INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA – CAMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE METAL-MECÂNICA
PROGRAMA DE PÓS-GRADUAÇÃO *STRICTO SENSU* EM MECATRÔNICA

MATHEUS AUGUSTO MARTIM

IMPLEMENTAÇÃO DE UM MIDDLEWARE PARA INTEGRAÇÃO DE SISTEMAS DE UMA ESTAÇÃO DE TRATAMENTO DE EFLUENTES

FLORIANÓPOLIS, 2019.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA – CAMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE METAL-MECÂNICA
PROGRAMA DE PÓS-GRADUAÇÃO *STRICTO SENSU* EM MECATRÔNICA

MATHEUS AUGUSTO MARTIM

IMPLEMENTAÇÃO DE UM MIDDLEWARE PARA INTEGRAÇÃO DE SISTEMAS DE UMA ESTAÇÃO DE TRATAMENTO DE EFLUENTES

Dissertação submetida ao Instituto Federal de Santa Catarina como parte dos requisitos para obtenção do título de Mestre em Mecatrônica.

Orientador: Prof. Roberto Alexandre Dias

FLORIANÓPOLIS, 2019.

CDD 628.54
M378i

Martim, Matheus Augusto
Implementação de um middleware para integração de sistemas de uma estação de tratamento de efluentes [DIS] / Matheus Augusto Martim; orientação de Roberto Alexandre Dias – Florianópolis, 2019.

1 v.: il.

Dissertação de Mestrado (Mecatrônica) – Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Inclui referências.

1. Middleware. 2. Controle e automação. 3. Integração de sistemas. I. Dias, Roberto Alexandre. II. Título.

Sistema de Bibliotecas Integradas do IFSC
Biblioteca Dr. Hercílio Luz – Campus Florianópolis

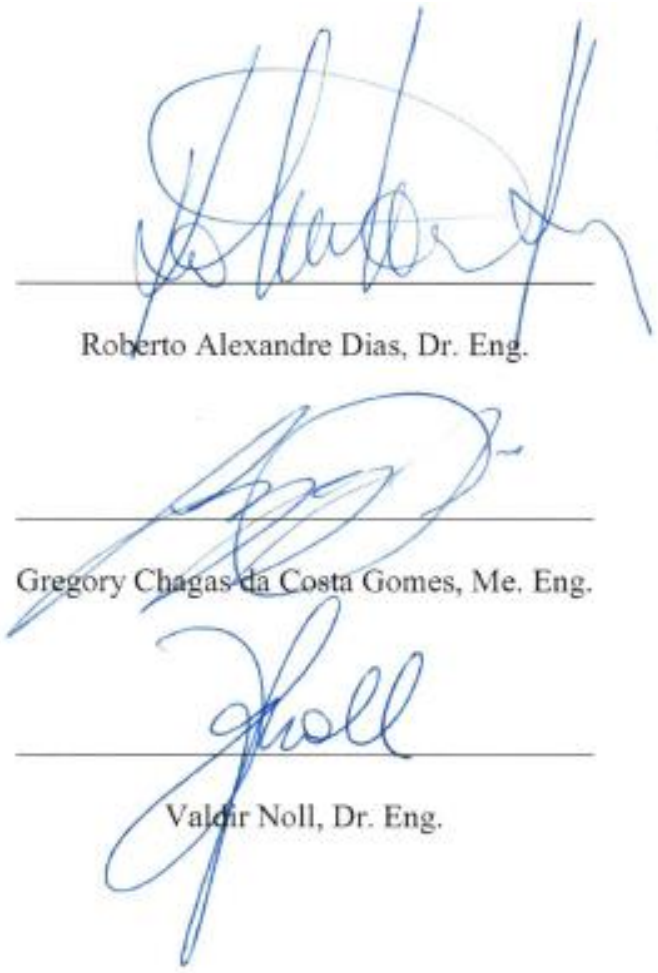
IMPLEMENTAÇÃO DE UM MIDDLEWARE PARA INTEGRAÇÃO DE SISTEMAS DE UMA ESTAÇÃO DE TRATAMENTO DE EFLUENTES

MATHEUS AUGUSTO MARTIM

Este trabalho foi julgado adequado para obtenção do Título de Mestre em Mecatrônica e aprovado na sua forma final pela banca examinadora do curso de Mestrado em Mecatrônica do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 28 de novembro de 2019.

Banca Examinadora:



Roberto Alexandre Dias, Dr. Eng.

Gregory Chagas da Costa Gomes, Me. Eng.

Valdir Noll, Dr. Eng.

RESUMO

A presente dissertação de mestrado traz o problema da interoperabilidade entre equipamentos e softwares industriais, mostrando a relevância de se adotar medidas e padronizações que permitam e facilitem a troca de informações.

O autor inicia por abordar as tecnologias envolvidas para a supervisão e controle da planta em estudo, detalhando os sistemas supervisórios mais utilizados nas principais concessionárias de água e esgoto do país. Também são abordados os protocolos industriais que historicamente vêm sendo utilizados nessas aplicações e por fim se encontram os *middlewares*, o objeto principal de estudo.

A proposta deste trabalho consiste em implementar um *middleware* que ofereça a possibilidade de interfaceamento entre um sistema supervisório de uma estação de tratamento de esgoto e um software proprietário que irá manipular em conjunto dados e controles da planta.

O trabalho prossegue com a implementação do *middleware*, utilizando o padrão de comunicação OPC *Classic* e abordando as vantagens, desvantagem e futuras otimizações da tecnologia do padrão de comunicação escolhido.

Este projeto será implementado na estação de tratamento de esgotos em Joinville, na unidade de Jarivatuba e está alinhado com a empresa Pesco Automação & Controle, que deverá realizar a entrega juntamente com a Companhia Águas de Joinville (CAJ) no segundo semestre de 2019.

Palavras-chave: *middleware*, OPC, controle e automação, integração de sistemas.

ABSTRACT

This master thesis brings the problem of interoperability between equipment and industrial softwares, showing the relevance of adopting measures and standardizations that allow and facilitate the exchange of information.

The author begins by addressing the technologies involved in the supervision and control of the plant under study, detailing the most used supervisory systems in the main water and sewage dealers in the country. Also covered the industrial protocols that have have been historically used in these applications and finally the middleware, the main object of study.

The purpose of this paper is to implementation a middleware that offers the possibility of interfacing between a supervisory system of a sewage treatment plant and a proprietary software that will jointly manipulate plant data and controls.

The project continues with the implementation of middleware, using the OPC communication standard and addressing the advantages, disadvantages and future optimizations of the technology of the chosen communication standard.

This project will be implemented at the Joinville sewage treatment plant in the Jarivatuba unit and is aligned with Pesco Automação & Control, which is expected to be delivered with Companhia Águas de Joinville (CAJ) during the second half of 2019.

Keywords: middleware, OPC UA, control and automation, systems integration.

Lista de Figuras

Lista de tabelas

Lista de abreviaturas

CPS	*Cyber-Physical Systems*
IoT	*Internet of Things*
MDIC	Ministério do Desenvolvimento da Indústria e Comércio
BNDES	Banco Nacional de Desenvolvimento Econômico e Social
OLE	*Object Linking and Embedding*
OPC	*OLE for Process Control*
COM	*Component Object Model*
DCOM	*Distributed Component Object Model*
SCADA	S*upervisory Control and Data Acquisition*
CLP	*Control Logic Program*
IHM	Interface Homem Máquina
MÊS	*Manufacturing Execution Systems*
ERP	*Enterprise Resourse Planning*
NASA	*National Aeronautics and Space Administration*
HTTP	*Hypertext Transfer Protocol*
FTP	*File Transfer Protocol*
SMTP	*Simple Mail Transfer Protocol*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
ISSO	*International Standards Organization*
OSI	*Open System Interconnection*
ISA	*International Society of Automation*
UTP	*Unshielded Twisted Pair*
STP	*Shielded Twisted Pair*

RPC	*Remote Procedure Call*
MOM	M*essage-Oriented Middleware*
ORB	*Object Request Brokers*
COM	*Component Object Model*
DCOM	*Distributed Component Object Model*
XML	*Extensible Markup Language*
HDA	*Historical Data Access Specification*
DA	*Data Access Specification*
AE	*Alarm & Events Specification*
OPC UA	*Unified Architecture*
RE	Reator
SOP	Soprador
PH	Potencial Hidrogeniônico
VM	Válvula Motorizada
MB	Motobomba
PAC	Painel de Automação Central
CPU	*Central Processing Unit*
POU	*Program Organization Unit*

SUMÁRIO

1. Introdução

O atual momento de transformação da tecnologia dentro da indústria, mostra que a conectividade cada vez mais permite a interligação entre o ambiente gerencial com as informações fabris de forma rápida e transparente, permitindo que as decisões de produção e manutenção sejam facilmente automatizadas. Essa nova indústria está sendo denominada de Industria 4.0 na Alemanha, *Industrial Internet* pelos Norte-Americanos e Manufatura Avançada no Brasil.

A Industria 4.0 é baseada em conceitos de sistemas *Cyber-Physical Systems* (CPS – ciber-físicos) e também conta com novas tecnologias onde se destacam a Internet das Coisas (*IoT*), *big data analytics* e a computação em nuvem. Tecnologias digitais que juntas promovem e permitem a rápida personalização e fabricação de produtos, sempre respondendo de forma ágil à produção conforme a demanda de suprimentos (Schwab, 2017).

O Brasil, por meio do Ministério do Desenvolvimento da Indústria e Comércio (MDIC) abriu em 2016, junto ao Banco Nacional de Desenvolvimento Econômico e Social (BNDES), dois programas de incentivo à Manufatura Avançada. O primeiro (BNDES ProBK), visa incentivar o aumento da capacidade produtiva, a modernização das instalações, novas fusões e aquisições. O segundo (BNDES Funtec), visa apoiar financeiramente projetos que estimulem o desenvolvimento tecnológico e a inovação de interesse estratégico para o Brasil.

Segundo (Azevedo, 2017), o grupo entrevistado, referente a região metropolitana de Campinas em São Paulo, região esta representada por indústrias de automação, química, petroquímica, borracha, alimentação, celulose e papel, montadoras, dentre outras, mostra desconhecimento da transformação digital no Brasil e também demonstra resistência à aplicação de novos conceitos tecnológicos na indústria, indicando a dificuldade que se encontra na implementação de novas tecnologias na indústria brasileira.

A crescente necessidade de interação com sistemas industriais fechados, para se obter controle e informações do processo, tem se mostrado uma das principais necessidades dessa nova indústria, e esse vem a ser o objeto de estudo desse trabalho, que se apropria de uma planta industrial de tratamento de efluentes que conta com um sistema SCADA (*Supervisory Control and Data Acquisition*). Para este sistema, deve-se desenvolver um *middleware* capaz de fornecer informações e permissões de acesso e controle a outros sistemas, sejam eles gerenciais ou industriais.

Os *middlewares*, são softwares capazes de mediar a comunicação entre sistemas com diferentes protocolos de comunicação. Existe uma grande aceitação da indústria por uma interface padrão entre os produtos e os desenvolvedores, conhecida por OPC (*OLE for Process Control*) que a maioria dos fabricantes de softwares de desenvolvimento SCADA, CLP (*Control Logic Program*),

IHM (Interface Homem Máquina), MES (*Manufacturing Execution Systems*), ERP (*Enterprise Resourse Planning*) adotam e oferecem como interface de comunicação entre cliente e servidor, para a obtenção de informações e controle (Gonçalves, 2012).

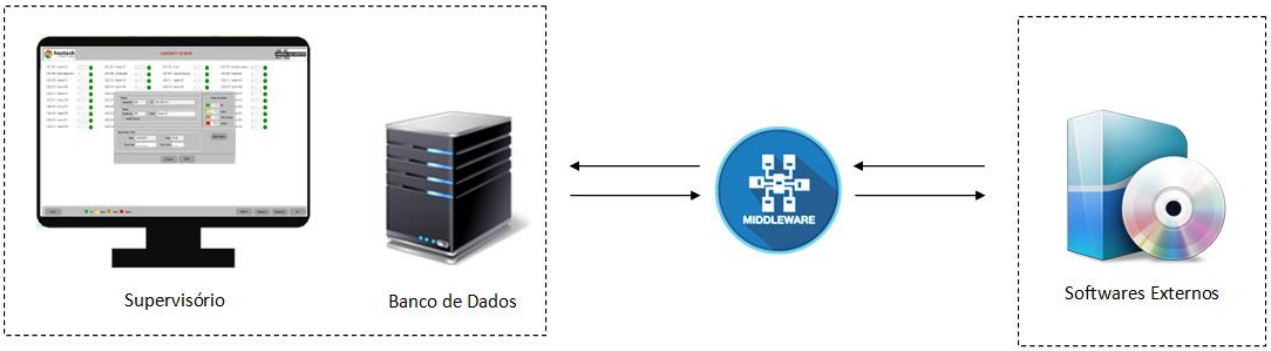
1.1 Definição do Problema

A Pesco Automação & Controle é uma empresa sediada em Florianópolis e fornecedora de soluções em automação e controle de indústrias, e desenvolveu um sistema supervisório responsável por controlar a estação de tratamento de efluentes (ETE), unidade de Jarivatuba, na cidade de Joinville. Esse sistema supervisório é responsável pelo controle e monitoramento de motobombas, válvulas motorizadas, sopradores de ar, sensores de controle de qualidade da água, entre outros equipamentos necessário para o tratamento de efluentes.

