funciona como uma ponte entre as instâncias do simulador e o TIA Portal.

Também existe a possibilidade de se realizar uma comunicação distribuída através do protocolo TCP/IP. Isso significa que o PLCSIM Advanced se comunica com outros dispositivos através da rede local através do adaptador de Ethernet existente no PC. Pode ser utilizado um ou mais acessos. Esse método é o menos recomendado, já que expõe a simulação para qualquer pessoa que tiver acesso a rede local (como foi dito anteriormente).

3.5 Factory IO

Podemos definir o Factory I/O como um ambiente de simulação 3D de linhas de produção, muito utilizado para testes no design de processos de automação industrial. O software foi desenvolvido para ser fácil de usar e permite a rápida criação de "fábricas virtuais" utilizando partes simuladas de grandes fabricantes da indústria de automação, como sensores e atuadores. O software já possui diversas cenas pré-montadas dos cenários mais comuns nas indústrias, como esteiras separadoras. O ambiente de trabalho também oferece uma ampla variedade de componentes que podem ser utilizados para montar a sua própria cena da fábrica virtual, entre eles podemos citar: esteiras, motores, sensores de proximidade, entre vários outros.

Figura 15 – Captura de tela da pagina inicial do Factory IO



Fonte: Captura de tela de autoria própria.

Por ser super flexível, o Factory IO funciona muito bem como uma plataforma de testes de sistemas de automação antes de se finalizar o design das fábricas reais. Ele pode funcionar em conjunto com os controladores programáveis dos fabricantes mais utilizados da indústria como os da linha SIMATIC da Siemens e os da Allen-Bradley. A conexão com o CLP pode ser direta utilizando do protocolo TCP/IP ou através de um servidor OPC, pois o software já possui suporte integrado.

4 METODOLOGIA

A metodologia da pesquisa desenvolvida foi dividida em dois momentos principais: a etapa teórica, na qual se realizou uma revisão bibliográfica sobre os conceitos e ferramentas utilizados no projeto, e a etapa prática, na qual se implementou as funcionalidades propostas para o sistema.

Para o desenvolvimento do projeto descrito neste trabalho, seguiram-se as seguintes etapas de execução:

4.1 Estudo das Ferramentas

O estudo das ferramentas a serem utilizadas foi realizado a partir da análise de um conjunto de referências bibliográficas de qualidade, visando ampliar os conhecimentos na área do tema proposto.

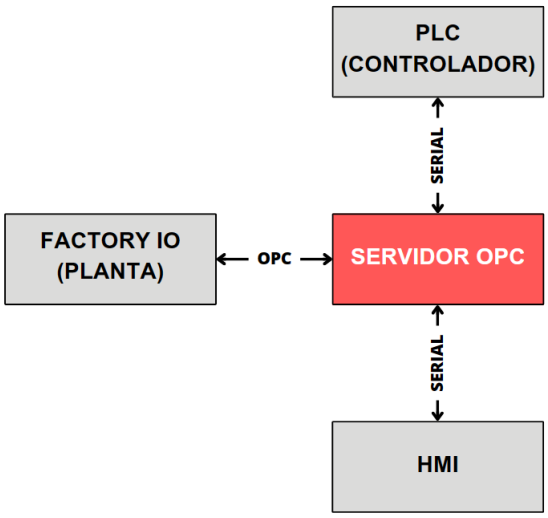
Nesta fase, foram estudados temas como servidor OPC UA e sua implementação, simulação de um CLP através do PLCSim Advanced, programação do CLP com o software TIA Portal e também como realizar a simulação de uma planta através do Factory IO.

4.2 Implementação

Após o fim da etapa de revisão bibliográfica da literatura existente foi dado início a implementação do trabalho, realizado em etapas. Como foi definido no Capítulo 2, nosso intuito é realizar o controle PID de uma planta contendo um sistema de tanque através do Simulink, ferramenta presente no MATLAB, com a comunicação feita através do protocolo OPC UA e comparar seu funcionamento com o controle sendo realizado diretamente pelo CLP.

Primeiramente controlaremos a planta pelo controlador do próprio CLP (simulado através do PLCSim Advanced), como mostrado na Figura 16, e analisaremos o comportamento deste sistema. O chamaremos de sistema 1.

Figura 16 – Diagrama de funcionamento do sistema 1

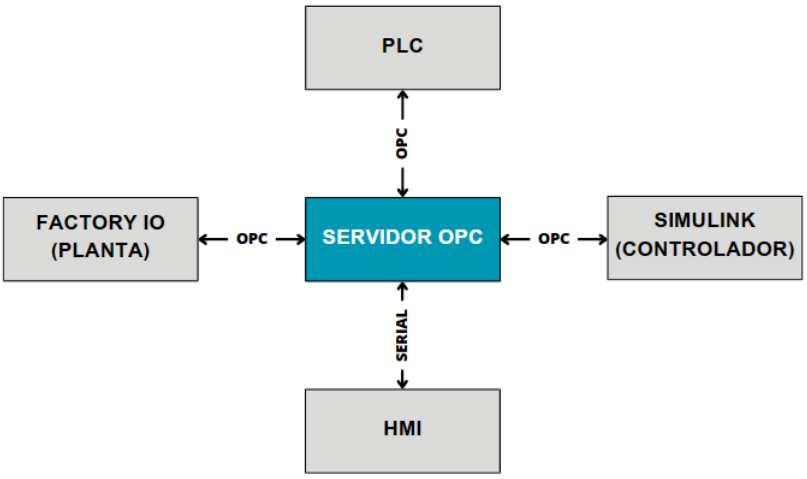


Fonte: Autoria própria.

Em seguida o sistema será modificado para que o controle PID seja feito pelo Simulink com a comunicação do controlador sendo feita através do protocolo OPC UA, como demonstrado no diagrama (17). No fim analisaremos o comportamento do sistema da mesma forma feita anteriormente, para então compararmos os dois resultados.

Ambos os sistemas utilizam uma IHM para analisar a resposta do sistema, e a comunicação é feita por comunicação serial diretamente com o CLP.

Figura 17 – Diagrama de funcionamento do sistema 2



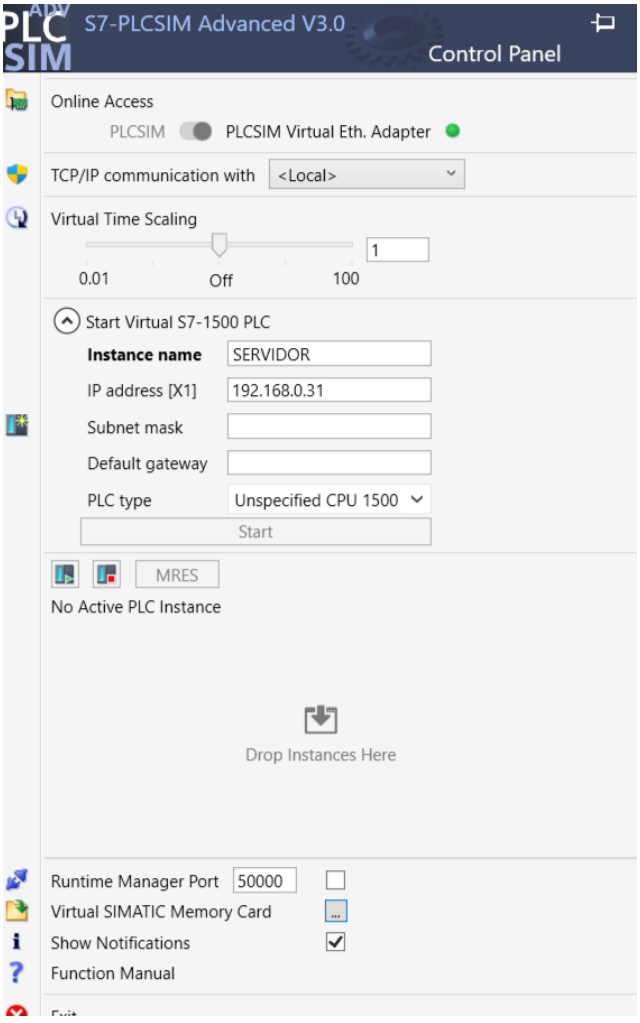
Fonte: Autoria própria.

Partindo do que foi definido, o primeiro passo é a instalação dos softwares, demonstrados a seguir:

4.2.1 Configuração PLCSIM Advanced

Após realizar a instalação do software seguindo as instruções fornecidas pela fabricante, iniciamos a configuração da simulação do CLP. Nesse passo é necessária apenas a definição de três itens: nome da instância, endereço IP e tipo da CPU. Aqui foi escolhido um endereço arbitrário (192.168.0.31) e uma CPU da família 1500. Ao iniciar a simulação, a instância do CLP já fica disponível de forma imediata na rede local.

Figura 18 – Captura de tela da página inicial do PLCSim Advanced V3.0

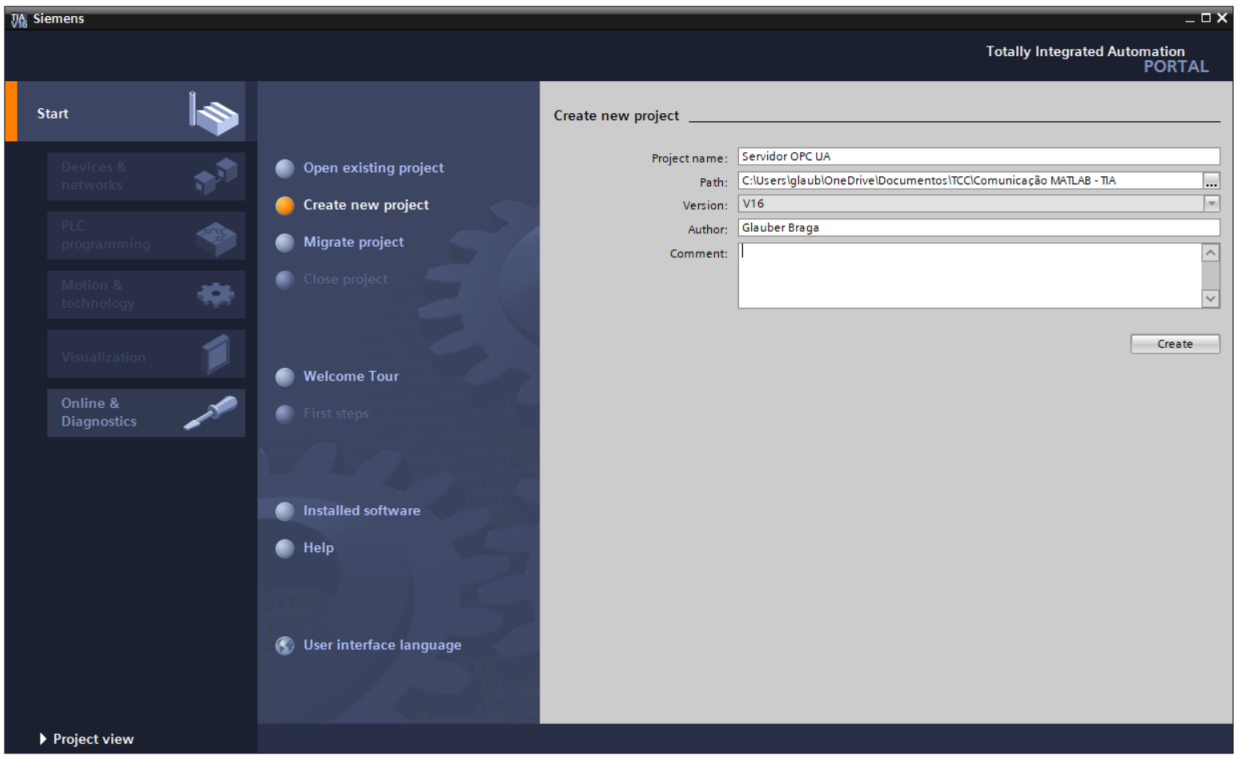


Fonte: Captura de tela de autoria própria.

4.2.2 Configuração do servidor OPC através do TIA Portal

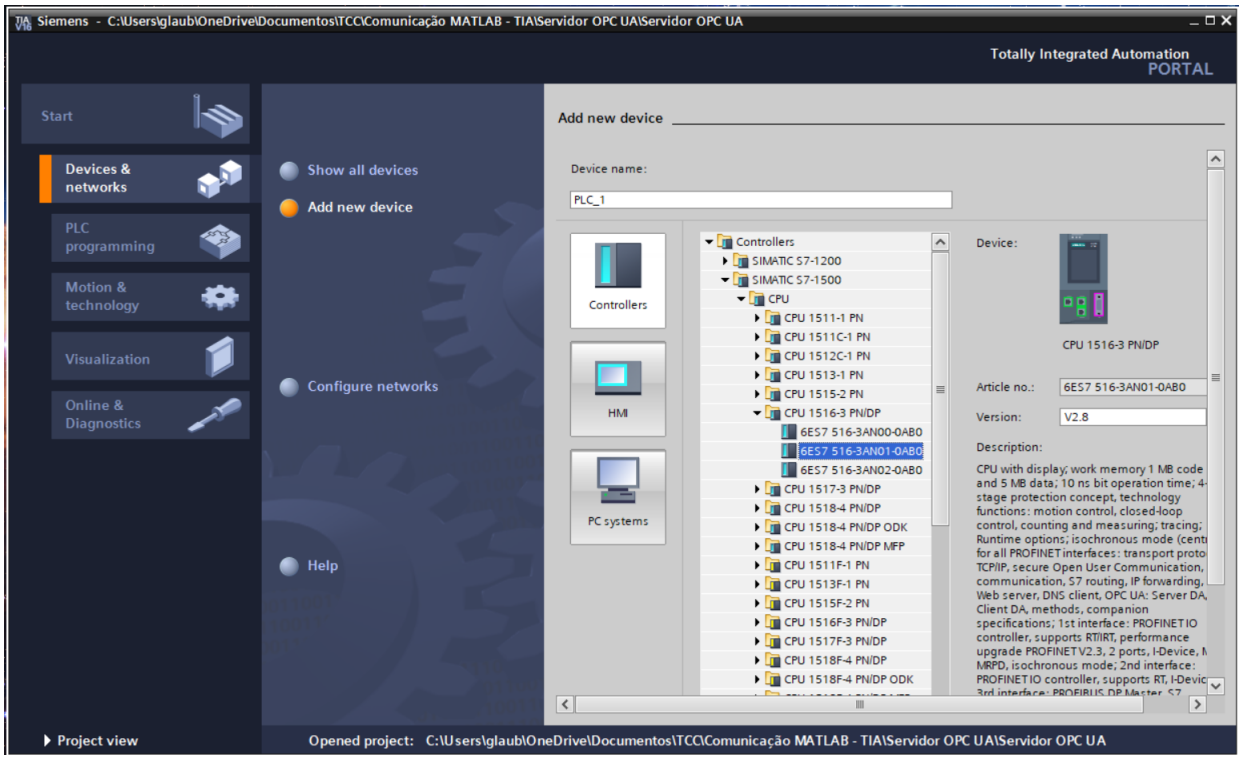
Em seguida é necessário realizar a instalação do software de programação do controlador (TIA Portal), que pode ser encontrado de forma gratuita, em versão de teste, no site oficial da fabricante (<https://support.industry.siemens.com/>). Depois de instalado, abrimos um novo projeto e realizamos algumas configurações básicas, como nome do projeto e dispositivo que iremos utilizar, que neste caso é uma simulação de um CLP com a CPU S7-1516-3 PN (19, 20).