Além do controle da ETE, o supervisório deverá se comunicar com um software específico de tratamento de efluentes adquirido pela empresa Águas de Joinville, responsável pelo tratamento da água e efluentes na cidade, e deverá fornecer permissões de acesso e controle a variáveis específicas para o tratamento do efluente. Por uma questão de segurança, interoperabilidade e permeabilidade de comunicação, optou-se por desenvolver um middleware com padrão de comunicação OPC, garantindo a confiabilidade e segurança de informações de ambos os desenvolvedores de softwares envolvidos na aplicação.

A ***Figura 1*** ilustra o interfaceamento realizado pelo *middleware* entre o sistema supervisório e o software de tratamento de efluentes.

Figura 1 - Esquemático da interface entre o supervisório e o software de tratamento de efluentes



Fonte: Própria

O *middleware* terá como principais funções gerenciar o acesso de usuários, garantindo o nível de permissão de troca de informações, gerir a troca de dados entre clientes e servidor e fornecer uma proposta de padronização do modelo implementado no servidor OPC para o controle de ETEs.

O modelo proposto irá agregar padronização à empresa Pesco que desenvolve essa mesma solução também para outros clientes que tratam água e esgoto. Dessa maneira se pretende implementar por meio de uma Normatização Técnica Interna (NTI), o modelo proposto para a comunicação de dispositivos de campo por meio do padrão de comunicação OPC.

A implementação de NTI's tem mostrado ótimos resultados na Pesco, garantindo a gestão e duplicação da informação e do conhecimento desenvolvido pelos profissionais. Essa medida se realiza pelo preenchimento de um documento denominado Detalhamento Técnico de Solução do Projeto (DTSP) que se torna disponível a todos os colaboradores. Este documento contém tais informações: detalhamento do projeto, falhas e dificuldades encontradas na implementação, componentes utilizados, soluções e metodologia utilizada para correção de problemas, além de descrever o passo a passo de como o projeto foi desenvolvido.

1.2 Objetivos

Objetivo Geral

O objetivo deste trabalho é implementar um *middleware* capaz de realizar a interface entre um sistema supervisório e um software de tratamento de efluentes, utilizando o padrão de comunicação OPC *Classic*.

Objetivos Específicos

Para atender o objetivo principal, este trabalho terá como objetivos específicos os seguintes itens:

- Estudar o sistema de tratamento de efluentes, implementado na ETE Jarivatuba;
- Verificar quais variáveis de controle do processo serão necessárias disponibilizar no *middleware*;
- Implementar um *middleware* com padrão de comunicação OPC *Classic*;
- Desenvolver uma base de testes;
- Simular os testes;
- Validar os testes;
- Desenvolver a documentação do projeto.

1.3 Justificativa e Relevância

Atualmente, segundo a Companhia Águas de Joinville (CAJ) 31,5% do esgoto da cidade de Joinville é coletado e tratado. 95 % desse esgoto é tratado na ETE de Jarivatuba, que hoje possui a capacidade para tratar 490 l/s. A cidade mais populosa do estado de Santa Catarina, conforme pesquisa de 2017 do IBGE, com 577.077 habitantes, está investindo na ampliação do sistema de tratamento de esgotos com a construção de uma nova ETE, que substituirá a antiga estação e ampliará a capacidade de tratamento de esgoto para 600 l/s.

A empresa Pesco Automação & Controle, realizou o desenvolvimento da automação da nova ETE de Jarivatuba, fabricando painéis de distribuição e de automação para toda a planta. Além do fornecimento dos painéis, foi desenvolvido o sistema supervisório que gerencia toda a planta. Junto a esse sistema supervisório de controle a CAJ adquiriu um sistema específico de tratamento de esgotos, que necessita coletar dados e permissões de controles do sistema central de gerenciamento.

A existência de diversas soluções, de diversos fabricantes para um padrão de comunicação industrial, levou ao mercado a existência de uma grande quantidade de soluções incompatíveis entre si. Neste trabalho foi utilizado o protocolo Ethernet IP para comunicar as CPUs às IOs remotas, o protocolo UDP para conectar as CPUs, e Modbus para se comunicar com o Supervisório. Porém o mercado industrial cada dia se mostra mais conectivo, permitindo e facilitando o acesso das informações de campo a nível gerencial.

Este trabalho apresenta uma proposta de solução com o desenvolvimento de um *middleware*, utilizando o padrão de comunicação OPC *Classic*, que garante a compatibilidade de comunicação entre os sistemas, além de se tratar de um protocolo altamente confiável e com ampla utilização na indústria. Esse *middleware* será capaz de gerenciar os dados e informações que serão trocados pelo sistema supervisório e o software de tratamento de esgotos adquirido pela CAJ.

Em um contexto social, onde a empresa Pesco Automação & Controle se faz presente, este trabalho contribui para um plano de aumento do volume de esgoto tratado no município de Joinville, fato este que altera diretamente em melhoria de saúde para a população joinvilense.

1.4 Condições Práticas e de Factibilidade da Pesquisa

Este trabalho está inserido em um projeto da empresa Pesco Automação & Controle de desenvolvimento da automação e controle da estação de tratamento de esgotos de Jarivatuba, na cidade de Joinville. Esta obra é um investimento de aproximadamente 52 milhões de reais, captado através de um crédito a fundo perdido do Orçamento Geral da União (OGU), para a construção de

uma nova estação de tratamento de esgoto de Jarivatuba que amplia o volume de esgoto tratado de 490 l/s para 600 l/s.

A empresa Pesco Automação & Controle atua no mercado de automação de processos industriais para sistema de tratamento de água e esgoto há 7 anos, desenvolvendo e colaborando com a melhoria nos processos automatizados de tratamento de água e efluentes. Seu Know-how foi construído com o desenvolvimento de projetos para a Companhia Catarinense de Água e Saneamento (CASAN) e a Companhia Rio Grandense de Saneamento (CORSAN), na qual já executou diversas automações de plantas no sul do país.

A empresa Pesco também é incentivadora de trabalhos e publicações acadêmicas e por meio de um programa de incentivo a qualificação profissional, realiza a liberação da carga horária de trabalho para os períodos de aula e de desenvolvimento da dissertação de mestrado.

Para o desenvolvimento dessa proposta de mestrado foi necessário um microcomputador com licença de desenvolvedor, do software E3, do fabricante Elipse, com o driver de comunicação OPC. Para a configuração e programação das CPUs será utilizado o CODESYS V3.5.12; a IHM será programada com o GALILEO 10; e os *gateways* e as remotas serão configurados com o software *IO Assistant*; essas plataformas e hardwares serão garantidas e custeadas pela empresa Pesco. Os hardwares adquiridos são da fabricante de componentes para automação EATON.

1.5 Estrutura da Dissertação

- Introdução - Este capítulo contextualiza o problema e apresenta os conceitos tecnológicos que serão abordados no desenvolvimento.
- Definição do Problema - Este capítulo irá definir o problema a ser resolvido e uma proposta de solução para o problema.
- Objetivo Geral e Específicos - Apresentará o objetivo geral e os objetivos específicos desta pesquisa.
- Justificativa e Relevância - Neste capítulo será apresentada uma justificativa para a realização do projeto e as necessidades da sua aplicação.
- Condições Práticas e de Factibilidade - Será apresentado os equipamentos necessários para a solução do problema proposto.
- Revisão Bibliográfica - Foi realizada uma pesquisa para a verificação das publicações nas áreas de *middlewares* e supervisórios com comunicação OPC *Classic* e OPC UA, além de

softwares e plataformas para a implementação da solução. Também serão estudados trabalhos relacionados a redes industriais.

- Desenvolvimento e implementação - Este capítulo irá apresentar a plataforma de desenvolvimento e irá descrever como foi realizado.
- Testes e validação - Neste capítulo serão apresentados os resultados dos testes e as validações realizadas no *middleware* desenvolvido.
- Conclusões - O capítulo final deverá apresentar as considerações finais sobre o projeto e propostas de pesquisas futuras.

2. Revisão Bibliográfica

2.1 Sistemas de supervisão SCADA

Os sistemas de supervisão, ou também conhecidos como sistemas SCADA, são softwares que realizam o monitoramento e o controle dos processos industriais. Também são capazes de armazenar em seu banco de dados as informações da aplicação e permitem uma interação de visualização e controle com o operador através de telas ilustrativas da operação (Branquinho, Moraes, Seidl, Junior, & Branquinho, 2014).

A implementação de sistemas de supervisão traz como principais benefícios aos clientes a qualidade no processo produtivo mediante o monitoramento e o controle na aplicação, além da redução de custos operacionais centralizando o comando e permitindo o controle remotamente. O processo gerencial das variáveis da aplicação se torna fácil, mediante a verificação dos relatórios e dos históricos, que podem ser gerados através do banco de dados.

De um modo geral, os sistemas SCADA são subdivididos em dois módulos. O primeiro módulo é responsável pelo desenvolvimento das telas gráficas representativas e também pelas configurações dos [1]*drivers* e comunicação do sistema. O segundo módulo, denominado de módulo de execução, é responsável pela execução da aplicação. Esta que deverá ser executada em um servidor responsável por gerenciar as comunicações, os acessos e o controle, além do armazenamento dos dados que permite a geração de relatórios, gráficos e históricos das variáveis e o tratamento de alarmes identificados durante a operação (Branquinho, Moraes, Seidl, Junior, & Branquinho, 2014).

Atualmente os principais softwares para desenvolvimento de plataformas SCADA vendidos no Brasil são:

ZenOn – Fabricado pela COPA-DATA e distribuído no Brasil pela empresa ABB, o sistema de controle e supervisão garante visualização, comunicação, e controle ideais, mesmo em redes complexas. A recém lançada versão 8.00 do software, traz destaque para novidades como a análise baseada em dados dos turnos de produção, expansão no controle de bateladas, otimizações no módulo de tendência estendida, melhor usabilidade ao criar telas e símbolos, controle de processos estatísticos, análise preditiva, entre outras funções de filtros e cálculos.

SIMATIC WinCC – Desenvolvido pela Siemens, esse software assim como as outras ferramentas SCADA também inclui em suas funcionalidades os controles padrões de análise de processo, geração de relatórios, gerenciamento de usuário, comunicações com inúmeros

[1] Os Drivers são utilizados para permitirem a aquisição de dados através da comunicação entre os hardwares e softwares proprietários.

equipamentos e telas de visualizações. O grande diferencial fica pela integração entre os softwares das soluções em automação desenvolvida pela Siemens.

ScadaBR – É um software livre, gratuito e de código-fonte aberto, para desenvolvimento de aplicações de automação, aquisição de dados e controle supervisório.

Elipse E3 – Fabricado e distribuído pela Elipse Software é uma consagrada ferramenta SCADA para monitoramento e controle de processos. Líder no mercado brasileiro, é a ferramenta mais difundida, desde simples interfaces até complexos centros de controle em tempo real.

2.2 Redes e Protocolos Industriais

O início da instrumentação de processo deu-se nos anos 40, onde o monitoramento de sinais era realizado através de sinais de pressão 3-15 psi. Após vinte anos, nos anos 60, os sinais analógicos e elétricos de 4-20 mA, foram utilizados para monitorar o controle no chão-de-fábrica. Com a evolução dos processadores e a chegada dos computadores nos anos 70, um novo conceito de controle se iniciou, porém esses computadores ainda eram grandes e caros, além de não resistir ao ambiente agressivo da indústria. Somente nos anos 80, com a necessidade de integrar os sensores de campo aos computadores controladores de processo que evidenciou a necessidade da criação de redes industriais (Lugli & Santos, 2009).

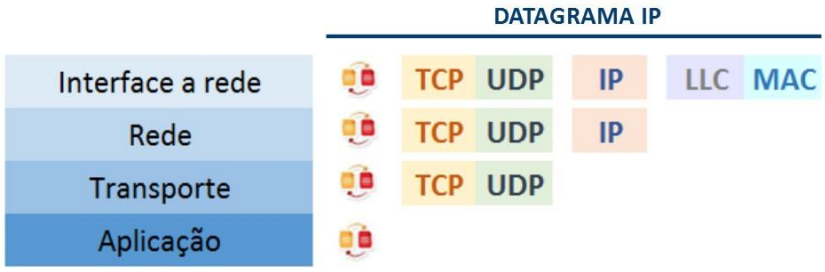
As redes industriais, por definição, são estruturas de comunicação que permitem a troca de informações entre diferentes equipamentos do processo de controle de uma indústria (Lugli & Santos, 2009). Essas estruturas podem ser divididas em meios físicos de transmissão e protocolos de comunicação. Como exemplo de meios físicos de transmissão podem ser citados o cabo de pares trançados UTP, a fibra ótica, as ondas de rádio, as micro-ondas, o infravermelho, o *bluetooth*, entre outros. Como exemplo de protocolos de comunicação os mais difundidos nas indústrias são: Modbus, CANopen, DeviceNet, EtherCAT, Profinet, Profibus, Ethernet.

A crescente necessidade de conectividade entre o ambiente de produção e o gerenciamento empresarial ampliou a abrangência de mercado das redes industriais, e então a criação de um modelo padrão se tornou necessária.

Na década de 70, a fim de se obter uma comunicação confiável, o departamento de defesa dos Estados Unidos desenvolveu um modelo de comunicação denominado TCP/IP. Este modelo futuramente foi utilizado também pela NASA e outras empresas para interligarem suas redes, resultando no que se conhece atualmente por internet (Mackay, Wright, Reynders, & Park, 2004).