Figura 19 – Captura de tela da página inicial do TIA Portal V16



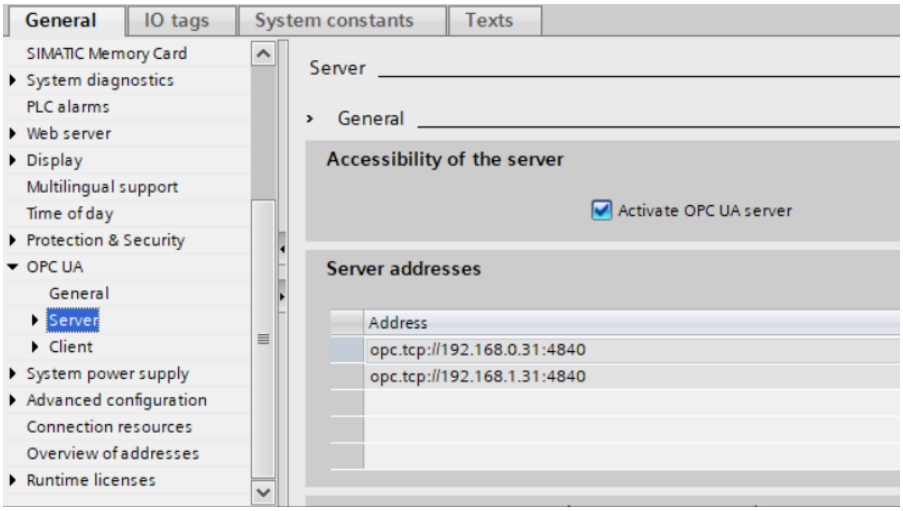
Fonte: Captura de tela de autoria própria.

Figura 20 – Captura de tela da página de definição de dispositivo do TIA Portal V16



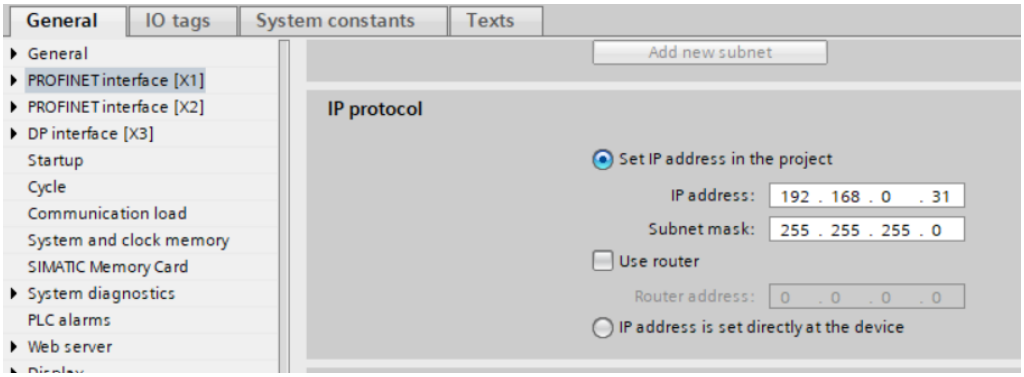
Para ativarmos o servidor OPC UA não precisamos realizar muitas alterações no projeto pois o próprio TIA Portal possui suporte integrado. Com o projeto aberto acessamos a página de configurações do dispositivo e alteramos duas configurações: ativamos o servidor OPC e alteramos o endereço IP da conexão para o mesmo que definimos na etapa 4.2.1.

Figura 21 – Captura de tela da página de ativação do servidor OPC UA



Fonte: Captura de tela de autoria própria.

Figura 22 – Captura de tela da página de configuração IP do dispositivo

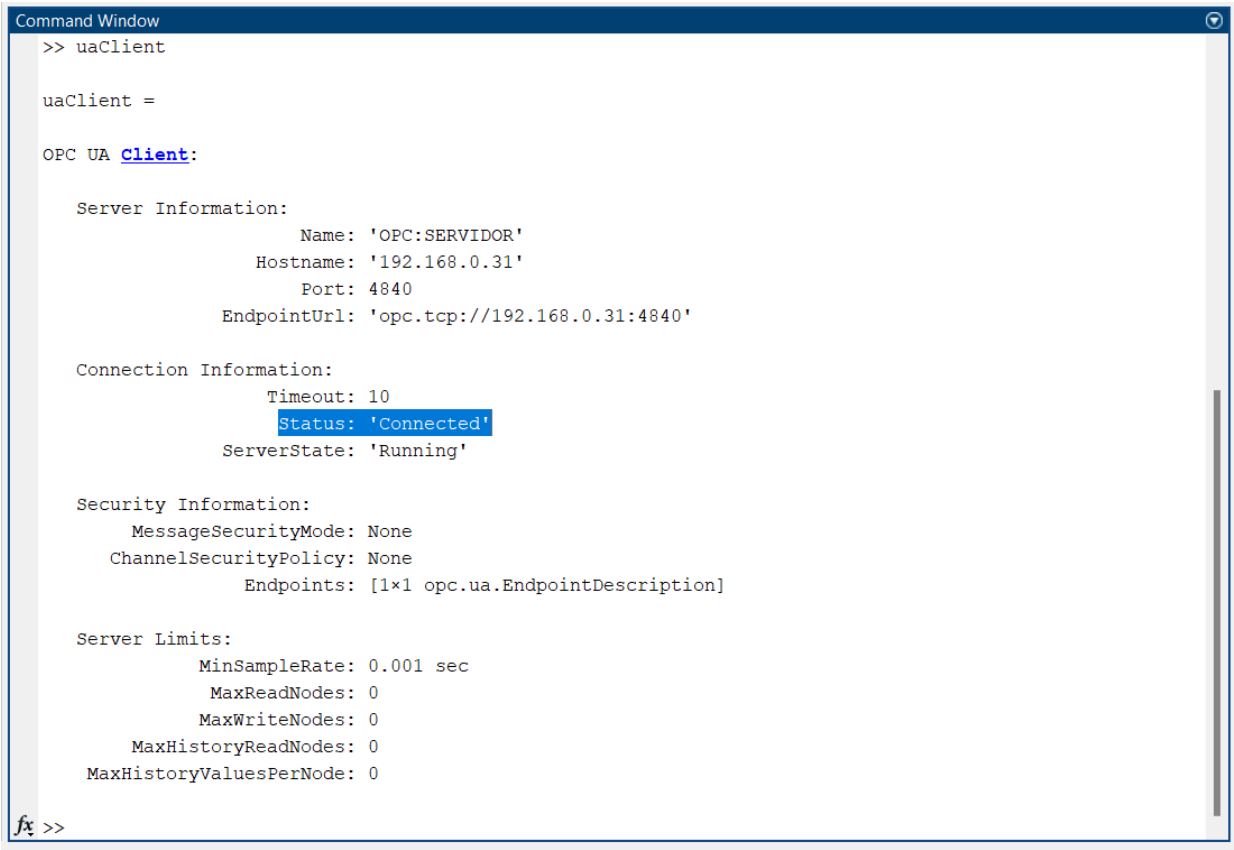


Fonte: Captura de tela de autoria própria.

Ao fim da configuração inicial, podemos compilar o projeto e carrega-lo na instância do PLCSim Advanced para realizar alguns testes utilizando o terminal de comandos do MATLAB. Utilizando os comandos descritos abaixo, verificamos que o servidor funciona e aceita conexão de um cliente OPC UA.

```
1    uaClient = opcua('192.168.0.31',4840)
2    connect(uaClient)
3    uaClient
```

Figura 23 – Captura de tela do terminal de comandos do MatLab

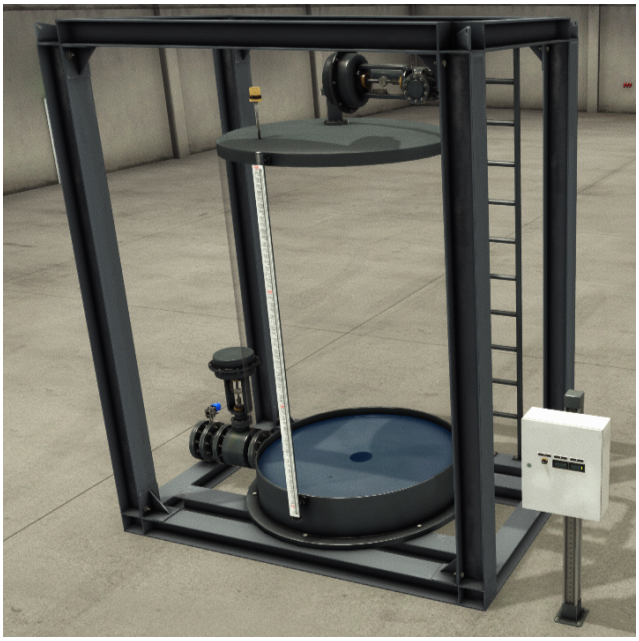


Fonte: Captura de tela de autoria própria.

4.2.3 Configurar a interface de comunicação entre Factory IO e servidor OPC UA

Para iniciarmos a conexão, devemos encontrar as variáveis que usaremos no processo e podemos fazer isso analisando a cena do Factory IO que realizaremos o controle. Nas imagens (24, 25, 26, 27) podemos encontrar estas variáveis, e por motivos de simplificação utilizaremos apenas: botoeira seletora de *setpoint*, sensor de nível d'água, display de *setpoint*, display de nível e bomba d'água.

Figura 24 – Captura de tela da cena utilizada no Factory IO

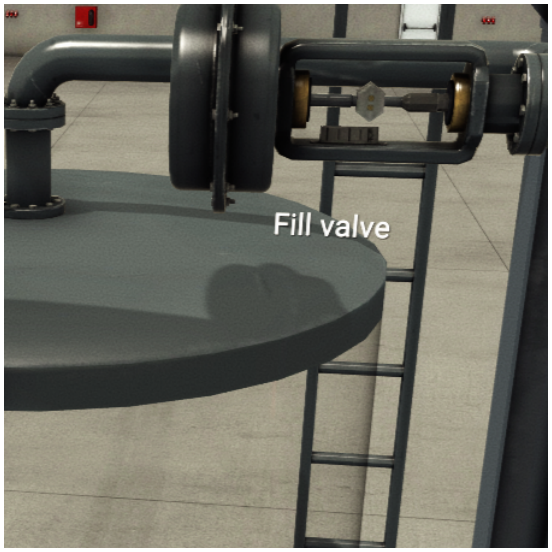


Fonte: Captura de tela de autoria própria.

Figura 25 – Captura de tela do painel de botoeiras e displays na cena do Factory IO

Fonte: Captura de tela de autoria própria.

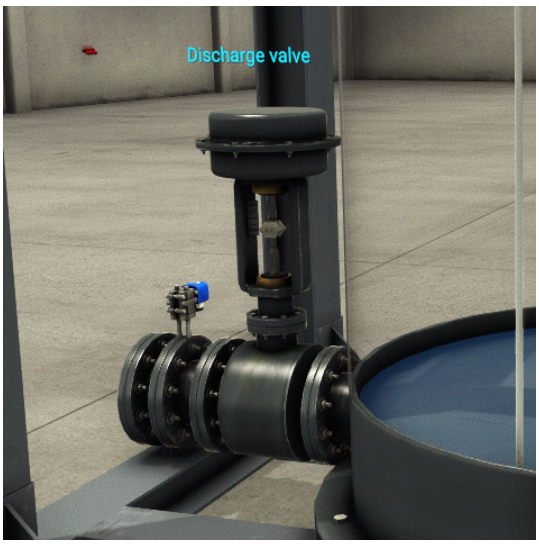
Figura 26 – Captura de tela da bomba d'água na cena do Factory IO



Fonte: Captura de tela de autoria própria.

Durante toda realização do trabalho foi utilizado um valor constante na válvula de descarga de água. A válvula é controlada na escala de 0 (totalmente fechada) a 10 (totalmente aberta), a partir disso definimos o valor constante 5.

Figura 27 – Captura de tela da válvula de descarte do tanque.



Fonte: Captura de tela de autoria própria.

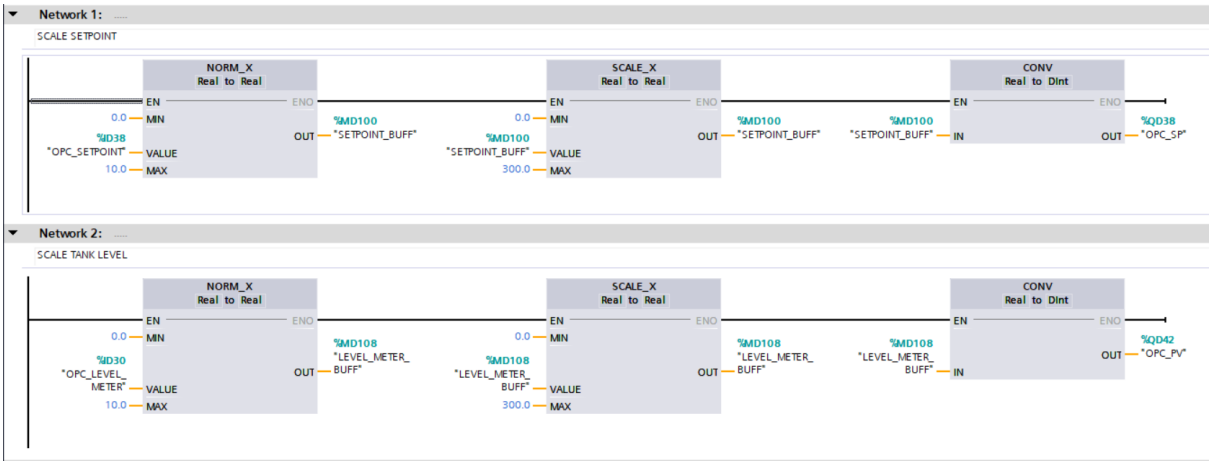
No próximo passo precisamos definir como o controlador PID irá interagir com a planta, e para isso faremos um pequeno código em ladder no TIA Portal V16 que servirá apenas como intermediador e facilitador da comunicação da planta com o servidor. Por questão de simplificação, utilizaremos o mesmo projeto feito anteriormente para a definição do servidor. O primeiro passo é a definição das variáveis que encontramos anteriormente (Figura 25) e, em seguida, definimos um conversor que recebe as informações da botoeira e do nível de água e os colocam em uma escala que permite o seu controle, conforme a Figura 29.

Figura 28 – Captura de tela das variáveis definidas no projeto do TIA Portal.

		Name	Data type	Address	Retain	Acces...	Writa...	Visibl...
1		SETPOINT_BUFF	Real	%MD100	☐	☑	☑	☑
2		PID_OUTPUT_REAL	Real	%MD124	☐	☑	☑	☑
3		PID_OUTPUT	Int	%MW116	☐	☑	☑	☑
4		OUT_PID_TIA	DWord	%MD130	☐	☑	☑	☑
5		OPC_SP	DInt	%QD38	☐	☑	☑	☑
6		OPC_SETPOINT	Real	%ID38	☐	☑	☑	☑
7		OPC_PV	DInt	%QD42	☐	☑	☑	☑
8		OPC_LEVEL_METER	Real	%ID30	☐	☑	☑	☑
9		OPC_FILL_VALVE	Real	%QD30	☐	☑	☑	☑
10		LEVEL_METER_BUFF	Real	%MD108	☐	☑	☑	☑

Fonte: Captura de tela de autoria própria.

Figura 29 – Captura de tela do conversor de escala no projeto do TIA Portal.

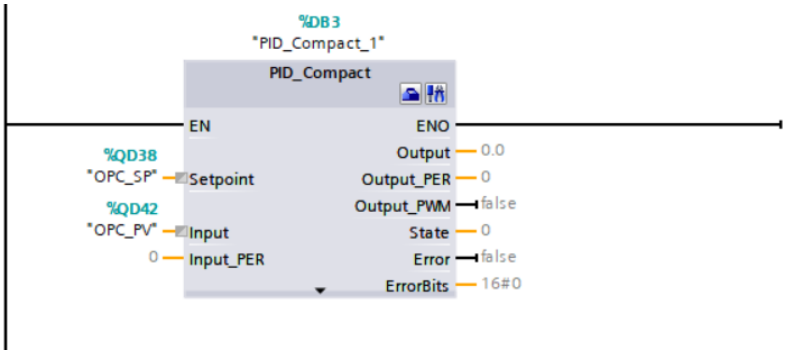


Fonte: Captura de tela de autoria própria.

Como definido anteriormente, um dos objetivos deste trabalho é realizar o controle da planta com um PID através do TIA Portal e comparar o seu desempenho com o controle através do Simulink com uso do servidor OPC UA. Dito isso, primeiramente iremos realizar o controle a partir do TIA Portal, e para isso se faz necessário configurar o controlador disponibilizado pelo pelo software que pode ser visto na Figura 30.

Na entrada *setpoint* do controlador associamos a variável *setpoint* e na entrada *input* utilizamos a o nível de água no tanque, ambas já na escala desejada, feito no passo anterior. A saída do controlador é enviada para um conversor de escalas.

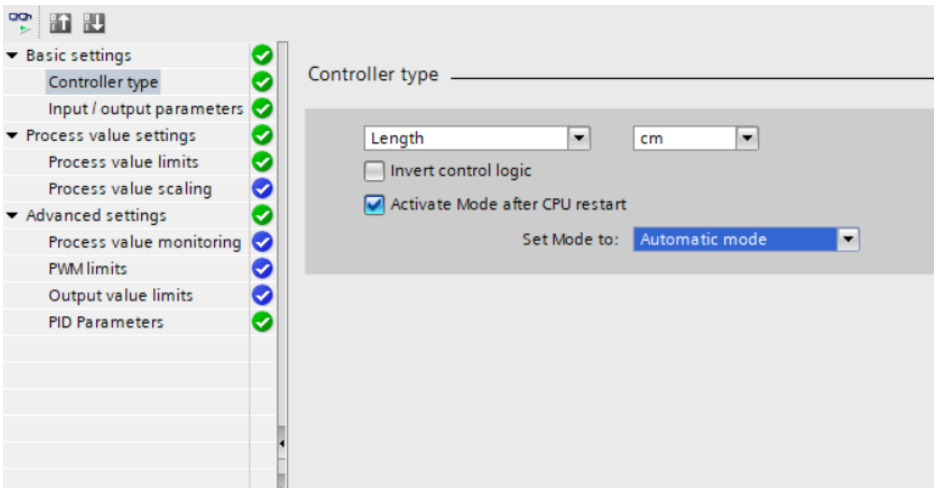
Figura 30 – Captura de tela do controlador PID disponibilizado pelo TIA Portal.



Fonte: Captura de tela de autoria própria.

O PID disponibilizado pelo TIA Portal precisa de algumas configurações antes de ser utilizado. Primeiro foi definido o tipo de controle que será realizado, neste caso é de medição de comprimento em centímetros pois a leitura que recebemos da planta se trata da altura da água no tanque. Logo após, definimos o intervalo de variação, que é de 0 cm a 300 cm.

Figura 31 – Captura de tela das configurações do controlador PID.



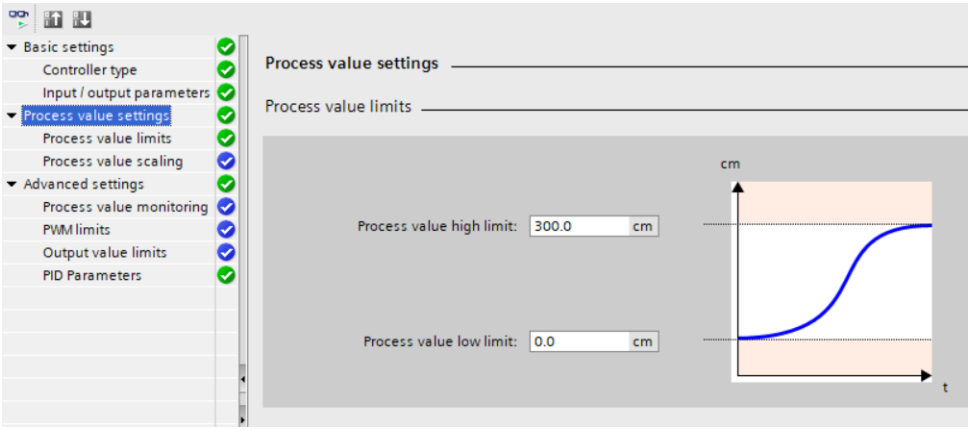
Fonte: Captura de tela de autoria própria.

Tabela 1 – Tabela contendo parâmetros do controlador PID encontrados pelo *auto-tunning*.

Kp	Ki	Kd
0.3	19.9	2.12

Fonte: Tabela de autoria própria.

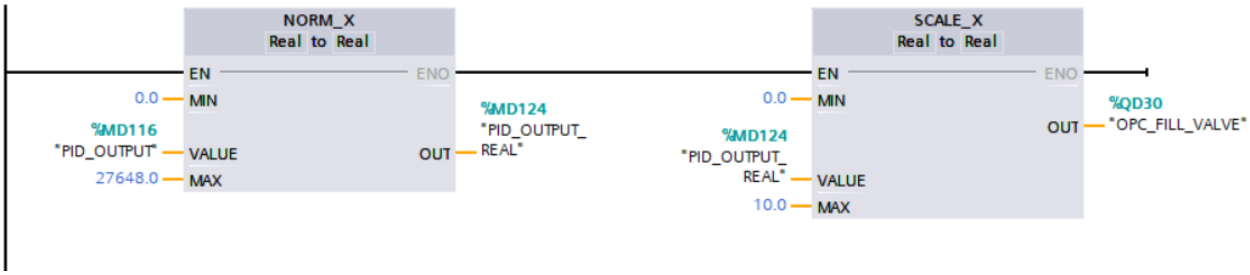
Figura 32 – Captura de tela das configurações do controlador PID.



Fonte: Captura de tela de autoria própria.

Após a configuração do controlador é necessário fazer a sua ligação com o restante do servidor e, para isso, precisamos realizar a conversão da saída do PID da mesma forma que foi feito com as variáveis lidas. A saída do PID varia entre 0 e 27648 (valores padrões definidos pelo TIA Portal), então a convertemos para os valores que o a válvula de enchimento do tanque opera, neste caso fica entre 0 e 10. Um pequeno código em ladder foi feito e pode ser visto na Figura 33.

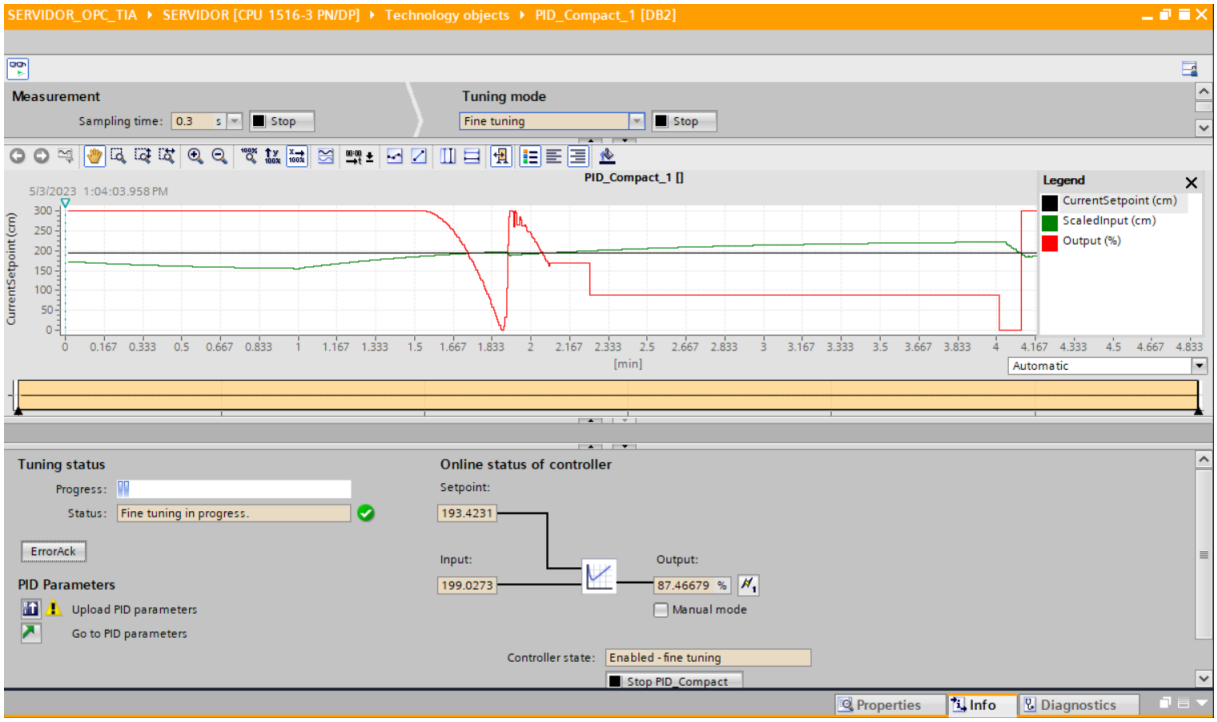
Figura 33 – Captura de tela do conversor de escala da saída do controlador PID.



Fonte: Captura de tela de autoria própria.

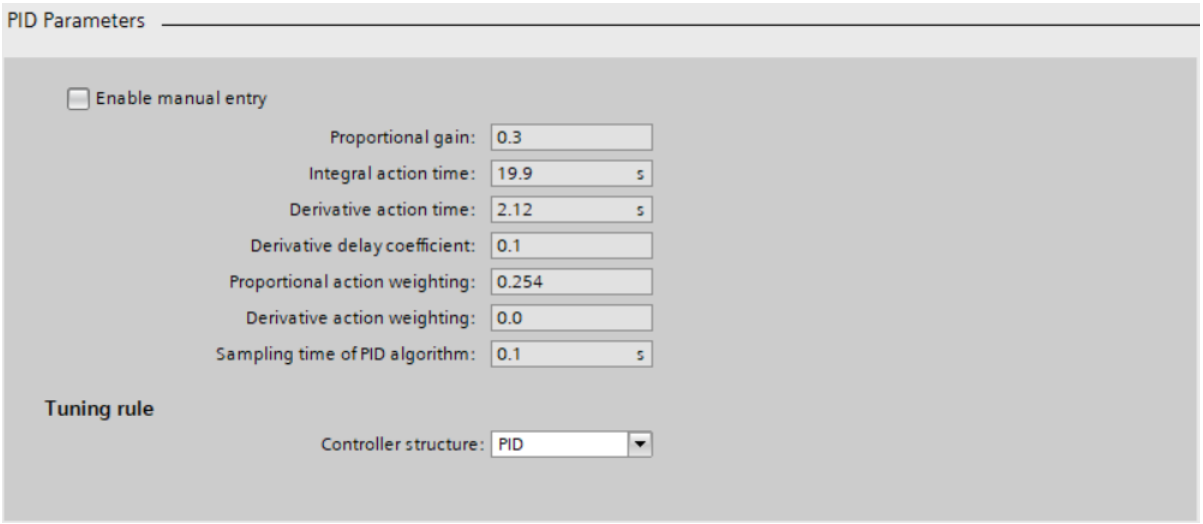
Também se faz necessário a configuração dos parâmetros do controlador, e para isso foi utilizada a ferramenta de *auto-tunning* disponibilizada pelo TIA Portal que os encontra automaticamente: foi definido um valor arbitrário de *setpoint* (neste caso, 193cm) e a válvula de descarga fixa no valor definido anteriormente (5). Iniciamos o processo e após a finalização recebemos os parâmetros do PID automaticamente. Os valores encontrados podem ser vistos na Figura 35 e na Tabela 1.

Figura 34 – Captura de tela da ferramenta *auto-tunning* do controlador PID.



Fonte: Captura de tela de autoria própria.

Figura 35 – Captura de tela dos parâmetros PID encontrados pela ferramenta *auto-tunning*.