O modelo TCP/IP desenvolvido pelo departamento de defesa dos Estados Unidos é composto por quatro camadas como pode ser visto na ***Figura 2***. A camada de aplicação realiza a comunicação entre os programas e os protocolos de transporte (HTTP, FTP, SMTP). A camada de transporte é responsável por dividir e organizar o pacote de dados transmitido em pacotes menores (TCP, UDP). A camada de rede, adiciona ao pacote de dados um cabeçalho contendo o endereço de envio e de recebimento. Por fim, a camada de interface de rede faz a transmissão e o recebimento dos datagramas IP (Costa, 2009).

Figura 2 - Camadas do modelo TCP/IP



Fonte: Própria

No início dos anos 80, a ISO (*International Standards Organization*) desenvolveu o modelo OSI, baseado no modelo TCP/IP, porém adicionando três camadas ao modelo, como pode ser visto na ***Figura 3***.

Figura 3 - Camadas do modelo OSI

Fonte: Própria

Neste modelo os dados são encapsulados e após adicionar um cabeçalho (*header*) são transmitidos para as camadas próximas (Mackay, Wright, Reynders, & Park, 2004). A camada de aplicação oferece serviços de redes às aplicações baseados em protocolos. A camada de apresentação, realiza a apresentação de dados e as conversões de formatos entre máquinas. A camada de sessão estabelece comunicação entre a origem e o destino. A camada de transporte liga os processos em computadores diferentes e cria o conceito de ligação. A camada de rede fornece o endereço global na

rede e cria o conceito de pacote. A camada de enlace de dados agrupa os bits para transmissão e cria o conceito de trama. Por fim a camada de meio físico transforma bits em sinais elétricos (Costa, 2009).

Mesmo com a criação do modelo OSI pela ISA, a relevância e reputação do modelo TCP/IP resultaram na adoção praticamente universal do modelo (Costa, 2009).

2.2.1 Ethernet TCP/IP industrial

A *Ethernet* é mundialmente utilizada para a comunicação entre computadores e com a inserção desses computadores nos controles industriais houve um período de experimentação dessa tecnologia em ambientes industriais.

Inicialmente esse protocolo foi considerado não ideal para a utilização nas redes de comunicação industriais por não ser determinístico[2], fato esse que caracteriza as redes industriais. Também não apresentava uma solução efetiva que garanta a entrega dos dados em casos de falha ou colisão durante a transmissão. Com a utilização de *switches* industriais essas falhas de colisão e perdas puderam ser controladas e gerenciadas (Lugli & Santos, 2009).

Com a popularização do *Ethernet* TCP/IP alguns fabricantes desenvolveram seus próprios protocolos a nível de aplicação, onde podemos destacar o Modbus TCP. Protocolo aberto que permite encapsular as tramas Modbus em *Ethernet*. O Modbus é o protocolo mais difundido nas aplicações industriais e permite a utilização de uma arquitetura em anel, garantindo a redundância da rede.

O protocolo *Ethernet* TCP/IP industrial, comumente utiliza os seguintes meios físicos: cabos de cobre UTP (sem blindagem) ou STP (com blindagem), com conectores RJ45, ou fibra ótica em longas distâncias (monomodo até 15km e multimodo até 3km).

A rede é estruturada utilizando os seguintes componentes:

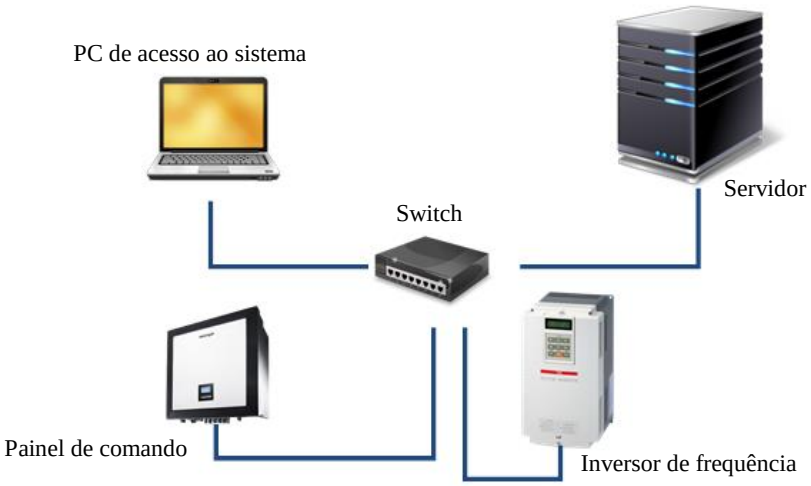
- *Hubs*: utilizados para repetir ou ampliar o sinal;
- *Switches*: responsáveis pelo envio e recebimentos dos dados;
- Roteadores: responsável pela configuração e gerenciamento da rede;
- *Gateways*: utilizado para traduzir protocolos.

As redes *Ethernet* TCP/IP podem ser configuradas em diversas topologias, onde entre elas destacam-se as três mais utilizados na indústria:

[2] As redes determinísticas permitem determinar com precisão o tempo necessário para a transferência de informações entre os integrantes da rede.

Estrela: os equipamentos estão interligados através de um equipamento intermediador central (*Switch*), por onde toda informação é obrigatoriamente passada. Esse equipamento é responsável pela entrega dos dados ao destinatário. A ***Figura 4*** demonstra tal topologia.

Figura 4 - Topologia estrela



Fonte: Própria

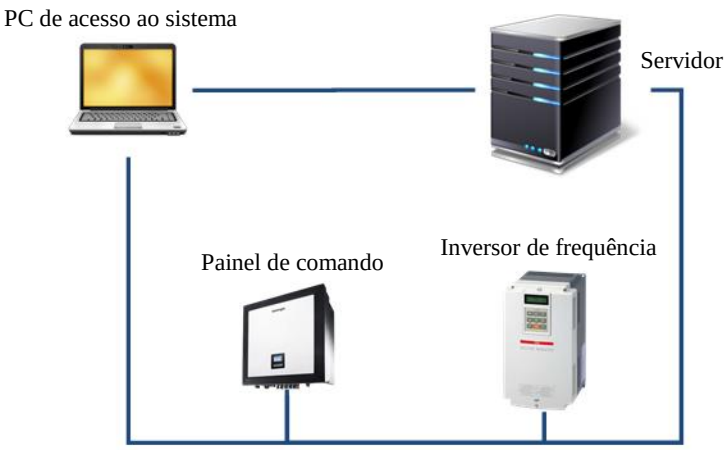
Barramento: é criado um único "barramento", onde as conexões dos equipamentos são feitas. Nessa topologia, apesar dos dados não trafegarem por dentro dos nós, apenas um equipamento tem permissão para escrever no barramento por vez. A ***Figura 5*** demonstra tal topologia.

Figura 5 - Topologia linear

Fonte: Própria

Anel: os componentes estão conectados em série, formando um caminho fechado. Todos os componentes necessitam de duas portas, uma entrada e uma saída, onde os dados são transmitidos unidirecionalmente de nó em nó até chegarem em seu destino. A ***Figura 6*** demonstra tal topologia.

Figura 6 - Topologia em Anel



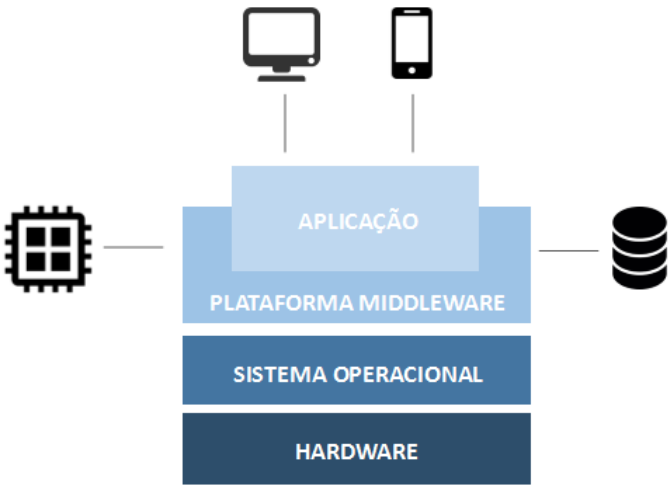
Fonte: Própria

Também entre as topologias mais encontradas na indústria e nas redes de computadores do mundo, está a topologia híbrida que une diversas topologias para se adequar à necessidade da aplicação. A combinação de topologias de redes agrega os benefícios e características de cada rede.

2.3 Middlewares

Os *middlewares* são softwares que realizam a mediação e o interfaceamento entre sistemas de dados e aplicações finais com diferentes protocolos de comunicação ou plataformas com bases diferentes entre si. São responsáveis pela troca de informação e dados entre os softwares, além de gerenciar também a troca de dados, possibilitando a implementação de módulos de segurança para as aplicações. A ***Figura 7*** representa essa interoperabilidade caracterizada pelo *middleware*, que provê uma camada entre a aplicação e o sistema operacional, escondendo aspectos de distribuição e deixando a aplicação sem a responsabilidade de gerenciar tais funções. (Araújo, 2016)

Figura 7 - Representação da posição relativa de um middleware



Fonte: Adaptada de (Filho, 2009)

A primeira utilização do termo "*middleware*" foi no final da década de 80, onde utilizou-se para descrever o gerenciamento de conexões em software, mas popularizou-se a partir da metade da década de 90, quando a tecnologia de redes passou a ser utilizadas em aplicações computacionais e industriais. (Bakken, 2003)

Novos paradigmas e serviços foram inseridos para tornar mais fácil e controlável a construção e desenvolvimento de aplicações. Serviços que provêm de mecanismos de tratamento das conexões entre objetos, em um programa local ou entre objetos que estejam distribuídos em rede. Serviços de gestão de recursos de hospedagem lógica do programa e interfaces para diversas formas de comunicação também foram inseridos. (Filho, 2009)

Em algumas soluções de automação, os sistemas centralizados implementados com Controladores Lógico Programáveis (CLPs) passaram a ser substituídos por sistemas distribuídos, com entradas e saídas distribuídas em equipamentos denominados "estações remotas". Isso fez com que os protocolos de comunicação para tais aplicações fossem aprimorados, exigindo uma alta segurança e desempenho na entrega das informações trocadas pelos equipamentos utilizados em ambientes industriais. (Filho, 2009) Essa distribuição da implementação em estações remotas pode ser observada em diversos artigos que procuram modelar soluções com abstração dos sistemas operacionais e da comunicação entre objetos distribuídos. (Bakken, 2003)

Os middlewares, segundo Bakken, 2003 podem ser divididos em quatro categorias, de acordo com a função de abrangência das atividades executadas:

- *Distributed Tuples* – Permite a distribuição de bancos de dados em um ambiente de rede. É uma das formas mais utilizadas de middlewares atualmente.

- *Remote Procedure Call (RPC)* – Permite que a lógica de uma aplicação seja distribuída ao longo de uma rede, permitindo que programas em sistemas remotos sejam chamados como se fossem sistemas locais.
- *Message-Oriented Middleware (MOM)* – Permite troca de informações entre programas através de mensagens, prevê a transparência de filas de mensagens que podem ser acessadas através de uma rede.
- *Object Request Brokers (ORB)* – Permite que objetos que compõem uma aplicação sejam distribuídos ao longo de uma rede.

Dos middlewares utilizados industrialmente, nos deparamos com uma grande tendência de utilização da plataforma de interoperabilidade *OLE for Process Control* (OPC), que será detalhada na seção a seguir.

2.3.1 OPC

A especificação OPC define conjuntos de objetos, interfaces e métodos para serem utilizados no controle e monitoramento de plantas industriais, facilitando a interligação entre sistemas que utilizam diferentes protocolos. (Rocha, 2013)

Essa especificação nasceu na década de 90, onde foi criada uma organização chamada de OPC Foundation, que reuniu um grupo de empresas junto a membros da Microsoft com o objetivo de desenvolver uma interface padronizada de comunicação baseado em OLE (*Object Linking and Embedding*), COM (*Component Object Model*) e DCOM (*Distributed Component Object Model*) para a troca de dados em sistemas operacionais Windows (Fonseca, 2002).

O grupo tinha como principal objetivo atender às necessidades da indústria com a criação de especificações para definir interfaces de softwares para padronizar a troca de dados entre o nível "máquina" e o nível de gerenciamento empresarial (Gonçalves, 2012).

Segundo Rocha, 2013, o padrão OPC *Classic* apresenta as seguintes especificações:

OPC Overwiew – Definição geral dos campos de aplicação das especificações OPC.
OPC and XML – Integração entre OPC e XML para aplicações via Internet (web)

OPC Security – Mecanismos de controle de acesso das informações do processo a clientes.
OPC Common Definitions and Interfaces – Definição das funcionalidades básicas para as demais especificações.

OPC Historical Data Access Specification (HDA) – Definição da interface para acesso a dados históricos.

OPC Data Access Specification (DA) – Definição da interface para leitura e escrita de dados de tempo real.

OPC Alarm & Events Specification (AE) – Definição da interface para monitoração de eventos.

OPC Commands – Desenvolvimento de interfaces que permitem a identificação, envio e monitoração de comandos entre clientes e servidores.

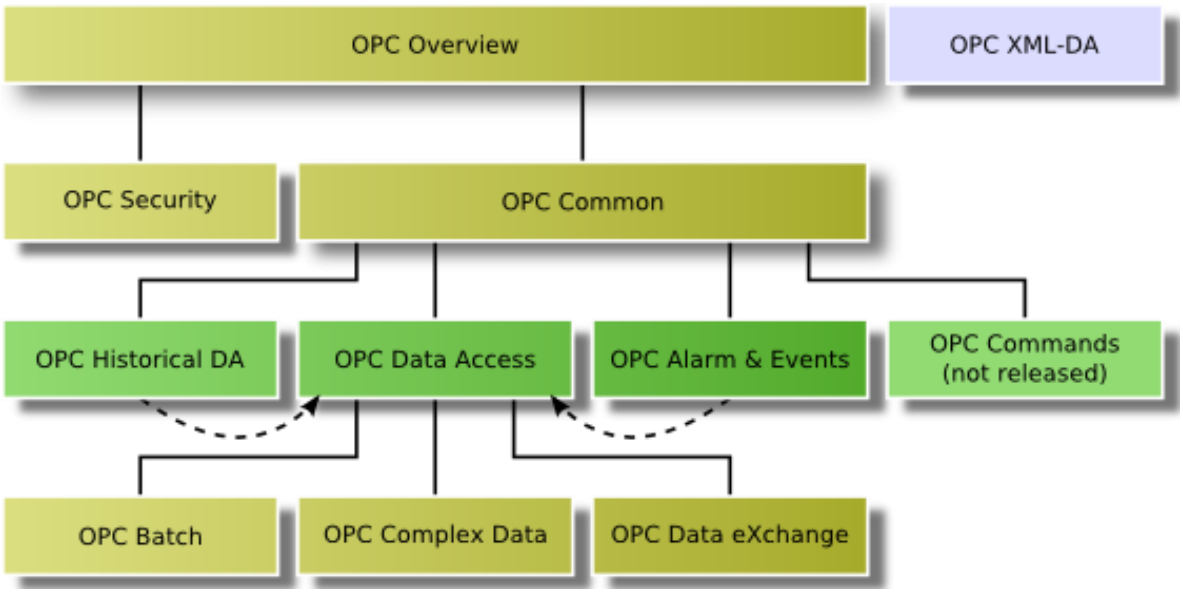
OPC Batch Specification – Definição da interface para utilização de políticas de segurança.

OPC Complex Data – Permite aos servidores a descrição e representação de formatos de dados mais complexos, tais como estruturas binárias e documentos XML.

OPC Data eXchange – Fornece mecanismos para servidores OPC DA trocarem dados através de redes *fieldbus*.

A ***Figura 8*** a seguir demonstra o fluxograma das especificações do OPC *Classic*.

Figura 8 - Especificações OPC *Classic*



Fonte: (Mahnke, Leitner, & Damm, 2009)

O padrão OPC *Classic* se mostra à frente no mercado de automação industrial, com a padronização das interfaces de comunicação entre cliente e servidor, a interoperabilidade entre sistemas de diversos fabricantes, a integração com sistemas MES e ERP, além de aplicações Windows como Excel, e a eliminação de drivers específicos ou proprietários. Vantagens essas garantidas por um padrão de comunicação aberto e com atualizações constantes (Fonseca, 2002).

2.3.1.1 OPC DA

Na especificação OPC *Data Access* a comunicação é realizada entre servidor e cliente, onde o servidor é o mediador entre os dispositivos de campo e a rede de supervisão e encapsula as informações, disponibilizando em sua interface. Por sua vez o cliente conecta ao servidor e pode acessar os dados. (Viegas, 2012)

Esse modelo define as regras e funções da transmissão de dados entre o servidor e os clientes. O modelo suporta a ligação de diversos clientes a diversos servidores. A comunicação entre clientes não é permitida, nem mesmo entre servidores. Esse modelo foi altamente aceito pelo mercado industrial, tendo surgido rapidamente empresas fornecedoras de servidores, clientes e drivers para comunicação de diversos hardwares. (Viegas, 2012)

Segundo (Rocha, 2013) são disponibilizados quatro mecanismos de comunicação entre clientes e servidores DA:

Chamadas síncronas: O cliente solicita uma operação de leitura ou escrita de dados e aguarda a resposta do servidor.

Chamadas assíncronas: O cliente solicita uma operação de leitura ou escrita de dados, mas não fica aguardando o servidor DA. Posteriormente, o cliente recebe uma notificação sobre o resultado da operação, chamada de função *handler*.

Subscription: O cliente seleciona uma taxa de atualização no qual o servidor irá realizar leituras cíclicas dos itens num determinado grupo e irá enviar as informações para o cliente, caso uma mudança de valor ocorrer.

Refresh: O cliente realiza a leitura de todos os itens de um determinado grupo, independentemente desses itens terem sofrido alteração em seu valor.

Segundo (Viegas, 2012) a comunicação estabelecida entre os clientes e o servidor utiliza os seguintes objetos dos componentes COM/DCOM:

Server – Objeto que representa o servidor e permite a criação e manipulação de grupos.

Group – Objeto que realiza o agrupamento e a gestão do estado dos itens.

Item – Objeto que concede visibilidade de rede a uma variável do servidor.

Client – Interface que permite a comunicação e troca de informação entre cliente e servidor.

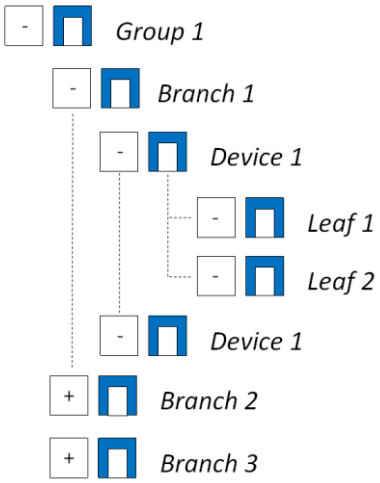
O cliente tem a possibilidade de organizar as variáveis de campo, criar grupos e condicionar o comportamento dos grupos permitindo estarem ativos ou inativos, ligados ou desligados, em *cache*,

também podem configurar o valor de banda morta e a taxa de atualização em que quer ser notificado sobre alterações de valores. (Viegas, 2012)

Os servidores OPC DA possuem as funções de efetuar a aquisição dos dados e disponibilizar para os clientes conectados a ele. Nesta especificação é utilizado o conceito de *namespace* para representar o conjunto de dados disponibilizados pelos servidores. (Rocha, 2013)

Para setorizar os dados que serão trocados conforme a necessidade de cada cliente, é utilizado a propriedade de grupos. Esta propriedade agrupa itens (*tags*) que descrevem as entradas, saídas e variáveis que serão utilizadas. Os itens podem se expandir em *branches* e estes podem se expandir em *leafs* como podem ser visto na ***Figura 9.***

Figura 9 - Estrutura *namespace* do servidor OPC



Fonte: Própria

Cada item é associado aos seguintes atributos:

Item – Nome;

Value – Valor ou estado do item;

Quality – Poderá ter a classificação de boa (*good*), má (*bad*) ou incerta (*uncertain*).

Time Stamp – Data de quando o valor foi obtido.

A ***Figura 10*** demonstra os atributos associados ao item *Branch 1*.

Figura 10 - Atributos do item

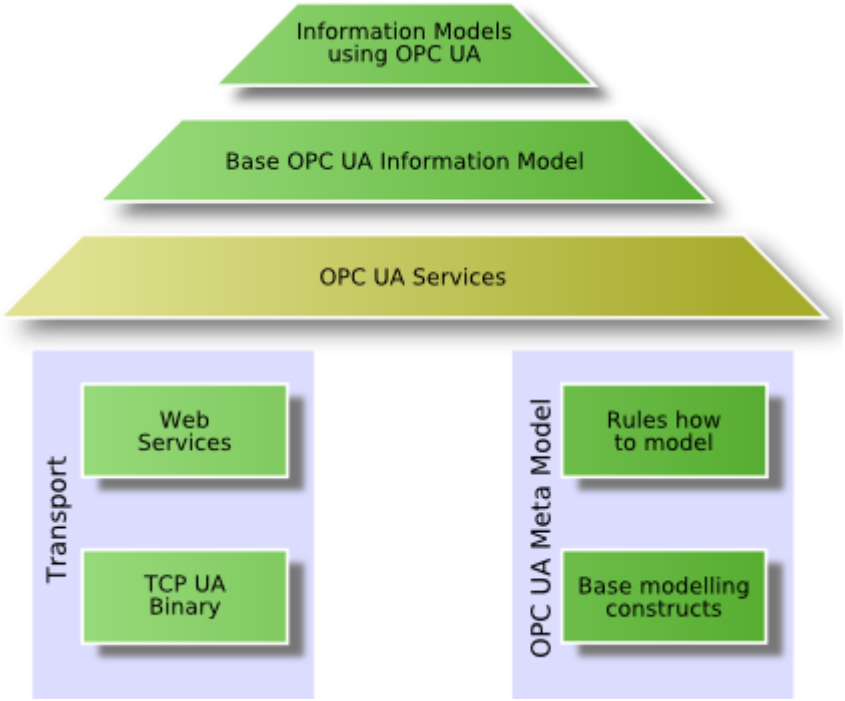
Item	Value	Quality	TimeStamp
Branch 1	0	Good	14/01/19 10:01:05

Fonte: Própria

2.3.1.2 OPC UA

O OPC UA (*Unified Architecture*) foi criado para substituir as especificações baseadas em COM, sem que houvesse perdas de desempenho. Assim sua base consiste nos mecanismos de transporte e modelamento de dados como pode ser visto na ***Figura 11***.

Figura 11 - Estrutura da base do OPC UA



Fonte: (Mahnke, Leitner, & Damm, 2009)

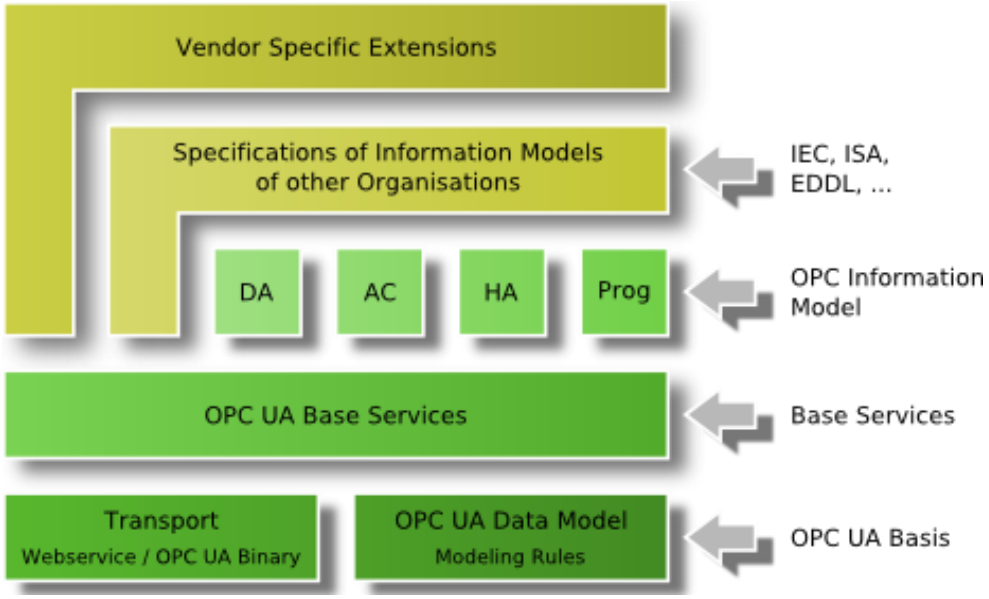
O transporte define diferentes mecanismos otimizados para diferentes casos. A primeira versão do OPC UA define a utilização do protocolo TPC para a comunicação intranet, assim como também para internet e serviços web (Mahnke, Leitner, & Damm, 2009).

A modelagem de dados define regras e a base de blocos de construção necessários para expor um modelo de informação com o padrão OPC UA. Também é responsável por determinar pontos de entradas nos endereços e as bases usadas para criar uma hierarquia (Mahnke, Leitner, & Damm, 2009).

O serviço UA desenvolve o papel de realizar a interface entre os servidores e os clientes, onde os servidores são enxergados como fornecedores de informação e os clientes como consumidores destas informações (Mahnke, Leitner, & Damm, 2009).

Para atender todas as características apresentadas pelo OPC *Classic*, foram definidos diversos modelos de informação nas especificações de base. A ***Figura 12*** mostra as diferentes camadas dos modelos de informação definidos pelo OPC.

Figura 12 - Camadas da Arquitetura OPC UA [9]



Fonte: (Mahnke, Leitner, & Damm, 2009)

DA (*Data Access*) - define as extensões especificas de automação, tais como a modelagem de dados analógicos ou discretos e a qualidade do serviço.

AC (*Alarm & Conditions*) - especifica um modelo avançado para o gerenciamento de alarmes de processo e monitoramento de condições.