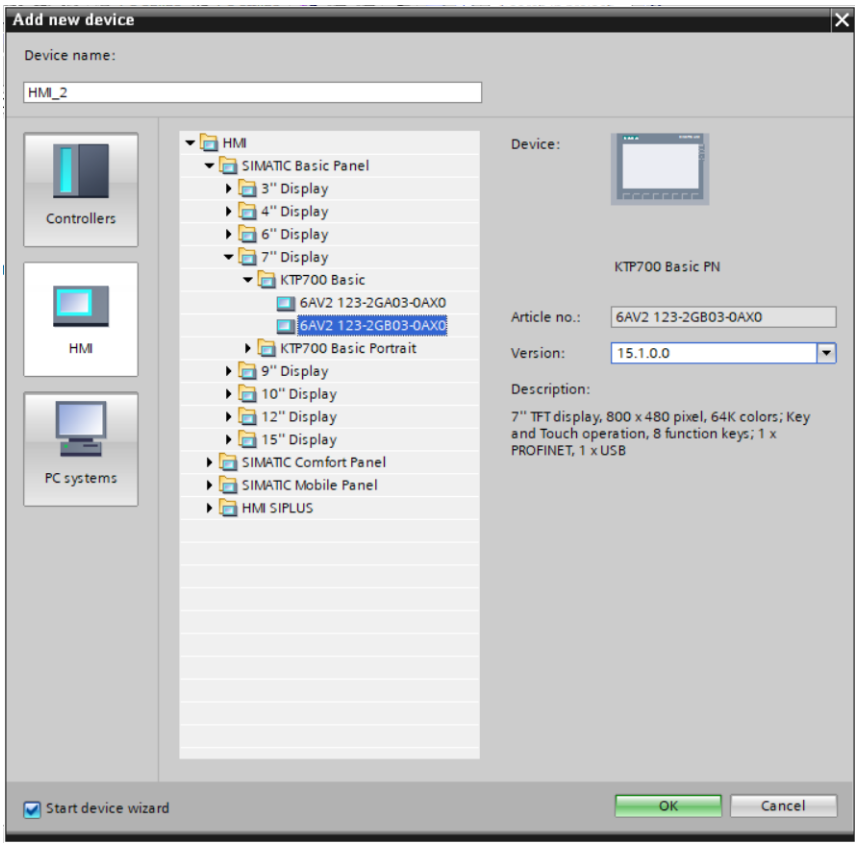
Fonte: Captura de tela de autoria própria.

A partir desse ponto nosso software já é capaz de realizar o controle da planta e podemos fazer os testes necessários, e o primeiro deles é verificar o se controlador funciona corretamente. Para isso, utilizaremos a ferramenta de simulação de uma IHM (Interface Homem Máquina) do próprio TIA Portal para analisar a resposta do controlador.

A configuração da IHM seguiu apenas alguns passos, sendo eles:

- Adicionar a IHM ao projeto (Figura 36): foi utilizado o modelo KTP700 Basic PN, de 7 polegadas, de versão 15.1.0.0;

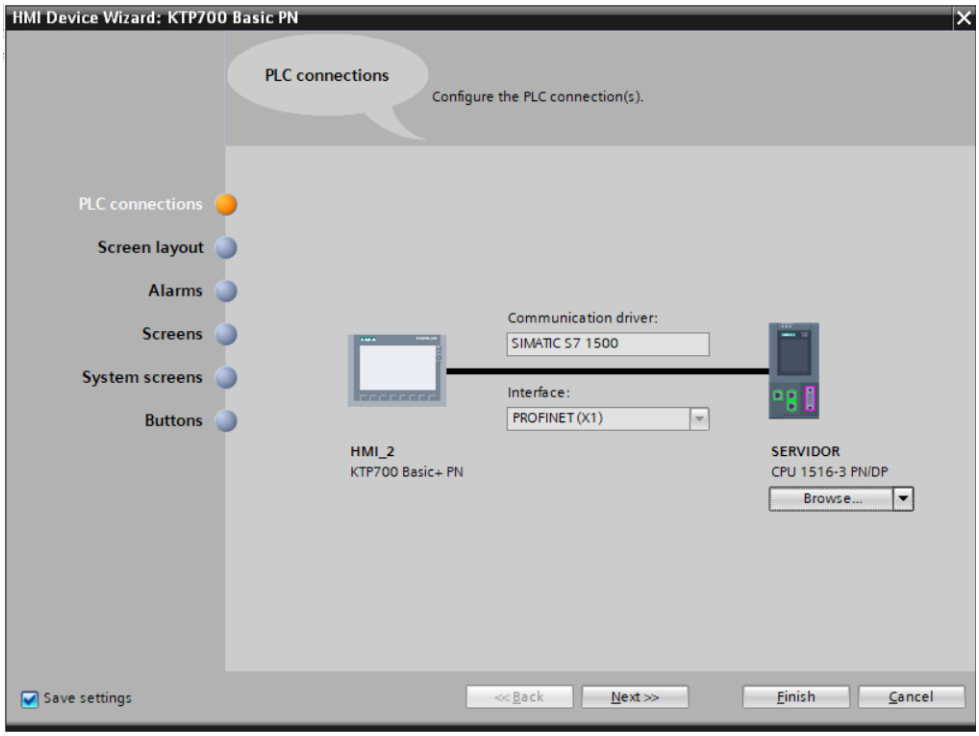
Figura 36 – Captura de tela adicionando a IHM ao projeto



Fonte: Captura de tela de autoria própria.

- Configurar o novo dispositivo virtual, conectando ao CLP configurado anteriormente (Figura 37);

Figura 37 – Captura de tela da página de conexão da IHM com o CLP



Fonte: Captura de tela de autoria própria.

- Associar à IHM as tags que serão utilizadas para visualização que são: nível do tanque e *setpoint* (Figura 38);

Figura 38 – Captura de tela da página de tags utilizadas pela IHM

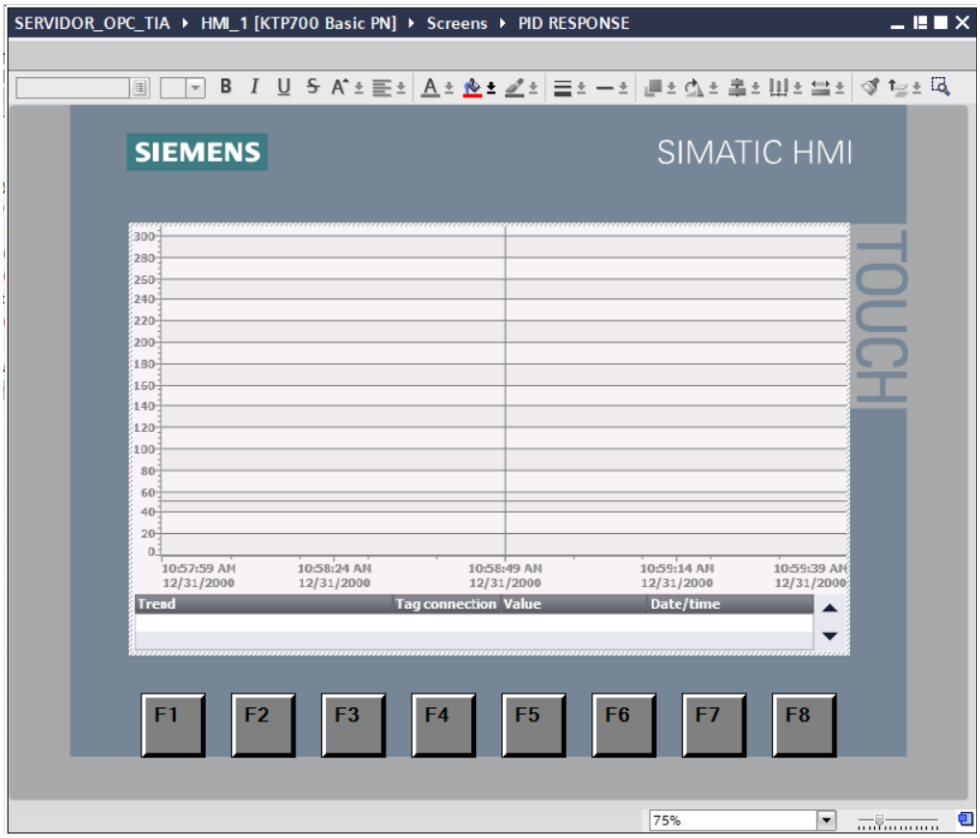
HMI tags

Name	Tag table	Data type	Connection	PLC name	PLC tag
kd	Default tag table	Real	HMI_Conne...	SERVIDOR	PID_Compact_1.CtrlP...
ki	Default tag table	Real	HMI_Connectio...	SERVIDOR	PID_Compact_1.CtrlPara...
kp	Default tag table	Real	HMI_Connectio...	SERVIDOR	PID_Compact_1.CtrlPara...
LEVEL METER	Default tag table	DInt	HMI_Connectio...	SERVIDOR	OPC_PV
SET POINT	Default tag table	DInt	HMI_Connectio...	SERVIDOR	DB_DADOS_SERVIDOR.TE...

Fonte: Captura de tela de autoria própria.

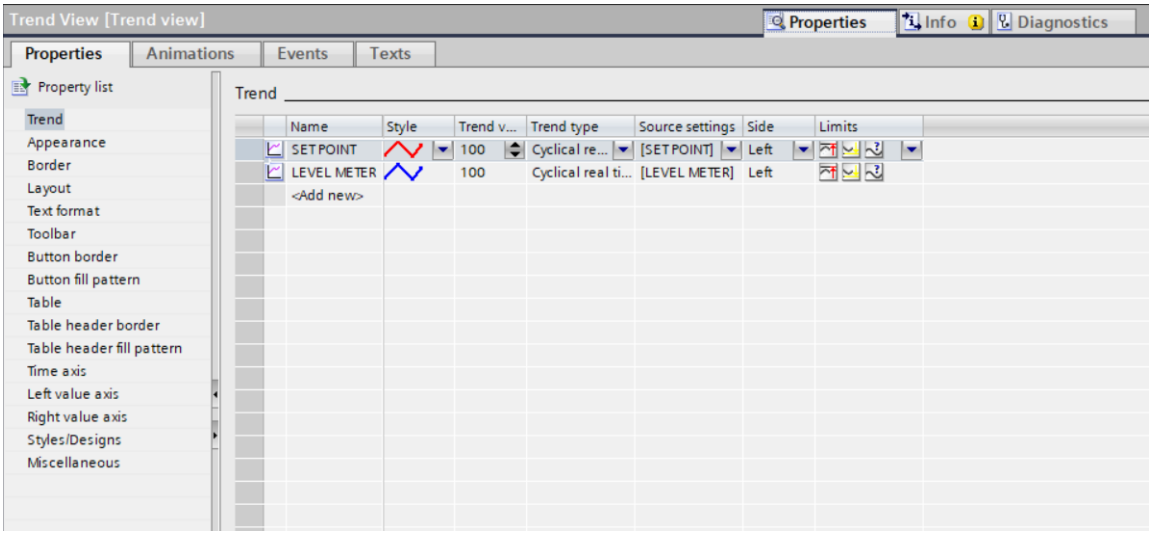
- Adicionar o bloco *trend view* (permite visualizar e analisar dados históricos e em tempo real de variáveis do controlador através de gráficos personalizados) à tela inicial (figura 39) e realizar as configurações necessárias (Figura 40);

Figura 39 – Captura de tela da ferramenta *trend view*



Fonte: Captura de tela de autoria própria.

Figura 40 – Captura de tela das configurações da ferramenta *trend view*



Fonte: Captura de tela de autoria própria.

Com a IHM devidamente configurada, nosso sistema já é capaz de realizar o controle da planta a partir do controlador do TIA Portal. Mostraremos o funcionamento do sistema no Capítulo 5.

Como foi definido no Capítulo 2, iremos comparar o seu funcionamento com a mesma planta desta vez controlada com um PID no Simulink conectado através de um servidor OPC UA. Para realizarmos o próximo passo, precisamos configurar as outras ferramentas para os testes com o que chamaremos de sistema 2.

4.2.4 Configuração Simulink

A partir desse ponto se faz necessário realizar a conexão do MATLAB/Simulink com o servidor para que ele possa receber e enviar dados do Factory IO, e para isso a tarefa será dividia em três partes:

4.2.4.1 Função MATLAB

Para realizar a conexão entre o Simulink e o servidor OPC, é necessário a criação de um bloco de função do MATLAB que será o responsável por se conectar ao servidor, obter os dados necessários para o controlador e enviar a resposta do controlador para a planta. A função do MATLAB irá receber como entrada a resposta do controlador e terá como saída o erro (*offset*), que é a diferença existente entre a variável de processo (que nesse caso é a leitura em tempo real do nível do tanque) e o *setpoint*. A leitura dos dados do servidor ocorre em tempo real, da mesma forma que os dados são enviados para a planta. A função se divide em:

- Definição e inicialização das variáveis de controle;
- Conexão com o servidor: o MATLAB possui componentes integrados que permitem que a conexão seja feita forma de rápida e direta, então só é necessário definir o endereço do servidor OPC, que pode ser encontrado nas configurações do servidor no TIA Portal;

```
1    uaClient = opcua('192.168.0.31',4840);
2    connect(uaClient);
```

- Obter o endereço das variáveis do servidor que serão utilizadas e inicializa-las;

```
1    Var_Node_In_LEVELMETER = opcuanode(3,"""
        LEVEL_METER_BUFF""",uaClient);
2
3    Var_Node_In_SETPOINT = opcuanode(3,"""SETPOINT_BUFF""",
        uaClient);
4
5    Var_Node_Out = opcuanode(3,"""PID_OUTPUT""",uaClient);
```