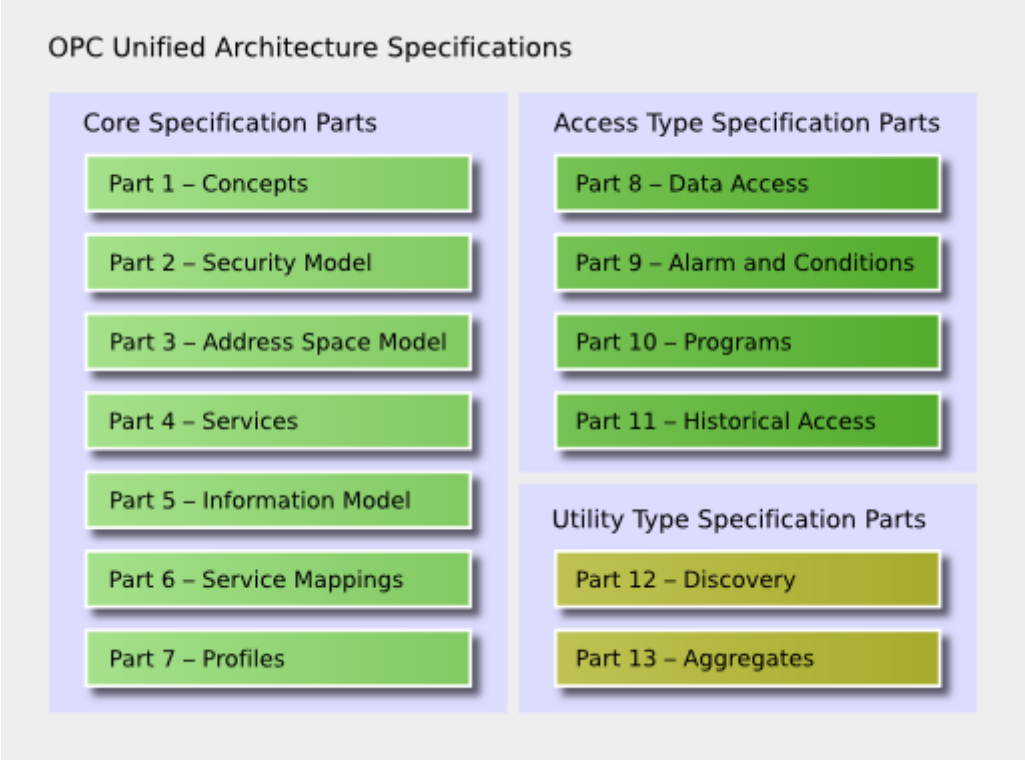
HA (*Historical Access*) - define os mecanismos para acessar dados eventos históricos.

Prog (*Programs*) – especifica um mecanismo para iniciar, manipular e monitorar a execução de programas (Mahnke, Leitner, & Damm, 2009).

2.3.1.2.1 Especificações OPC UA

A ***Figura 13*** apresenta uma visão geral das especificações da arquitetura, divididas em especificações principais , que definem a base para o OPC UA, e as partes específicas do tipo de acesso, onde se especifica principalmente os modelos de informações do padrão OPC UA.

Figura 13 - Especificações OPC UA



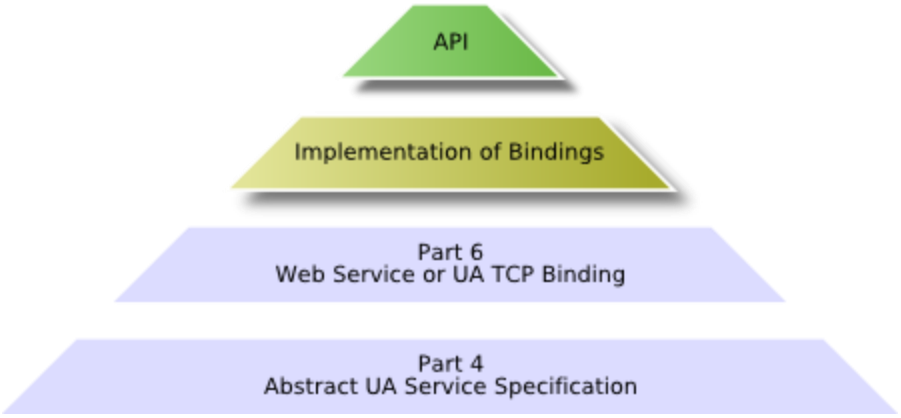
Fonte: (Mahnke, Leitner, & Damm, 2009)

As duas primeiras partes (C*oncepts* e *Security Model*) não são normativas e definem apenas os conceitos e os modelos de segurança da arquitetura.

As partes 3 (*Address Space Model*) e 4 (*Services*) são os principais documentos para o projeto e desenvolvimento de aplicativos OPC UA. A parte 3 especifica as construções dos blocos com as instâncias e o tipo de informação para construir o "*OPC UA Server Address Space*". A parte 4 representa as possíveis interações entre os clientes e as aplicações dos servidores (Mahnke, Leitner, & Damm, 2009).

Na ***Figura 14*** podem ser vistas as camadas de comunicação da arquitetura do OPC UA.

Figura 14 - Camadas de Comunicação da Arquitetura OPC UA



Fonte: (Mahnke, Leitner, & Damm, 2009)

A parte 6 define o mapeamento do serviço de mensagens, os mecanismos de segurança aplicados nas mensagens, e o confiável meio de transporte da mensagem são definidas. Essa parte é indicada apenas aos implementadores dos *UA Stacks* (Mahnke, Leitner, & Damm, 2009);

A parte 7 define os *Profiles* como subconjuntos úteis de recursos do OPC UA. A especificação define os subconjuntos em dois níveis, onde no primeiro nível são unidades de conformidade definindo um pequeno conjunto de funcionalidades que são sempre utilizadas juntas e podem ser testadas com o CTT e verificadas como unidade. O segundo nível são perfis compostos por uma lista de unidades de conformidade. Um perfil deve ser implantado completamente e será verificado como um conjunto completo durante a certificação dos produtos OPC UA. A lista de perfis suportados e usados é trocada durante o estabelecimento da conexão entre cliente e servidor, permitindo a aplicação determinar se as características necessárias são suportadas pelo parceiro de comunicação (Mahnke, Leitner, & Damm, 2009).

A parte 8, *Data Access*, especifica como representar e usar dados de automação e características específicas como unidades de engenharia (Mahnke, Leitner, & Damm, 2009).

A parte 9, *Alarm and Conditions*, especifica os alarmes de processo e as condições de monitoramento de estados das máquinas e eventos específicos (Mahnke, Leitner, & Damm, 2009).

A parte 10, *Programs*, define uma máquina de estados padrão para a execução, manipulação e monitoramento de programas (Mahnke, Leitner, & Damm, 2009).

A parte 11, *Historical Access*, especifica o uso dos serviços de acesso ao histórico e como apresentar as informações sobre a configuração do histórico de dados e eventos (Mahnke, Leitner, & Damm, 2009).

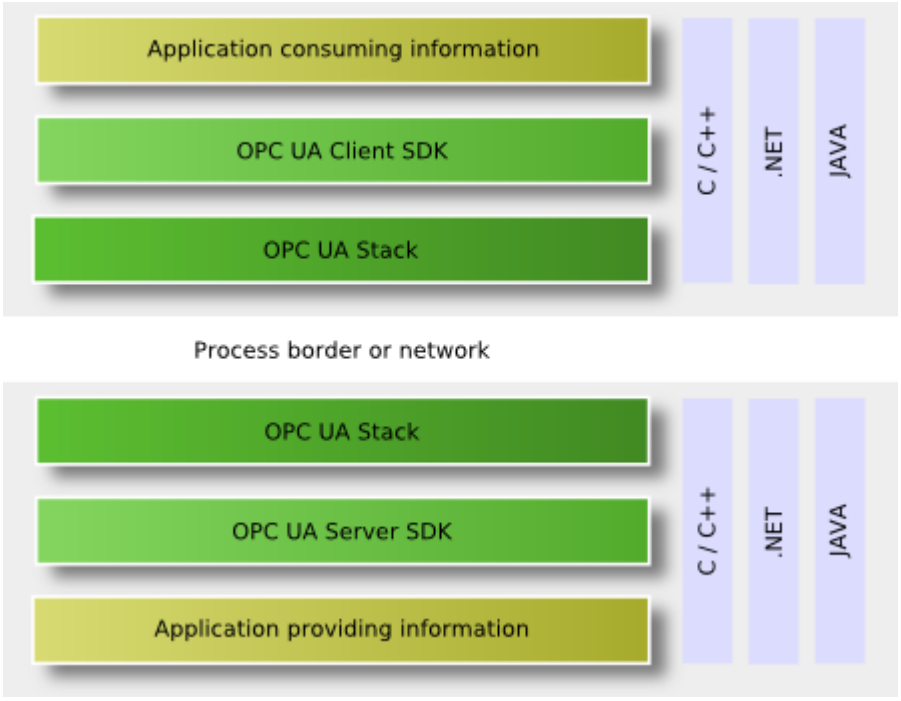
As partes 12 e 13, *Utility Type Specification Parts*, definem como encontrar servidores na rede e como o cliente pode obter as informações necessárias para habilitar e estabelecer a comunicação com um determinado servidor (Mahnke, Leitner, & Damm, 2009).

2.3.1.2.2 Camadas de Software OPC UA

Assim como o OPC clássico, o OPC UA também utiliza o conceito cliente-servidor, onde o aplicativo que deseja expor suas próprias informações é chamado de servidor e o aplicativo que deseja consumir as informações são chamados de cliente (Mahnke, Leitner, & Damm, 2009).

Uma aplicação OPC UA é composta de três camadas de software como pode ser visto na ***Figura 15***. Esses softwares podem ser implementados em C/C++, NET, ou JAVA, porém não se limitam apenas a essas linguagens de programação e plataformas de desenvolvimento. O sistema contém a funcionalidade específica para a aplicação e o mapeamento para o OPC UA, utilizando um *OPC Stack* e um *OPC UA Software Developing Kit* (SDK) (Mahnke, Leitner, & Damm, 2009).

Figura 15 - Camadas de Software OPC UA

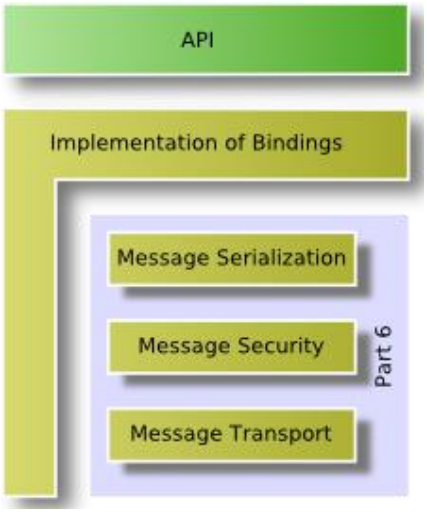


Fonte: (Mahnke, Leitner, & Damm, 2009)

O OPC UA define três camadas de pilha e perfis diferentes para cada camada. A camada de codificação de mensagens define a serialização dos parâmetros do Serviço em formato binário e XML. A camada de segurança da mensagem especifica como as mensagens devem ser protegidas usando os padrões de segurança do Web Service ou uma versão binária UA dos padrões do Web

Service. A camada de transporte de mensagens define o protocolo de rede usado, que pode ser UA TCP ou HTTP e SOAP para Serviços da Web. A ***Figura 16*** ilustra as diferentes camadas da pilha de comunicação UA.

Figura 16 - Camadas de Pilhas de Comunicação UA



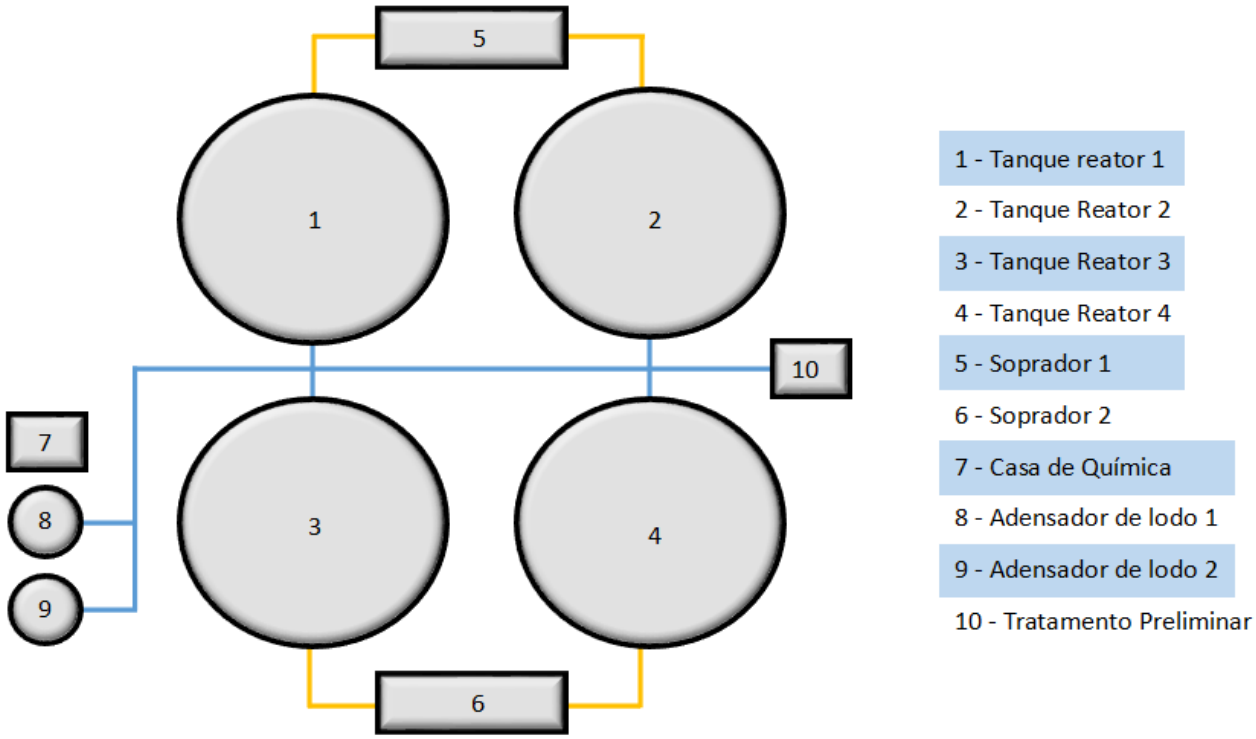
Fonte: (Mahnke, Leitner, & Damm, 2009)

3. Desenvolvimento e Implementação

3.1 A planta a ser controlada

A nova ETE Jarivatuba é uma planta com alta capacidade de tratamento de esgoto e para isso conta com quatro tanques de reatores totalizando a capacidade de tratamento de efluentes de 600 l/s. Além disso possui uma estação preliminar de tratamento, dois tanques de adensadores de lodo, duas casas de sopradores e uma casa de química. A ***Figura 17*** demostra o esquemático dos tranques distribuídos pela planta.