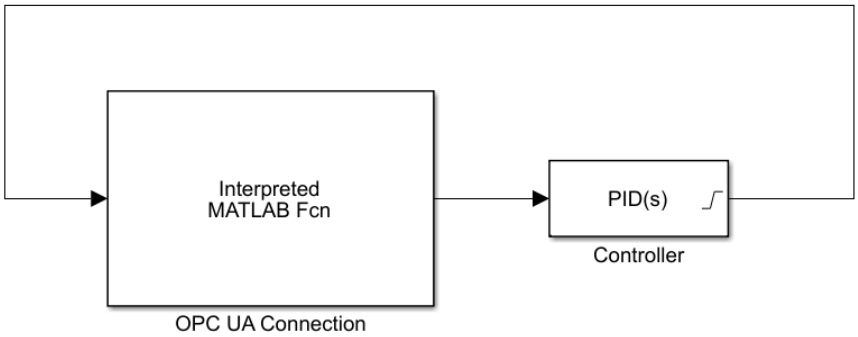
- Ler dos dados a partir das variáveis definidas anteriormente e enviá-las para o controlador. No fim, a função recebe a resposta dele e a envia para o servidor OPC.

```
1    [SETPOINT_AUX, ~, ~] = readValue(uaClient,
        Var_Node_In_SETPOINT);
2
3    [LEVELMETER, ~, ~] = readValue(uaClient,
        Var_Node_In_LEVELMETER);
4
5    writeValue(uaClient, Var_Node_Out, SAIDA_PID);
6    aux = SETPOINT_AUX - LEVELMETER;
7    final = double(aux);
8    SETPOINT = double(final);
```

4.2.4.2 Bloco Simulink

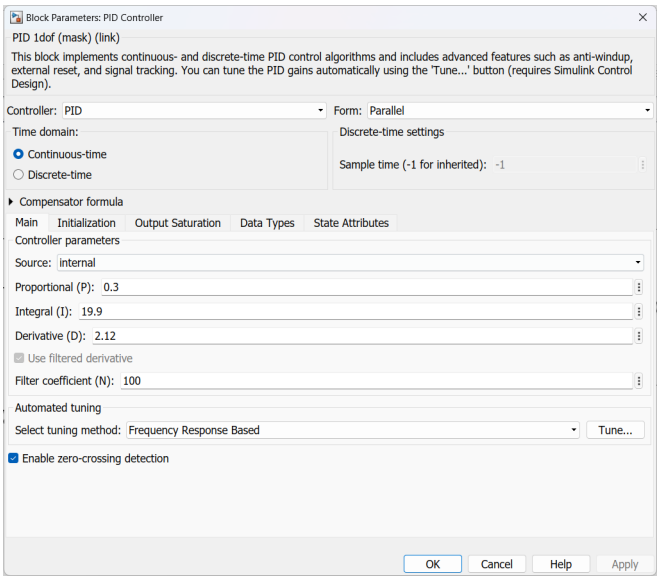
O próximo passo é criar o controlador em si e isso também pode ser feito de forma bem simples utilizando o bloco PID disponibilizado pelo Simulink. Realizamos as configurações necessárias, adicionando os parâmetros encontrados pelo *fine tunning* do TIA Portal e o software já é capaz de realizar o controle da planta.

Figura 41 – Captura de tela do controlador PID no Simulink



Fonte: Captura de tela de autoria própria.

Figura 42 – Captura de tela das configurações do controlador PID no Simulink

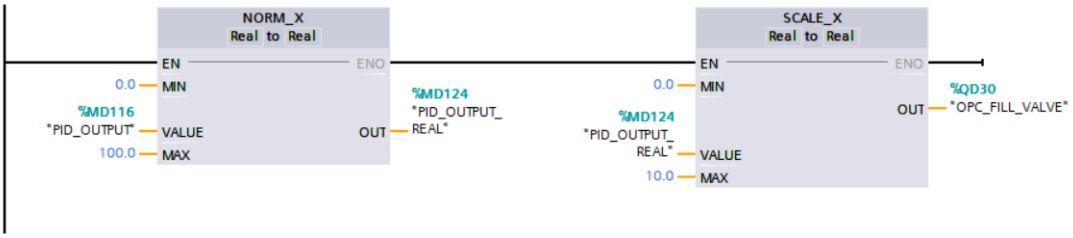


Fonte: Captura de tela de autoria própria.

4.2.4.3 Alteração no servidor OPC UA

Enquanto o controle for feito através do Simulink precisamos realizar uma pequena modificação no projeto do sistema 1, desativando o controlador PID utilizado anteriormente para que o controlador do Simulink assuma seu lugar. Também é necessário alterar o conversor de escala do controlador para que ele receba a resposta do controlador através do servidor OPC, que é enviada através do bloco de função do MATLAB criado anteriormente.

Figura 43 – Captura de tela das configurações do conversor de escala do controlador PID configurado para o Simulink.



Fonte: Captura de tela de autoria própria.

Toda conexão da planta é feita através do servidor e já foi configurada anteriormente, sem necessidade de realizar alterações. A partir desse ponto o nosso sistema 2 já pode realizar o controle da planta e podemos seguir para a análise dos resultados.

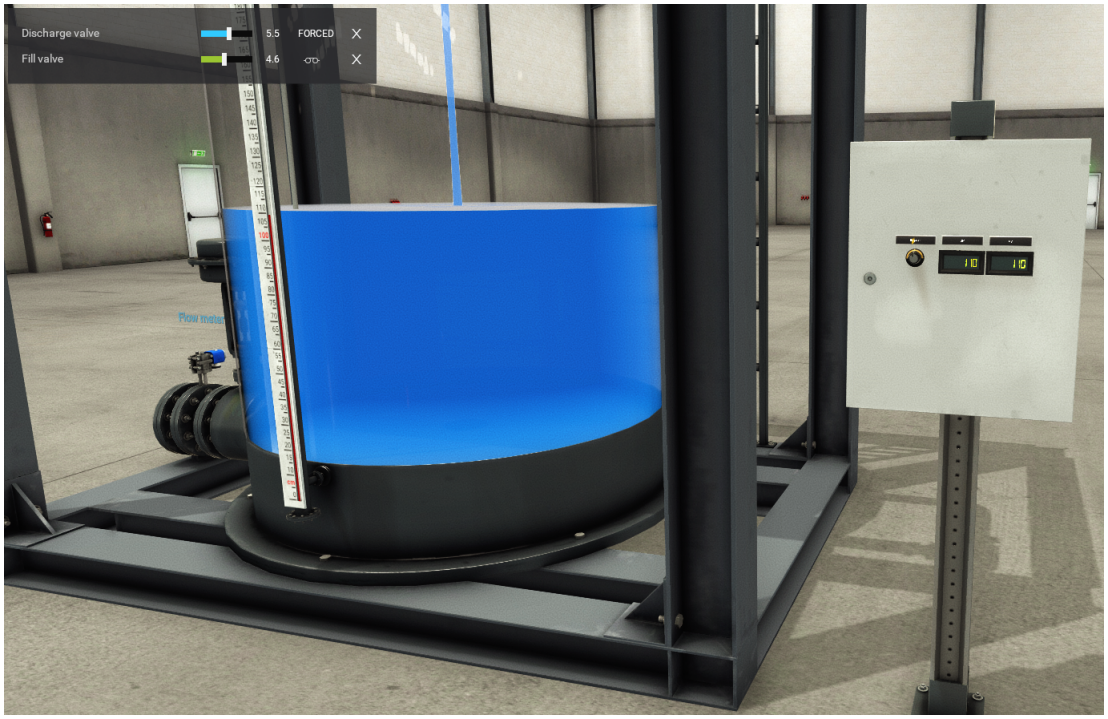
5 RESULTADOS

Ao finalizar as configurações necessárias no capítulo 4, já podemos realizar o controle da planta e realizar os testes necessários. Primeiramente analisaremos os resultados do sistema 1.

5.1 Controle via TIA Portal (Sistema 1)

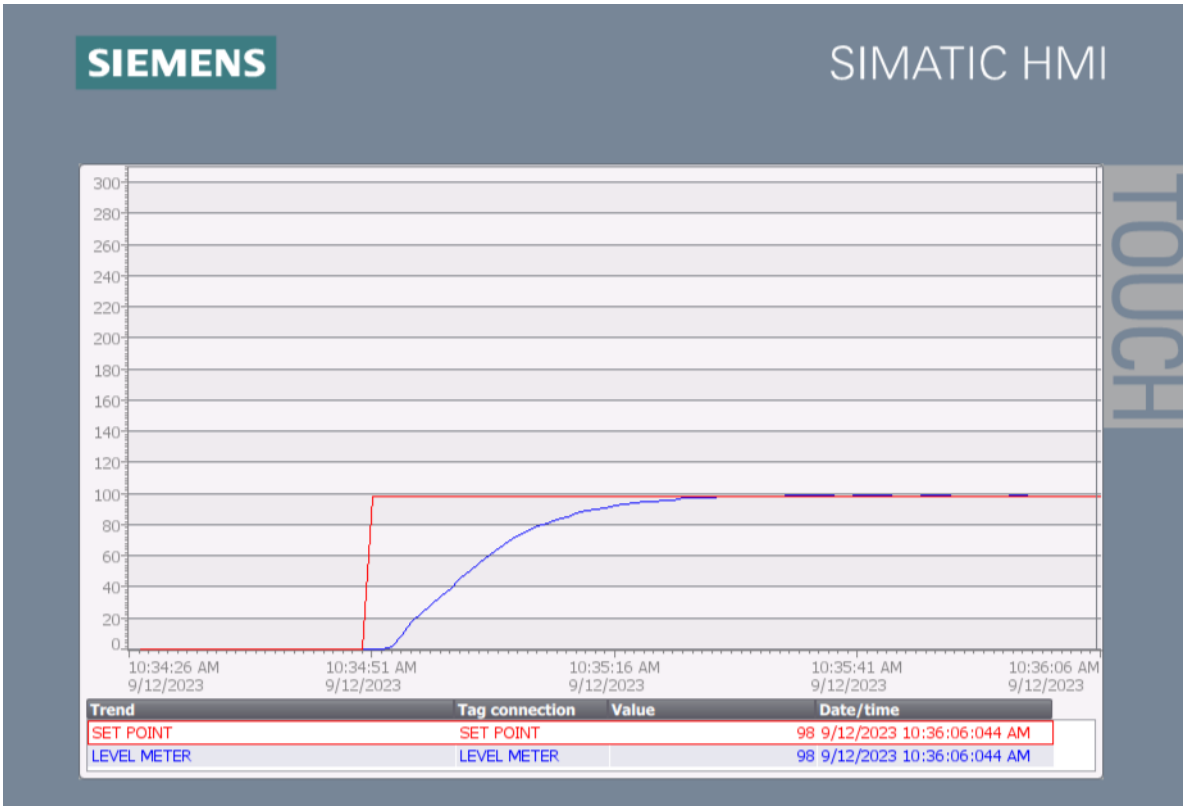
Iniciamos todas as ferramentas já configuradas no esquema do sistema 1 e começamos os testes. No primeiro momento queremos verificar se o sistema realiza o controle corretamente e para isso foi escolhido um valor arbitrário de *setpoint*, neste caso 98 cm. Aguardamos o funcionamento da planta e analisamos a resposta do controlador através da IHM, como podemos ver na Figura 45.

Figura 44 – Captura de tela do Factory IO com *setpoint* arbitrário



Fonte: Captura de tela de autoria própria.

Figura 45 – Captura de tela da resposta do controlador com *setpoint* no valor 98

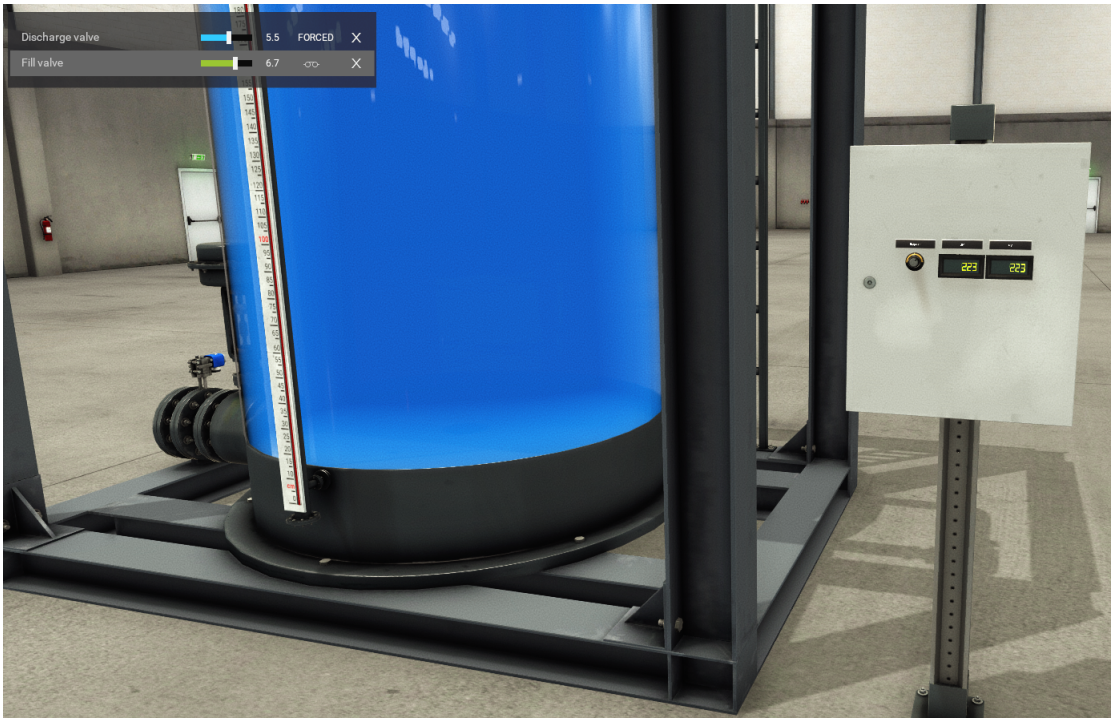


Fonte: Captura de tela de autoria própria.

A partir da captura de tela mostrada na Figura 45 podemos perceber que o controlador consegue atuar na planta de forma correta com os parâmetros encontrados pelo *auto-tunning* demonstrados na tabela 1. Analisaremos detalhadamente o funcionamento do controlador na seção 5.3.

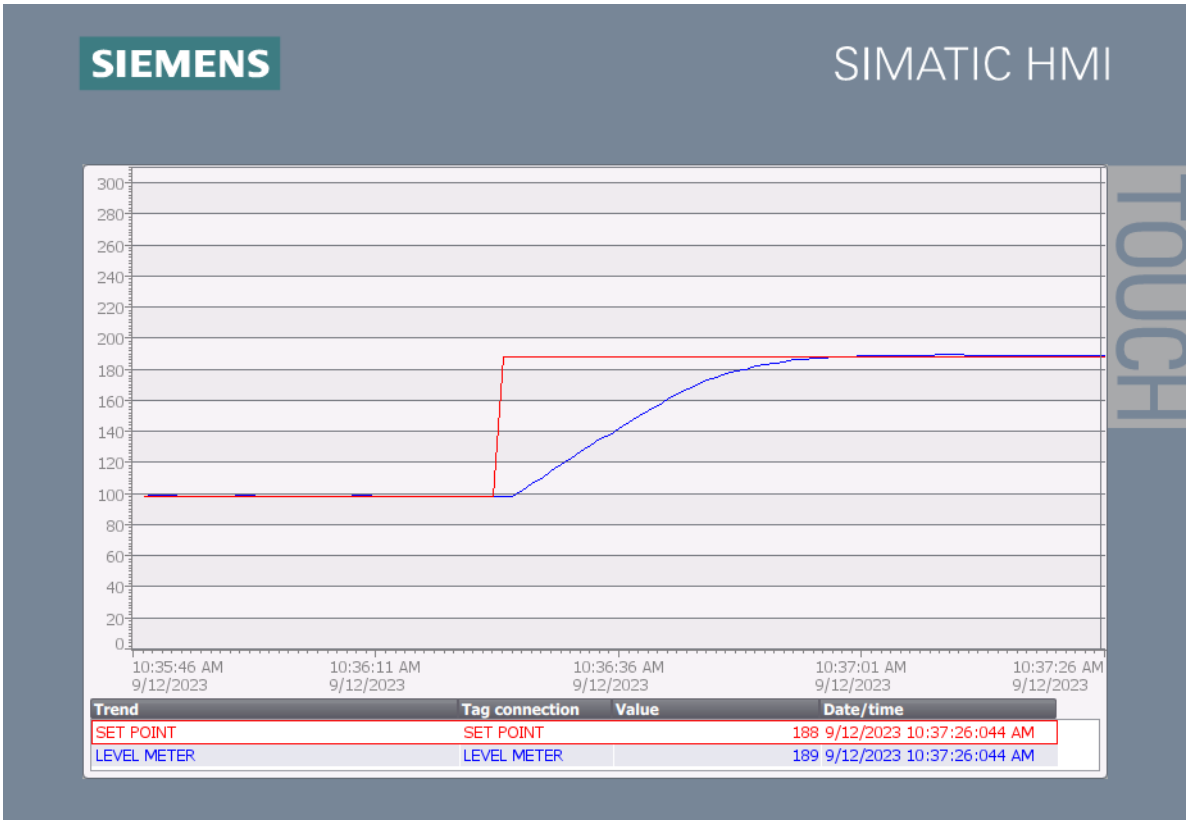
Em seguida, iremos verificar como o controlador se comporta com uma mudança de *set-point*. Nesse caso aumentamos seu valor para 188 cm, e conseguimos perceber que o controlador consegue acompanhar o valor de *setpoint* em tempo real.

Figura 46 – Captura de tela do Factory IO com *setpoint* definido no valor 188



Fonte: Captura de tela de autoria própria.

Figura 47 – Captura de tela da resposta do controlador com *setpoint* no valor 188

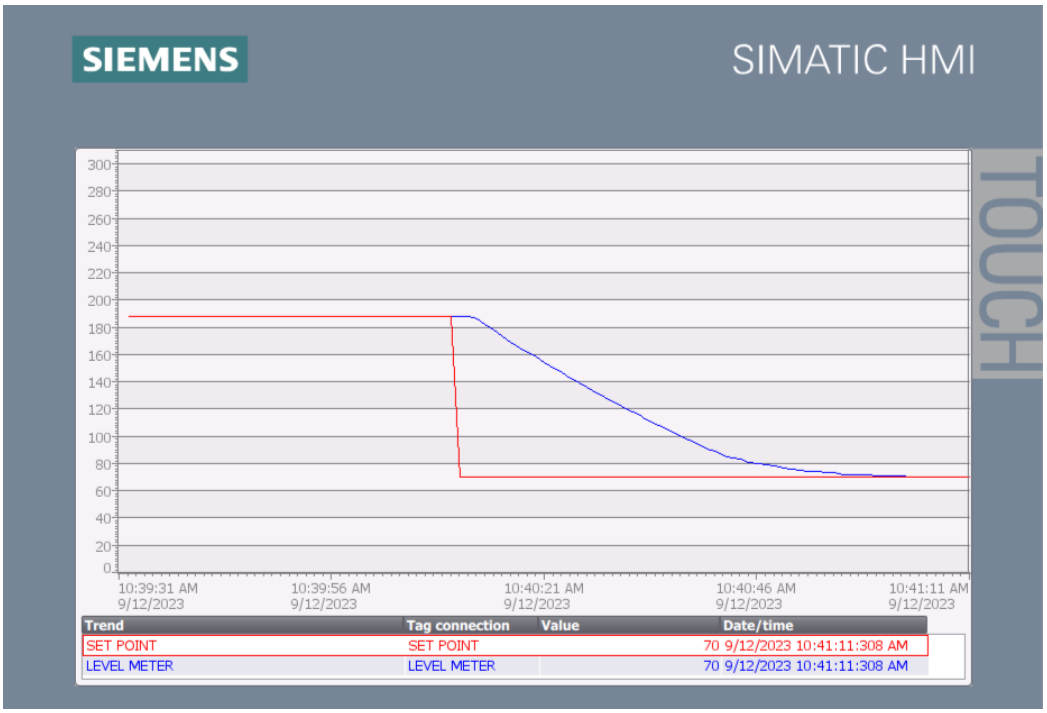


Fonte: Captura de tela de autoria própria.

Analisando a Figura 47 percebemos que o controlador consegue alcançar o valor definido independente de qual seja o valor atual do nível d'água.

Por último, testaremos a sua capacidade de diminuir o nível do tanque. Na seção 4.2.3 definimos um valor fixo para a válvula de descarte da planta, permitindo uma vazão constante. Nesse último teste, queremos verificar se o controlador consegue trabalhar em conjunto com a válvula para diminuir o nível d'água. Foi alterado o valor de *setpoint* abaixo do definido anteriormente, nesse caso foi 70 cm (118 cm abaixo do *setpoint* anterior).

Figura 48 – Captura de tela da resposta do controlador após alteração do *setpoint* para 70



Fonte: Captura de tela de autoria própria.

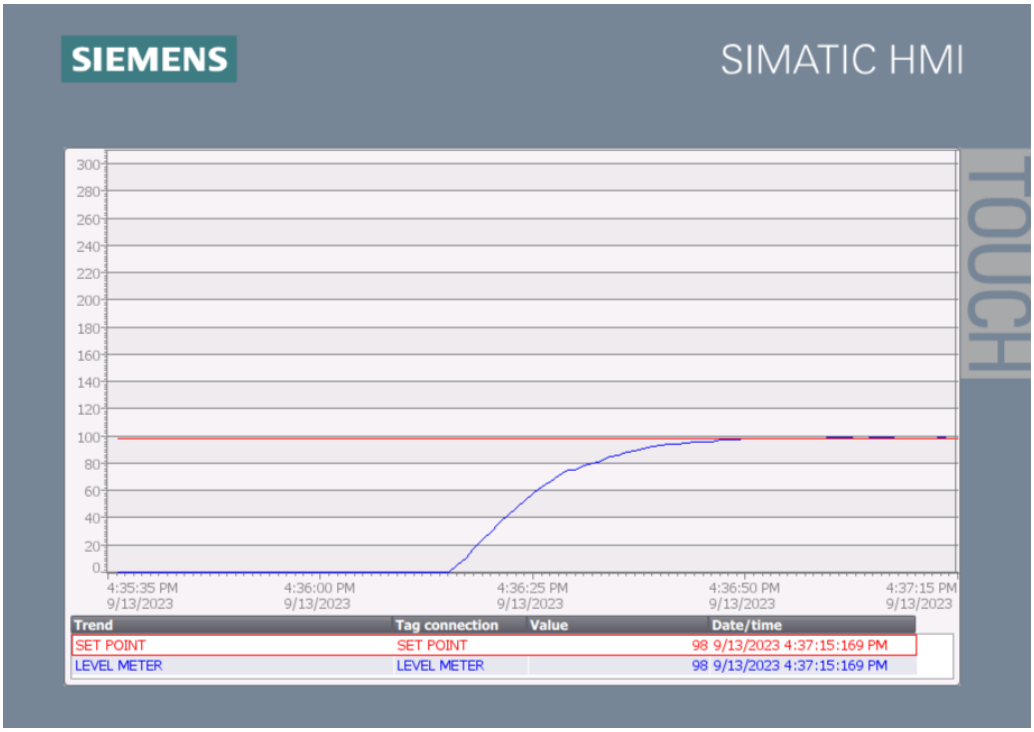
Fundamentado nesses testes, podemos concluir que o controlador PID configurado no TIA Portal consegue realizar o controle da planta no Factory IO sem nenhum problema.

Com o fim da análise inicial do sistema 1 mudaremos agora para o sistema 2, realizando os mesmos testes demonstrados anteriormente.

5.2 Controle via Simulink (Sistema 2)

Neste cenário, o projeto foi modificado para o funcionamento do sistema 2 (como demonstrado na seção 4.2.4) para darmos início aos testes. Seguiremos as mesmas etapas feitas na seção 5.1. Em um primeiro momento queremos verificar se o sistema 2 consegue realizar o controle corretamente da mesma forma do sistema 1 e para isso utilizaremos os mesmos valores de *setpoint* do sistema anterior, neste caso 98 cm. Aguardamos o funcionamento da planta e analisamos a resposta do controlador através da IHM na Figura 49.

Figura 49 – Captura de tela da resposta do controlador do Simulink com *setpoint* no valor 98



Fonte: Captura de tela de autoria própria.

Figura 50 – Captura de tela do Factory IO com *setpoint* definido no valor 98

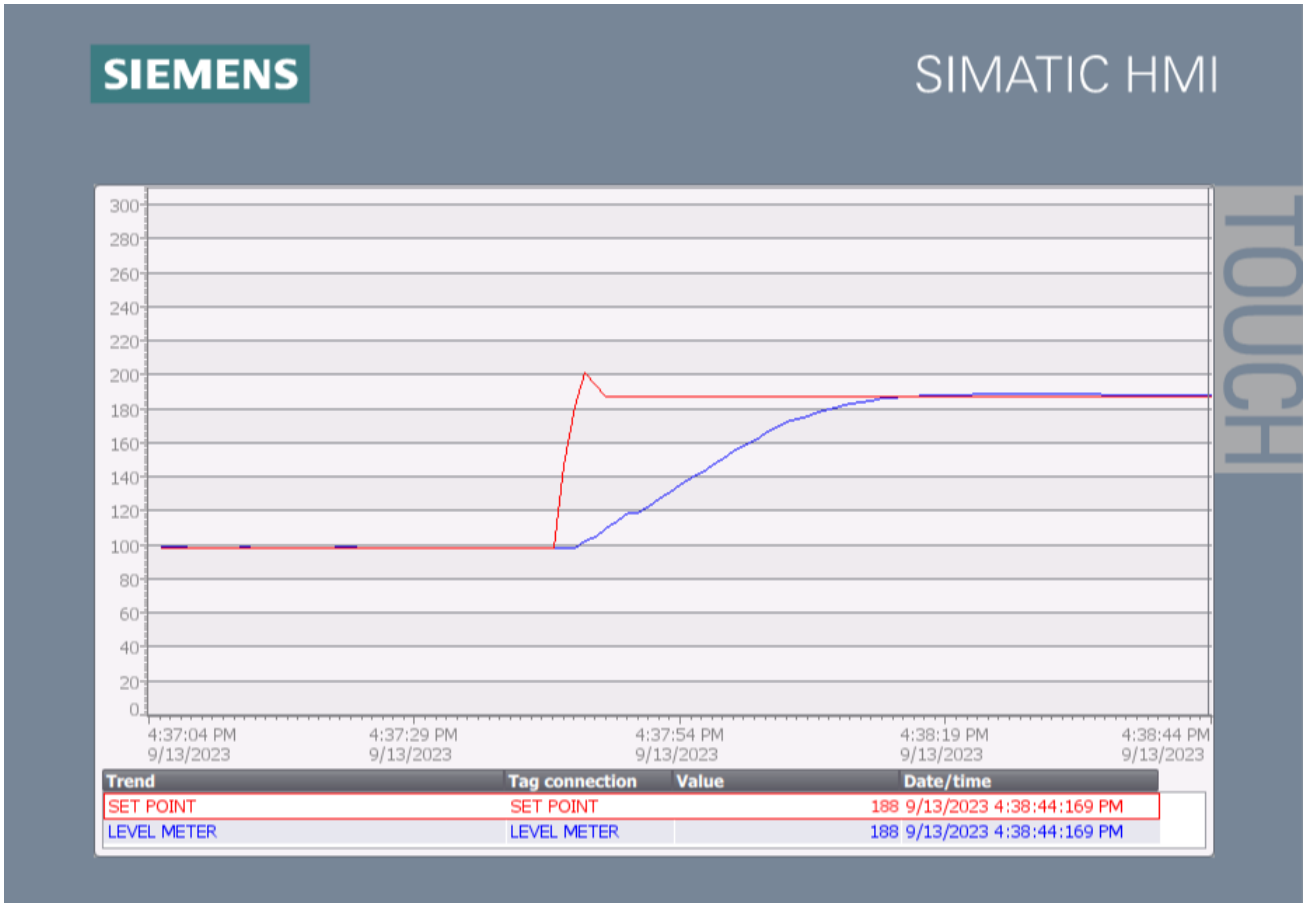
Fonte: Captura de tela de autoria própria.

A partir da captura de tela mostrada na Figura 49 podemos analisar que o controlador consegue atuar na planta de forma correta com os mesmos parâmetros do PID encontrados pelo *auto-tunning* do sistema 1, desta vez sendo o controlador do Simulink. No primeiro momento,

percebemos que o controlador de comporta quase que forma praticamente idêntica. Analisaremos com mais detalhes na seção 5.3.

Em seguida, da mesma forma feita com o sistema 1, iremos verificar como o controlador se comporta com uma mudança de *setpoint*. Nesse teste aumentamos o valor dele para o mesmo valor do teste feito no sistema 1, 188 cm. Percebemos através da Figura 51 que o controlador consegue acompanhar o valor de *setpoint* em tempo real, e de novo, de forma praticamente idêntica.