Figura 17 - Planta da ETE Jarivatuba



Fonte: Própria

Inicialmente o processo conta com um tratamento preliminar que retira os objetos sólidos e arenosos do efluente. Cada tanque de reator é composto por uma válvula motorizada (VM) de entrada, uma válvula motorizada de saída, uma motobomba (MB) de entrada, um sensor de nível ultrassônico, um sensor de PH e um sensor de oxigênio.

Os sopradores são responsáveis por oxigenar os tanques de reatores durante a reação biológica, e os tanques de adensadores de lodo são responsáveis por tratar o excesso de lodo descartado no processo de tratamento. Cada tanque de adensador de lodo é composto por uma motobomba e um raspador.

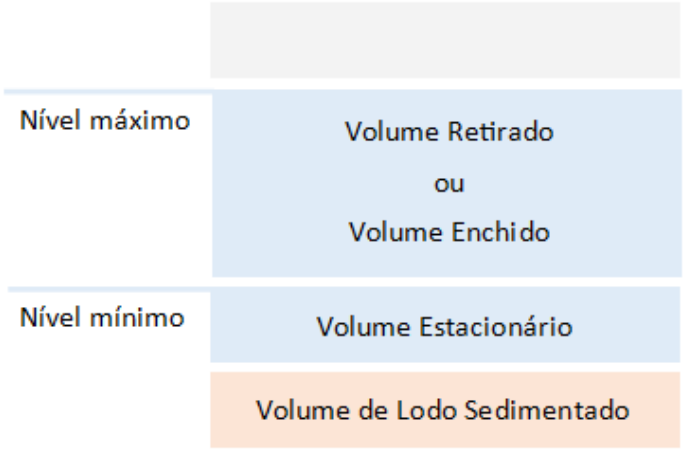
Por fim se encontra a casa de química, responsável por aplicar produtos que regulam o fator de qualidade do esgoto tratado e que será devolvido ao rio.

3.2 O processo Reator em Bateladas Sequenciais (SBR) de tratamento de esgotos

No início dos anos 70 surgiu no tratamento biológico de águas residuárias o Reator em Bateladas Sequenciais. Esse processo consiste em associar todas as unidades e processos em um único tanque, tornando o processo simplesmente sequencial ao tempo. Essa simplicidade e flexibilização do Reator em Bateladas Sequenciais tornou o processo muito popular. (Thans, 2008)

O processo incorpora um tanque com volume variável, porém pode se dividir seu volume total em duas partes independentes. A primeira denominada volume estacionário é compreendida pelo volume de lodo sedimentado diluído no volume de efluente tratado não retirado. A segunda parte compreende o volume que será retirado de efluente tratado ou novo efluente que ainda será tratado. A ***Figura 18*** representa esse volume variável do tanque reator.

Figura 18 - Representação de um reator de SBR



Fonte: Adaptada de (Thans, 2008)

Segundo Thans, 2008, usualmente um ciclo de um Reator em Bateladas Sequenciais consiste de cinco etapas:

1ª Etapa: Enchimento – o esgoto após passar pelo tratamento preliminar alimenta o tanque até o nível determinado pela operação. Nesse momento, de um ponto de vista da automação, deve-se abrir a válvula motorizada de entrada, ligar a motobomba de entrada e também ligar o soprador para iniciar o processo de aeração.

2ª Etapa: Reação biológica – deve se manter o fornecimento de oxigênio através do soprador, para que haja reações biológicas de consumo de matéria orgânica, durante o tempo determinado pelo operador.

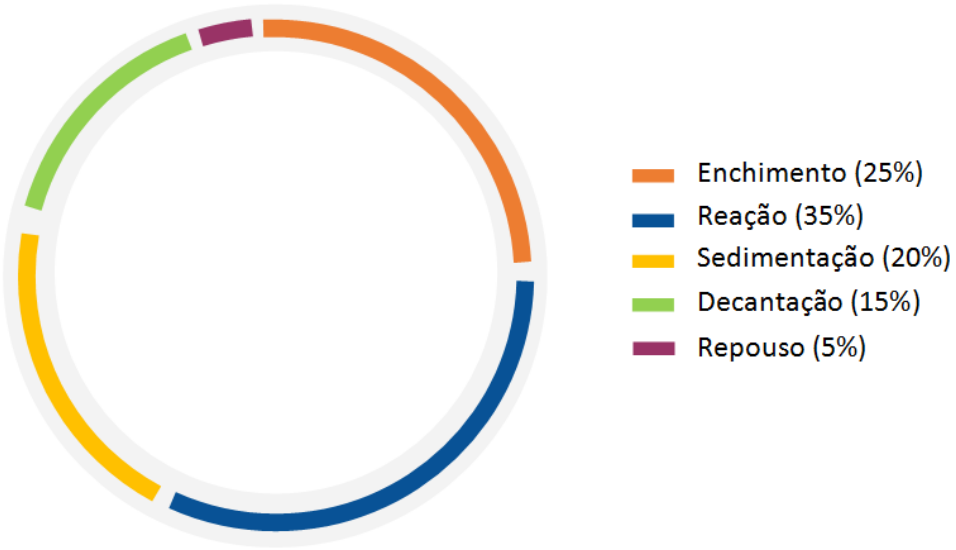
3ª Etapa: Sedimentação – interrompe o fornecimento de oxigênio, desligando os sopradores, e é aguardada a sedimentação dos sólidos em suspensão.

4ª Etapa: Retirada – o efluente clarificado começa a ser retirado através do vertedor flutuante. Nesse momento a válvula motorizada de saída deve ser aberta. Também é importante se manter uma distância de transição entre o efluente clarificado e a manta de lodo sedimentada.

5ª Etapa: Repouso e retirada de lodo em excesso – durante o repouso pode ser retirado o lodo em excesso e aguarda-se para o início de um novo ciclo.

Segundo Metcalf, 2003, a porcentagem do tempo de cada período, em relação à duração do ciclo total, pode ser dividida inicialmente em 25% do tempo gasto no enchimento, 35% do tempo gasto no processo de reação, 20% do tempo gasto na sedimentação, 15% do tempo gasto na decantação e por fim 5% do tempo gasto em um estado de repouso. Esse ciclo temporal pode ser observado na ***Figura 19***.

Figura 19 - Porcentagem do tempo do processo de tratamento por SBR



Fonte: Própria

3.3 Delimitando o objeto de estudo

Para o objeto de estudo dessa dissertação de mestrado foram escolhidas duas áreas da planta que exercem funções essenciais no tratamento de esgotos: o tanque reator e o soprador. Essas áreas

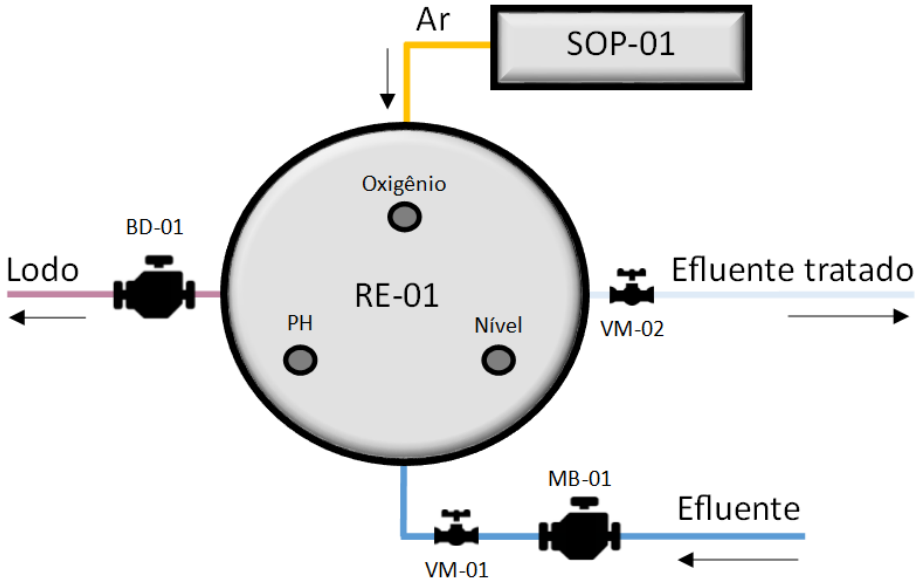
obrigatoriamente deverão compartilhar suas variáveis de leitura e comando com o software de tratamento de efluentes. Como o processo de tratamento de esgoto pode ser executado nos quatro tanques de reatores de forma exatamente igual, limitou-se ao controle do tanque reator 1 (RE-01) e também ao controle do soprador 1 (SOP-01).

O RE-01 é composto por uma válvula motorizada de entrada, uma válvula motorizada de saída, uma motobomba de entrada e uma bomba de lodo. Também no RE-01 encontram-se um sensor de nível, um sensor de PH e um sensor de oxigênio.

O conjunto SOP-01 é composto por um soprador que quando acionado realiza a oxigenação do tanque RE-01. Esse controle é realizado através da medição do sensor de oxigênio existente no tanque reator.

A delimitação da área a ser controlada é simbolizada juntamente a seus elementos de controle pela ***Figura 20*** abaixo.

Figura 20 - Seleção da área da planta a ser controlada

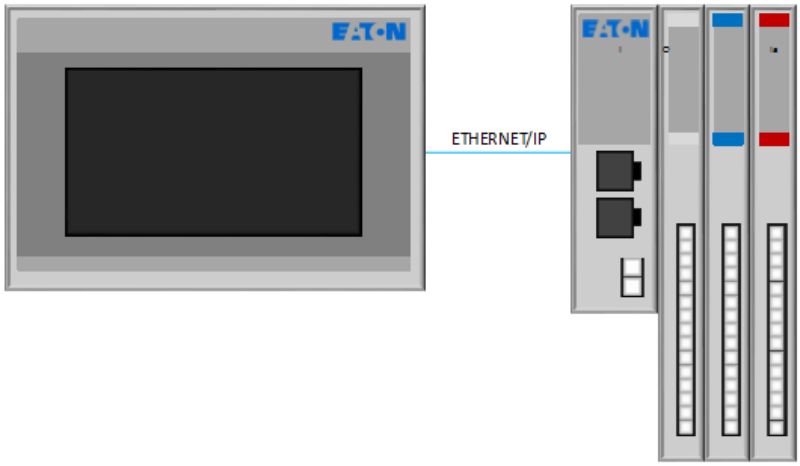


Fonte: Própria

3.4 Arquitetura dos hardwares

A partir da delimitação da área a ser controlada, observou-se a necessidade dos hardwares para a demanda de automação. Para o controle do RE-01 foi utilizado uma IHM com CLP (XV102). Também se utilizou um gateway ethernet (XNE-GWBR-2ETH-IP), um cartão de 16 entradas digitais (XNE-16DI), um cartão de 16 saídas digitais (XNE-16DO) e um cartão de oito entradas analógicas (XNE-8AI). A ***Figura 21*** demonstra a configuração de hardware utilizada para controlar o reator.

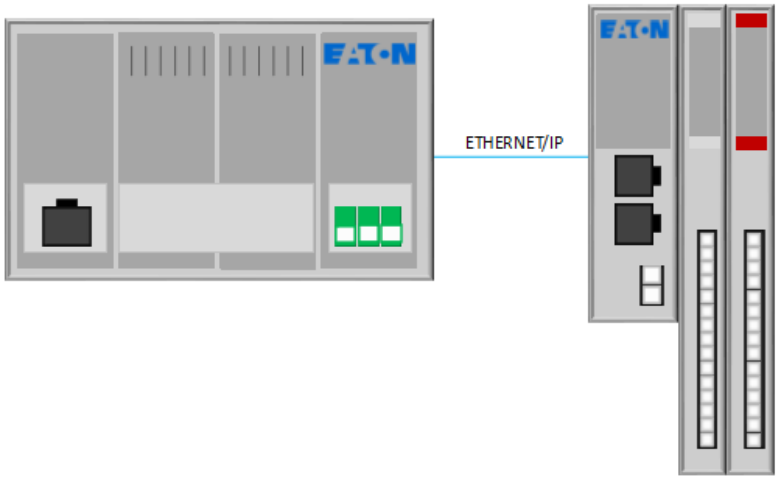
Figura 21 - Hardware para controle do RE-01



Fonte: Própria

Para o controle do SOP-01 foi utilizado um CLP (XC-152-D6-11), um gateway ethernet (XNE-GWBR-2ETH-IP), um cartão de 16 entradas digitais (XNE-16DI), um cartão de 16 saídas digitais (XNE-16DO). A ***Figura 22*** demonstra a configuração de hardware utilizada para controlar o soprador.

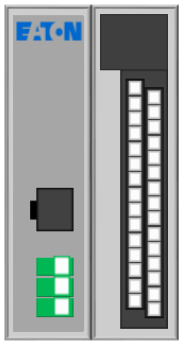
Figura 22 - Hardware para controle do SOP-01



Fonte: Própria

Foi solicitado pelo cliente que os sistemas pudessem ser controlados tanto remotamente, quanto localmente, para isso se fez necessário a utilização de CPUs em cada local de controle e também um CLP centralizador, localizado no Painel de Automação Central (PAC). O CLP central é composto por uma CPU XC-CPU202-EC4M-8DI-6DO-XV. A ***Figura 23*** representa a CPU central.