Figura 51 – Captura de tela da resposta do controlador do Simulink após alteração do *setpoint* para 188 cm

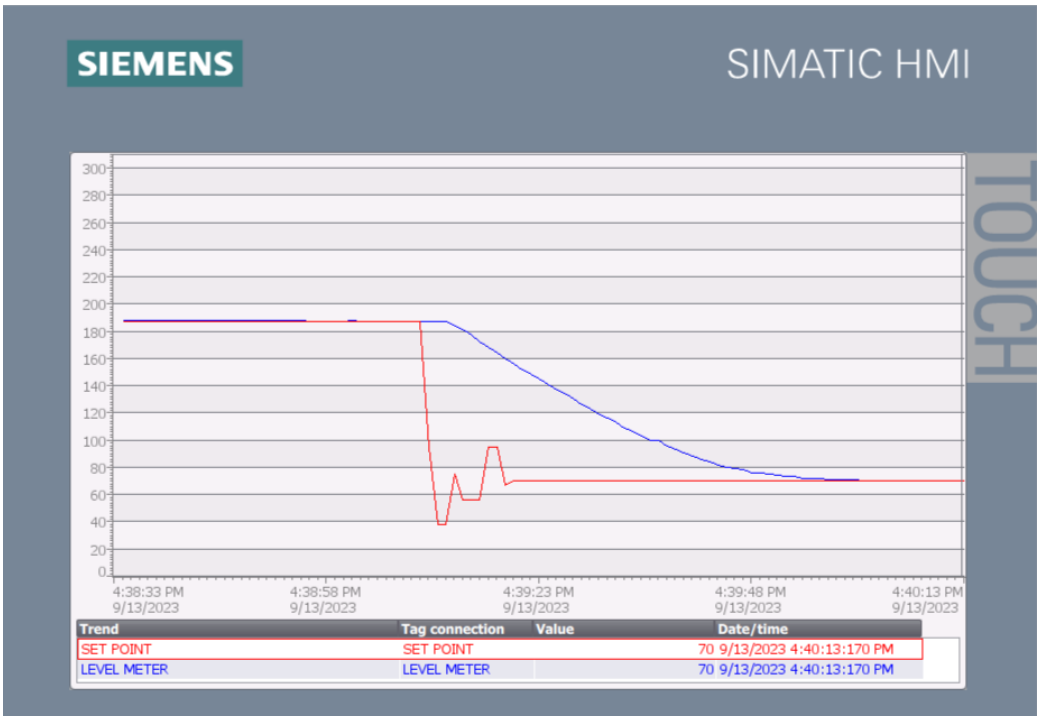


Fonte: Captura de tela de autoria própria.

Ao analisar a Figura 51, percebe-se que o controlador do Simulink também consegue alcançar o valor definido independente de qual seja o valor atual do nível d'água.

Por último testaremos a sua capacidade de diminuir o nível do tanque, já que queremos verificar se o controlador conectado com a planta através de um servidor OPC consegue se equiparar com a conexão feita diretamente. Desta vez, o sistema controlador está recebendo e enviando dados na planta em tempo real através do servidor OPC UA, onde a controlador deve trabalhar conjunto com a válvula para diminuir o nível d'água. Foi definido um valor de *setpoint* abaixo do utilizado anteriormente, nesse caso 70 cm (118 cm abaixo do valor anterior).

Figura 52 – Captura de tela da resposta do controlador do Simulink após alteração do *setpoint* para 70 cm



Fonte: Captura de tela de autoria própria.

Fundamentado nesses testes, podemos concluir que o sistema 2 que possui controlador PID configurado no Simulink com os mesmos parâmetros do sistema 1 e que se conecta através do OPC UA consegue realizar o controle da planta no Factory IO sem nenhum problema, da mesma forma que o sistema 1. Com os mesmos valores de *setpoint* o sistema se comporta de forma praticamente que idêntica.

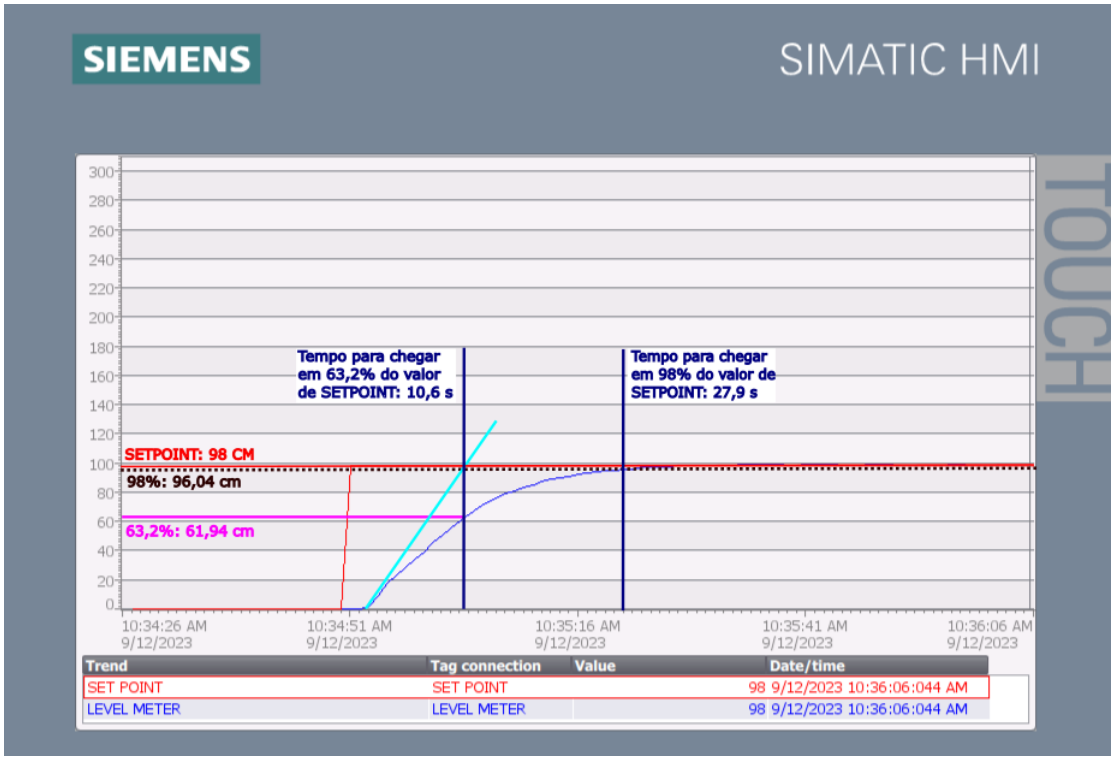
Na próxima seção iremos realizar uma análise detalhada de cada sistema e em seguida, faremos um comparativo dos dois.

5.3 Análise dos sistemas

Analisando a resposta do sistema a partir dos gráficos apresentados anteriormente, já temos uma ideia de que o funcionamento dos dois sistemas é praticamente idêntico. Pela resposta do controlador percebemos que se trata de um sistema de primeira ordem e por isso, conseguimos fazer a análise de acordo com a resposta ao degrau.

Primeiramente iremos verificar a resposta do sistema 1 saindo do nível 0 cm com *setpoint* 98 cm (Figura 53). Para o sistema alcançar 63,2% do valor de *setpoint* o controlador leva aproximadamente 10,6 segundos, e para alcançar 98% do valor de *setpoint* leva 27,9 segundos.

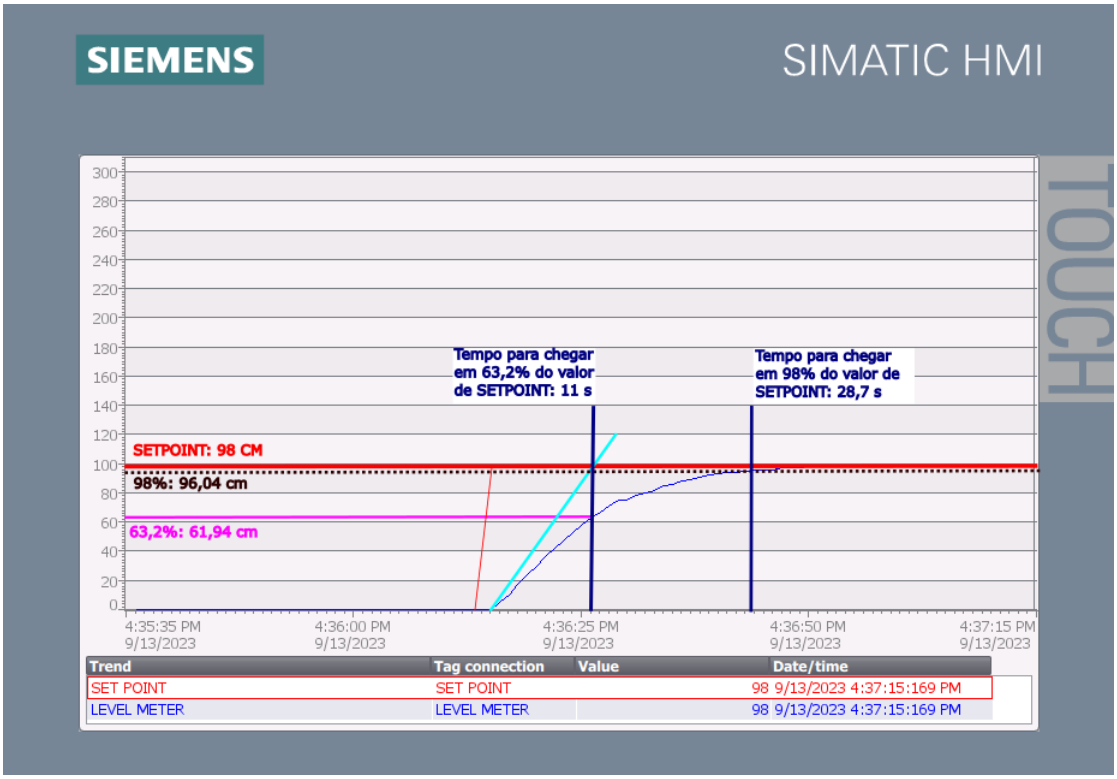
Figura 53 – Análise da resposta do controlador do TIA com *setpoint* 98 cm



Fonte: Captura de tela de autoria própria.

Comparando com o sistema 2 (Figura 54) confirmamos com dados que o sistema se comporta de forma praticamente idêntica. Para o sistema alcançar 63,2% do valor de *setpoint* o controlador leva aproximadamente 11 segundos, e para alcançar 98% do valor de *setpoint* leva 28,7 segundos.

Figura 54 – Análise da resposta do controlador do Simulink com *setpoint* 98 cm



Fonte: Captura de tela de autoria própria.

Podemos ver de forma detalhada na tabela 2 a comparação do tempo de subida de ambos os sistemas, e através desses dados confirmamos nossa suspeita de que os sistemas funcionam de forma praticamente idêntica.

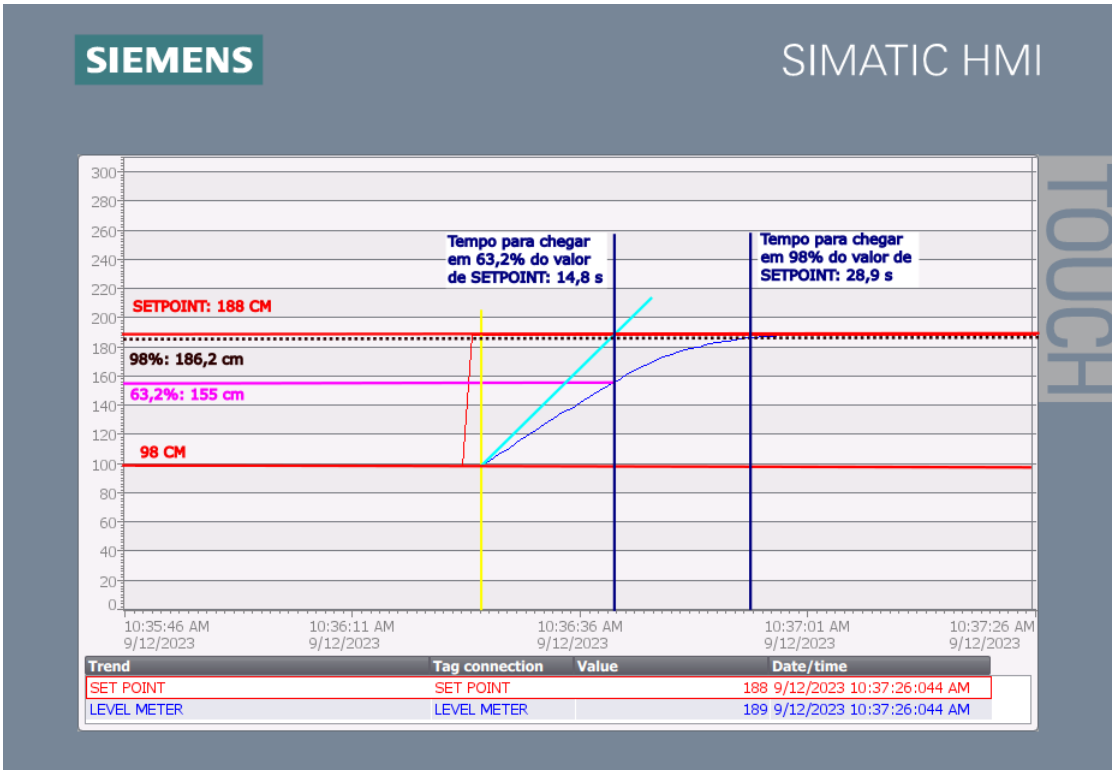
Tabela 2 – Tabela comparando os tempos de resposta dos sistemas nos testes com *setpoint* em 98 cm

	SISTEMA 1	**SISTEMA 2**
63,2 % (s)	10,6	11
98 % (s)	27,9	28,7

Fonte: Tabela de autoria própria.

Continuando nossos testes, analisaremos se o sistema se comporta de forma parecida alterando o valor do *setpoint* em tempo real. Primeiramente iremos verificar a resposta do sistema 1 saindo do nível 98 cm com *setpoint* 188 cm (Figura 56). Para o sistema alcançar 63,2% do valor de *setpoint* o controlador leva aproximadamente 14,8 segundos, e para alcançar 98% do valor de *setpoint* leva 28,9 segundos.

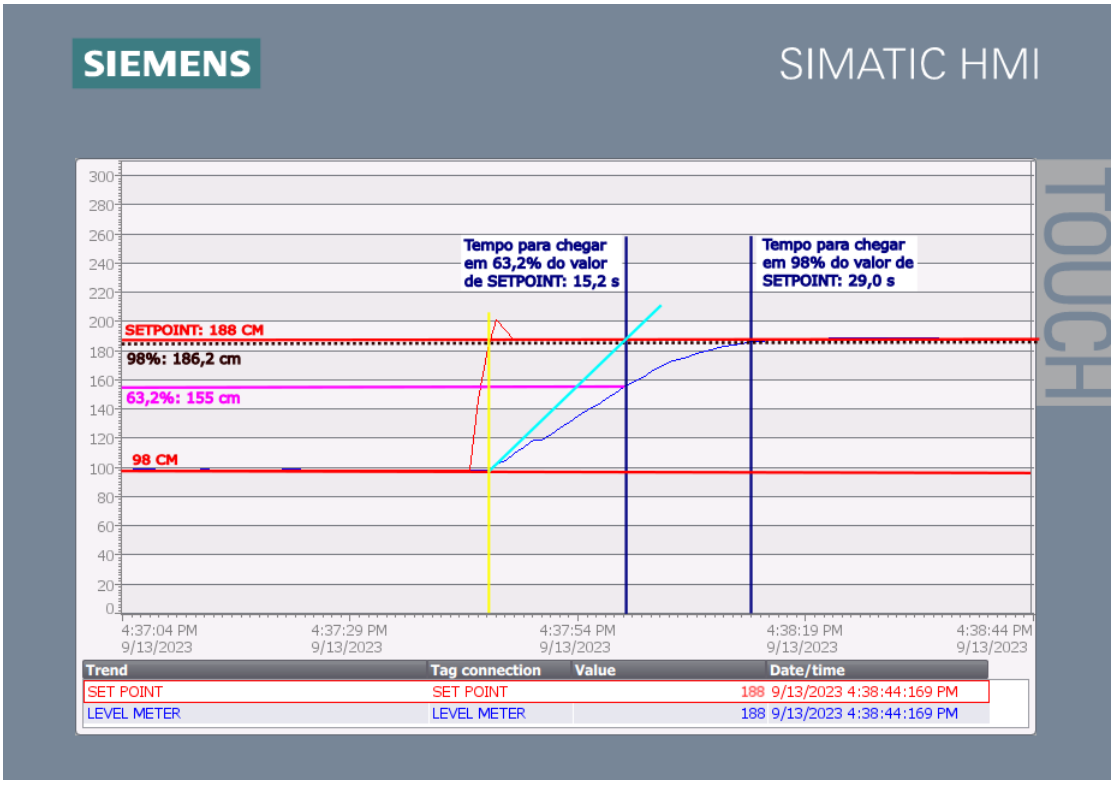
Figura 55 – Análise da resposta do controlador do TIA com *setpoint* 188 cm



Fonte: Captura de tela de autoria própria.

Comparando com o sistema 2 (Figura 56) confirmamos com dados que o sistema se comporta de forma parecida. Para o sistema alcançar 63,2% do valor de *setpoint* o controlador leva aproximadamente 15,2 segundos, e para alcançar 98% do valor de *setpoint* leva 29 segundos.

Figura 56 – Análise da resposta do controlador do Simulink com *setpoint* 188 cm



Fonte: Captura de tela de autoria própria.

Mais uma vez percebemos que o sistema esta funcionando de forma praticamente de forma idêntica. A diferença no tempo de resposta no máximo não passa de 0,4 segundos. Na tabela 4 vemos a diferença do tempo de subida de ambos os sistemas, e através desses dados confirmamos novamente que os sistemas funcionam de forma igual.

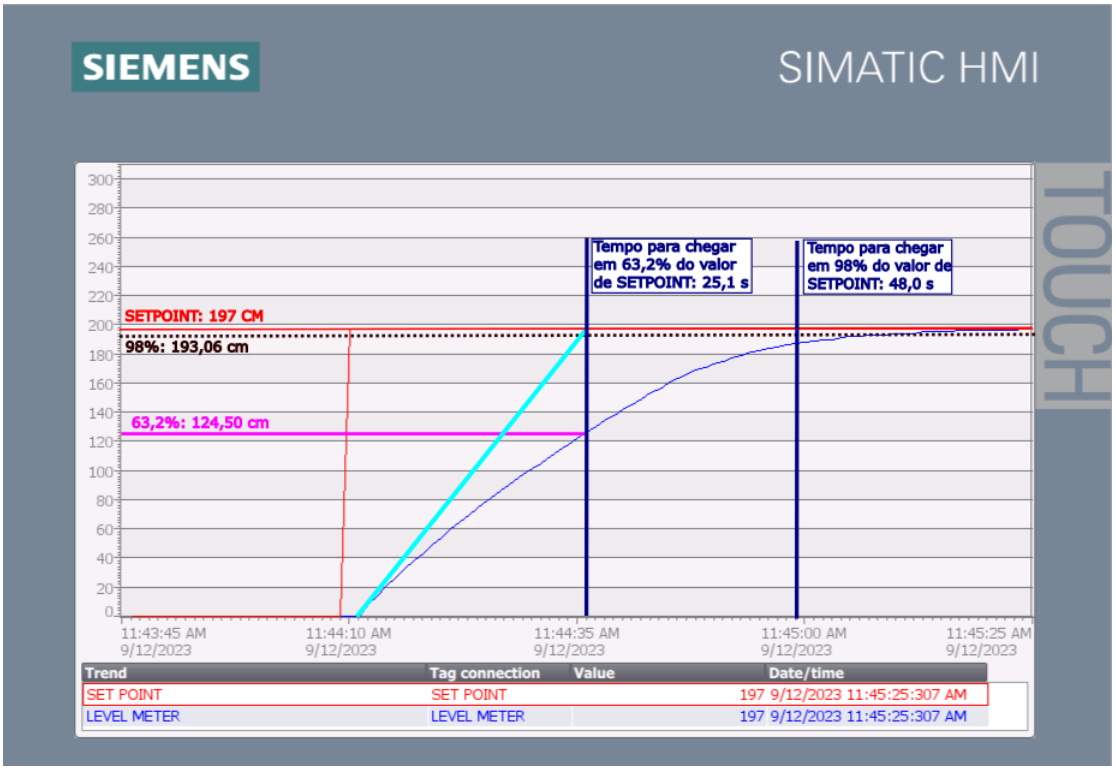
Tabela 3 – Tabela comparando os tempos de resposta dos sistemas nos testes com *setpoint* em 188 cm

	SISTEMA 1	**SISTEMA 2**
63,2 % (s)	14,8	15,2
98 % (s)	28,9	29,0

Fonte: Tabela de autoria própria.

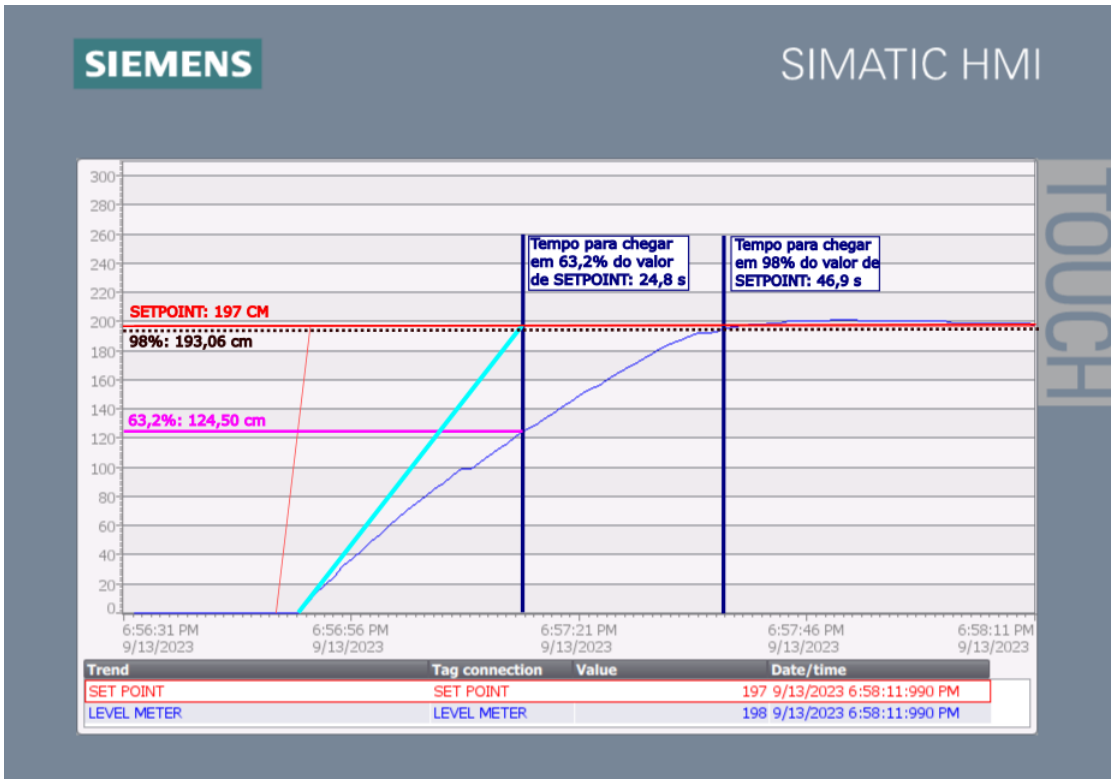
Por último, escolhemos um valor arbitrário de *setpoint* para confirmar o bom funcionamento do sistema. Neste caso, foi o valor 197 cm.

Figura 57 – Análise da resposta do controlador do TIA com *setpoint* 197 cm



Fonte: Captura de tela de autoria própria.

Figura 58 – Análise da resposta do controlador do Simulink com *setpoint* 197 cm



Fonte: Captura de tela de autoria própria.

Tabela 4 – Tabela comparando os tempos de resposta dos sistemas nos testes com *setpoint* em 188 cm

	SISTEMA 1	**SISTEMA 2**
63,2 % (s)	25,1	24,8
98 % (s)	48,0	46,9

Fonte: Tabela de autoria própria.

6 CONCLUSÃO

6.1 Considerações Finais

A proposta deste trabalho é a criação de uma rede industrial com softwares que não possuem o mesmo protocolo de comunicação nativo e integrá-los através do padrão OPC UA, possibilitando a troca de informações e de comandos a partir de qualquer um dos componentes. O OPC UA é um protocolo de comunicação aberto e independente que oferece uma série de benefícios para sistemas de automação industrial, como interoperabilidade, flexibilidade e segurança.

Neste trabalho a rede criada consistiu em uma planta de controle de tanques simulada através do Factory IO e controlada por um PID no SIMULINK, onde toda a comunicação entre eles foi feita utilizando um servidor OPC UA. O objetivo foi comparar o desempenho do controlador em dois cenários distintos: o primeiro onde o controle PID da planta foi feito diretamente por um PLC, e o segundo onde o controle PID foi feito pelo SIMULINK, sendo a conexão com a planta realizada completamente através do padrão OPC UA. No segundo cenário, testamos a capacidade do padrão utilizado de enviar e receber comandos de forma rápida e segura.

Na análise de resultados, os testes realizados demonstraram que o padrão OPC UA consegue reproduzir os resultados de forma praticamente idêntica, sem perda de desempenho. Com isso, podemos afirmar que o padrão OPC UA é uma tecnologia eficiente e segura para a integração de sistemas de automação industrial. Este trabalho traz contribuições no sentido de demonstrar as vantagens do padrão OPC UA em relação aos protocolos proprietários e para aplicar o padrão OPC UA em um cenário da indústria 4.0.

6.2 Trabalhos Futuros

Pretendemos expandir as funcionalidades do sistema, dando ênfase na integração de sistemas de automação industrial de diferentes fabricantes. Para tanto, pretendemos atingir os seguintes objetivos:

- Realizar o controle de várias plantas simultaneamente através do mesmo servidor OPC UA;
- Obter e analisar métricas de vários processos de forma simultânea, podendo alterar parâmetros do sistema em tempo real;
- Realizar os testes em uma planta física, utilizando um PLC real.

REFERÊNCIAS

MAHNKE, W.; LEITNER, S.-H.; DAMM, M. OPC Unified Architecture. 2009. ed. Berlim, Germany: Springer, 2009.

DESENVOLVIMENTO DE SERVIDORES, O.; DA, O.; UA E WRAPPERS PARA APLICAÇÃO EM, A. OPC UA e Wrappers para aplicação em Automação. [s.d.].

OSTROWSKI, I. et al. Lesson learned from semester of online teaching of automation using simulators. **IOP conference series. Materials science and engineering**, v. 1016, n. 1, p. 012005, 2021.

BAUER, H. et al. Hardware implementation of an OPC UA server for industrial field devices. IEEE transactions on very large scale integration (VLSI) systems, v. 29, n. 11, p. 1998–2002, 2021.

CAVALIERI, S.; SALAFIA, M. G.; SCROPPO, M. S. Integrating OPC UA with web technologies to enhance interoperability. **Computer standards & interfaces**, v. 61, p. 45–64, 2019.

JOHN, K.-H.; TIEGELKAMP, M. (EDS.). **Iec 61131-3: Programming Industrial Automation Systems**. New York, NY: Springer, 2014.

OnRemote Real-time Communication between MATLAB and PLC Based on OPC Technology. [s.l: s.n.].

BARBOSA, C. F. **MÉTODO PARA MEDIÇÃO DE TEMPOS EM OPERAÇÕES OPC UA NO ACESSO A DADOS DE TEMPO REAL DE UM SISTEMA DE AUTOMAÇÃO.** Florianópolis: Universidade Federal de Santa Catarina, 2018.

ALPHONSUS, E. R.; ABDULLAH, M. O. A review on the applications of programmable logic controllers (PLCs). **Renewable and Sustainable Energy Reviews**, v. 60, p. 1185–1205, 2016.

GAMES, R. OPC client DA/UA - FACTORY I/O. Disponível em: <https://docs.factoryio.com/manual/drivers/opc>. Acesso em: 13 oct. 2023.

OPC standards communication. Disponível em: <https://www.mathworks.com/help/icomm/opc.html>. Acesso em: 13 oct. 2023.

Unified Architecture. Disponível em: <https://opcfoundation.org/about/opc-technologies/opc-ua/>. Acesso em: 13 oct. 2023.

TIA Portal Siemens - Software SIMATIC da Siemens. Disponível em: <https://www.siemens.com/br/pt/produtos/software/industria/automacao/tia-portal.html>. Acesso em: 13 oct. 2023.

MATLAB. Disponível em: <https://www.mathworks.com/products/matlab.html>. Acesso em: 13 oct. 2023.

HEYNICKE, R. et al. IO-Link Wireless enhanced factory automation communication for Industry 4.0 applications. **Journal of sensors and sensor systems**, v. 7, n. 1, p. 131–142, 2018.

SHYR, W.-J. et al. Application of cyber-physical system technology on material color discrimination. **Electronics**, v. 11, n. 6, p. 920, 2022.