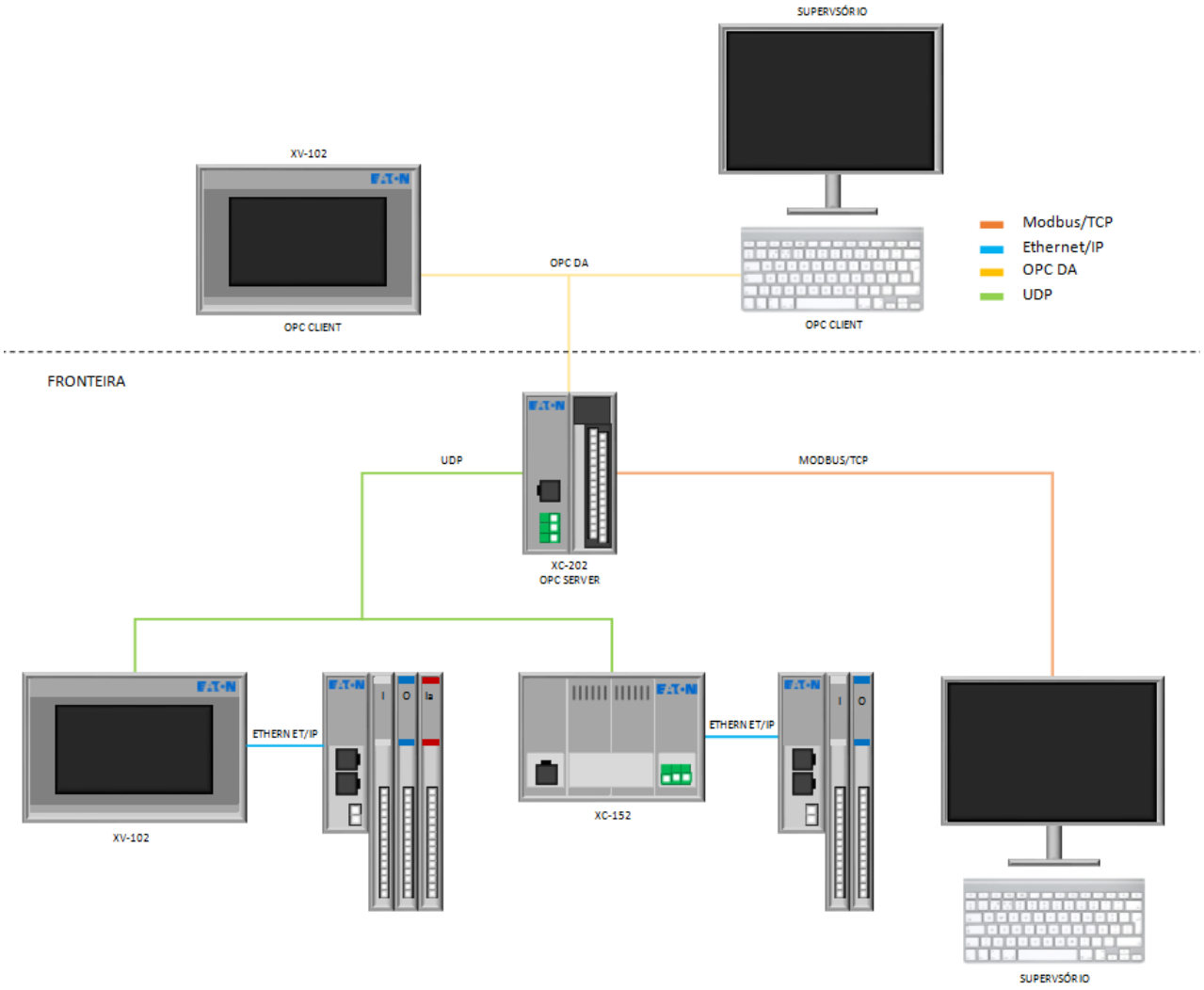
Figura 23 - CPU central



Fonte: Própria

A ***Figura 24*** representa toda a arquitetura de hardware e também de rede que foi utilizada no projeto. Além da centralização realizada na CPU XC-202, ela também é responsável por realizar a fronteira para outros dispositivos que desejam acessar dados da rede e controlá-los, através do servidor OPC implementado internamente.

Figura 24 - Arquitetura de hardware e rede da planta



Fonte: Própria

Essa é uma arquitetura de hardware muito comum na indústria, onde se pode encontrar diversos protocolos de rede. A IHM/CPU do reator se comunica com seus periféricos através do protocolo Ethernet/IP, assim como também a CPU do soprador. Entre as CPUs do reator e do soprador, e a CPU centralizadora, é utilizado o protocolo UDP. E entre o Supervisório e a CPU centralizadora utiliza-se o protocolo Modbus-TCP. A partir da CPU centralizadora da aplicação, todo o acesso é realizado através do padrão de comunicação OPC e gerenciado pelo servidor OPC do CLP XC-202.

3.5 Variáveis a serem controladas

Como já descrito anteriormente o tratamento de esgotos engloba o controle de diversos equipamentos para a fluidez do processo. Nessa seção serão detalhadas as variáveis de controle e de sinalização.

No processo de tratamento, o reator (RE-01) é responsável pelo controle de duas válvulas motorizadas e duas motobombas. Para a operação e supervisão, as válvulas possuem os seguintes comandos e sinalizações, demonstrados pela ***Tabela 1***.

Tabela 1 - Comandos e sinalizações das válvulas motorizadas

Comandos	Sinalizações
Abre	Abrindo
Fecha	Fechando
	Aberta
	Fechada

Fonte: Própria

Já os comandos e sinalizações das motobombas podem ser observados na ***Tabela 2***.

Tabela 2 - Comandos e sinalizações das motobombas

Comandos	Sinalizações
Liga	Ligado
Desliga	Desligado
Automático	Falha
Manual	
Tempo ligado	

Fonte: Própria

Além dos controles e sinalizações o reator também conta com a indicação dos valores do sensor de nível, do sensor de oxigênio, do sensor de PH e com a seleção de comando remoto e local.

O soprador responsável pela oxigenação do tanque reator, possui os seguintes comandos e sinalizações:

Tabela 3 - Comandos e sinalizações do soprador

Comandos	**Sinalizações**
Liga	Ligado
Desliga	Desligado
Automático	Falha
Manual	
Tempo ligado	

Fonte: Própria

A seguir se encontra a ***Tabela 4*** que é composta de todas as variáveis de controle e sinalização do reator e do soprador, separadas por área e por dispositivos de campo a serem controlados.

Tabela 4 - Variáveis de controle e sinalização do reator e do soprador

		Comandos	Sinalizações
Reator	VM de Entrada	Abre	Abrindo
		Fecha	Fechando
			Aberta
			Fechada
	VM de Saída	Abre	Abrindo
		Fecha	Fechando
			Aberta
			Fechada
	Bomba de Entrada	Liga	Ligado
		Desliga	Desligado
		Automático	Falha
		Manual	
		Tempo ligado	
	Bomba de Lodo	Liga	Ligado
		Desliga	Desligado
		Automático	Falha
		Manual	
		Tempo ligado	

	Comandos	Sinalizações
Soprador	Liga	Ligado
	Desliga	Desligado
	Automático	Falha
	Manual	
	Tempo ligado	

Fonte: Própria

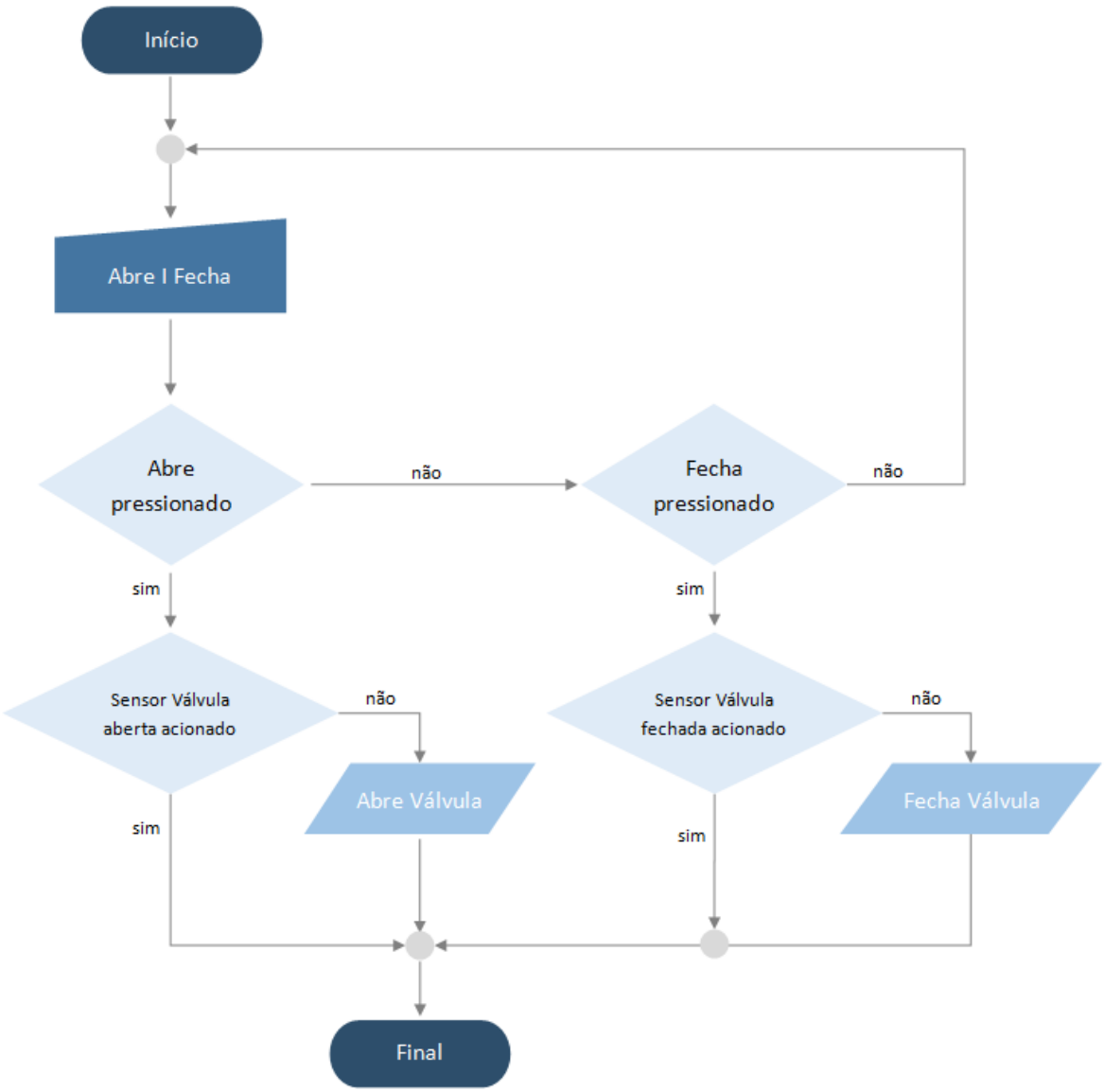
3.6 Modelamento da implementação

Inicialmente investigou-se o padrão de funcionamento das válvulas motorizadas, através de manuais e especificações técnicas. Para seu funcionamento, dois conjuntos de variáveis são necessárias. O primeiro conjunto é formado pelos botões de comando (abre e fecha) e o segundo seriam os sensores de posicionamento da válvula, que indicam que a válvula está aberta ou fechada. Dessa maneira, se pressionado o botão que comanda a abertura da válvula e o sensor de válvula aberta estiver acionado, a válvula se manterá na mesma posição, do contrário (sensor de válvula fechada acionado, ou nenhum sensor acionado) a válvula iniciará seu movimento de abertura e só interromperá quando o sensor de válvula aberta for acionado.

A operação de fechamento funciona de forma semelhante à abertura, sendo assim se pressionado o botão que comanda o fechamento da válvula e o sensor de válvula fechada estiver acionado, a válvula se manterá na mesma posição, do contrário (sensor de válvula aberta acionado, ou nenhum sensor acionado) a válvula iniciará seu movimento de fechamento e só interromperá quando o sensor de válvula fechada for acionado.

Além das sinalizações de válvula aberta e válvula fechada também é indicado ao operador se a válvula está em movimento de abertura e de fechamento. A ***Figura 25*** demonstra o fluxo de lógica para as operações de abertura e fechamento da válvula motorizada.

Figura 25 - Fluxograma de operação da válvula motorizada



Fonte: Própria

O modelamento da válvula motorizada foi desenvolvido generalizando sua utilização. Dessa maneira, seu modelo pode ser utilizado com todas as outras válvulas também da planta, mesmo que elas possuam funções extras, que podem ser incluídas unicamente em seus modelos.

A ***Figura 26*** demonstra essa generalização do modelamento da válvula motorizada, considerando inicialmente as entradas, à esquerda, e sequencialmente as funções de saídas.

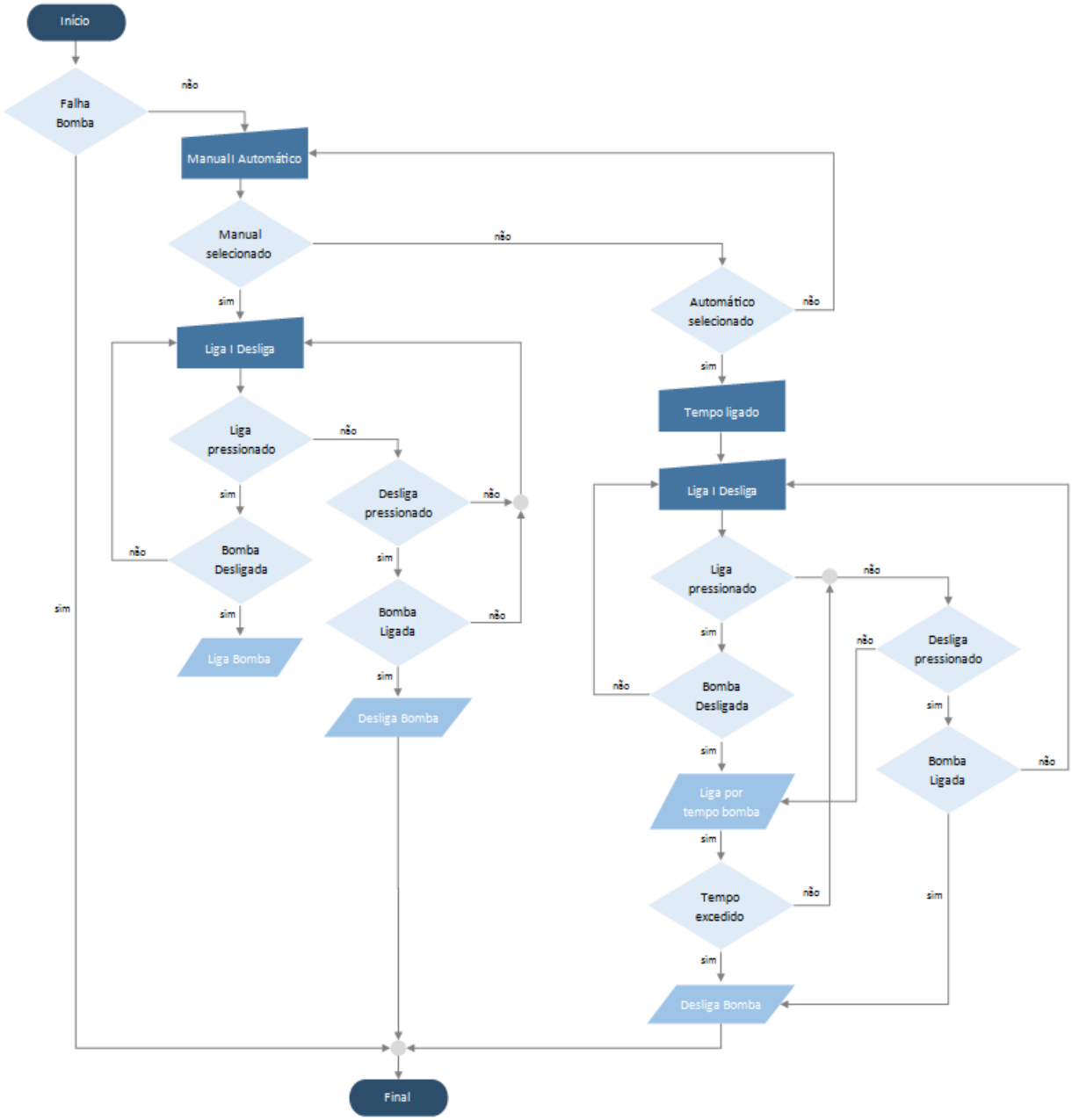
Figura 26 - Modelamento da válvula motorizada

Fonte: Própria

Da mesma maneira que foi realizado com as válvulas motorizadas, foi realizado o estudo do funcionamento das motobombas, analisando manuais e especificações técnicas. Para a operação das motobombas, quatro variáveis de comando são importantes. A primeira variável a se confirmar é se a bomba está em falha. Após a confirmação de operação normal da motobomba, é verificado se a chave seletora se encontra em manual ou em automático. Se o modo manual estiver selecionado, o controle da operação será realizado através dos comandos de botões de liga e desliga.

Já no modo em automático a operação verifica o tempo programado que a bomba irá permanecer ligada e após o comando de ligar inicia o funcionamento e a contagem de tempo. Quando exceder o tempo programado, a bomba será desligada. A ***Figura 27*** demonstra o fluxo lógico de operação e funcionamento das motobombas.

Figura 27 – Fluxograma de operação das bombas

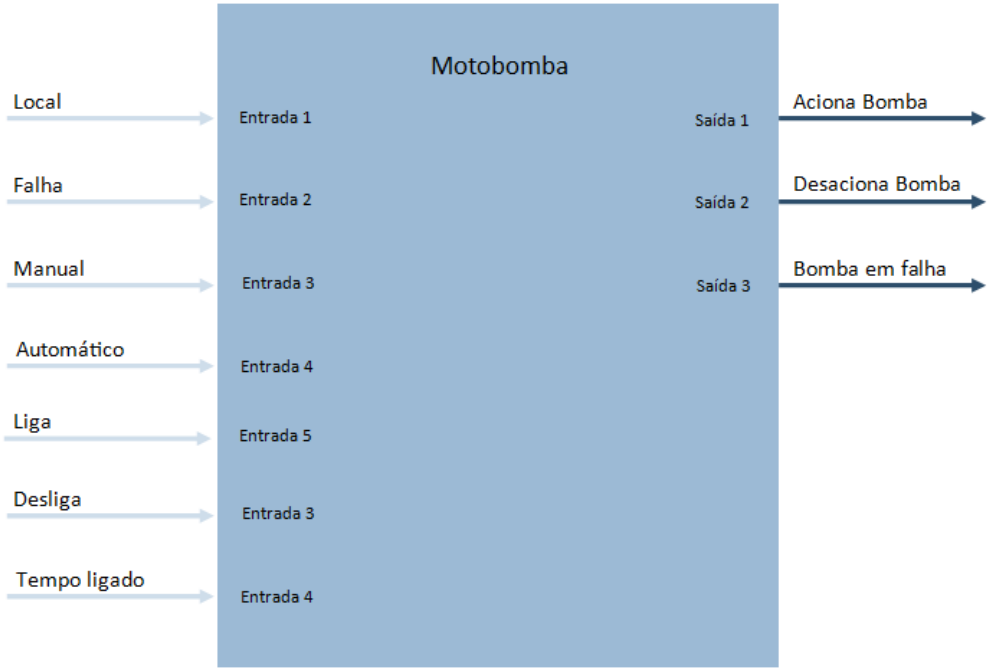


Fonte: Própria

Assim como as válvulas motorizadas, o modelamento da motobomba foi desenvolvido generalizando sua utilização. Dessa maneira seu modelo pode ser utilizado com todas as outras motobombas da planta, mesmo que elas possuam funções extras, que podem ser incluídas unicamente em seus modelos.

A ***Figura 28*** demonstra essa generalização do modelamento da motobomba, considerando inicialmente as entradas, à esquerda, e sequencialmente as funções de saídas.

Figura 28 - Modelamento da motobomba



Fonte: Própria

No ANEXO I - Fluxograma de modelagem das válvulas motorizadas e no ANEXO II - Fluxograma de modelagem das motobombas podem ser encontrados os fluxogramas de modelagem completos das válvulas motorizadas e também das motobombas. Esses fluxogramas foram utilizados para a implementação da lógica de programação dos módulos CLPs.

3.7 Configuração de entradas e saídas dos hardwares

O *hardware* utilizado para o controle do Reator é composto por uma IHM com CLP, cartão de entrada digital, cartão de saída digital e cartão de entrada analógica. Utilizou-se as seguintes entradas e saídas para os sensores das válvulas motorizadas e os sinais de falhas das motobombas como mostrado nas ***Tabela 5***, ***Tabela 6***, ***Tabela 7***.

Tabela 5 - Entradas digitais do Reator

Entradas Digitais	Descrição
I11	Bomba de Entrada em Falha
I12	Bomba de Lodo em Falha
I13	Válvula de Entrada Aberta
I14	Válvula de Entrada Fechada
I15	Válvula de Saída Aberta
I16	Válvula de Saída Fechada

Fonte: Própria

Tabela 6 - Saídas digitais do Reator

Saídas Digitais	Descrição
O1	Aciona Válvula de Entrada
O2	Desaciona Válvula de Entrada
O3	Aciona Válvula de Saída
O4	Desaciona Válvula de Saída
O5	Aciona Bomba de Lodo
O6	Desaciona Bomba de Lodo
O7	Bomba de Lodo em Falha
O8	Aciona Bomba de Entrada
O9	Desaciona Bomba de Entrada
O10	Bomba de Entrada em Falha

Fonte: Própria

Tabela 7 - Entradas analógicas do Reator

Entradas analógicas	Descrição
Ia1	Sensor de Nível
Ia2	Sensor de PH
Ia3	Sensor de Oxigênio

Fonte: Própria

O soprador, composto por uma CPU, um cartão de entrada digital e um cartão de saída digital, utilizou-se das seguintes entradas e saídas para os comandos e sinalizações como mostrado na ***Tabela 8*** e na

Tabela 9.

Tabela 8 - Entradas digitais do Soprador

Entradas Digitais	Descrição
I1	Liga Soprador
I2	Desliga Soprador
I3	Manual
I4	Automático
I5	Remoto
I6	Local

Fonte: Própria

Tabela 9 - Saídas digitais do Soprador

Saídas Digitais	Descrição
O1	Soprador Ligado
O2	Soprador Desligado

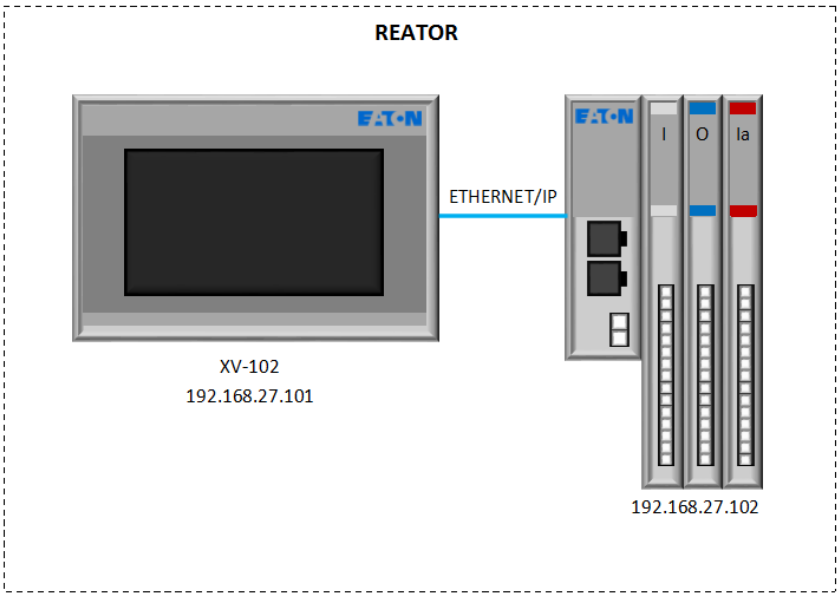
Fonte: Própria

3.8 Configuração de rede dos hardwares

Definiu-se o endereço IP de rede 192.168.27.xx para esse projeto. Iniciando o mapeamento de rede dos *hardwares* sempre pela CPU e continuando a sequência numérica com o *gateway* responsável pela comunicação das remotas. A cada mudança de setor é acrescido o valor de dez, permitindo que no futuro se amplie essa rede, incluindo novos *hardwares*.

Através dessa padronização iniciou-se o mapeamento pelo IP: 100, e o primeiro ambiente a ser mapeado na rede foi o Reator, adquirindo o IP: 192.168.27.101 (CPU) e IP: 192.168.27.102 (Remotas), como pode ser visto na ***Figura 29***.

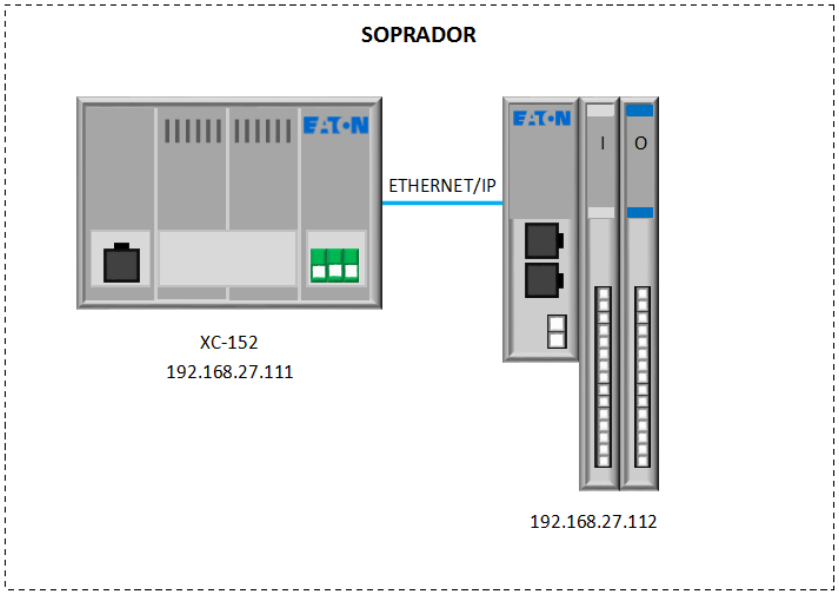
Figura 29 - Mapeamento de rede do Reator



Fonte: Própria

Sequencialmente mapeou-se o soprador, inserindo o IP: 192.168.27.111 (CPU) e o IP: 192.168.27.112 (Remotas), como pode ser visto na ***Figura 30***.

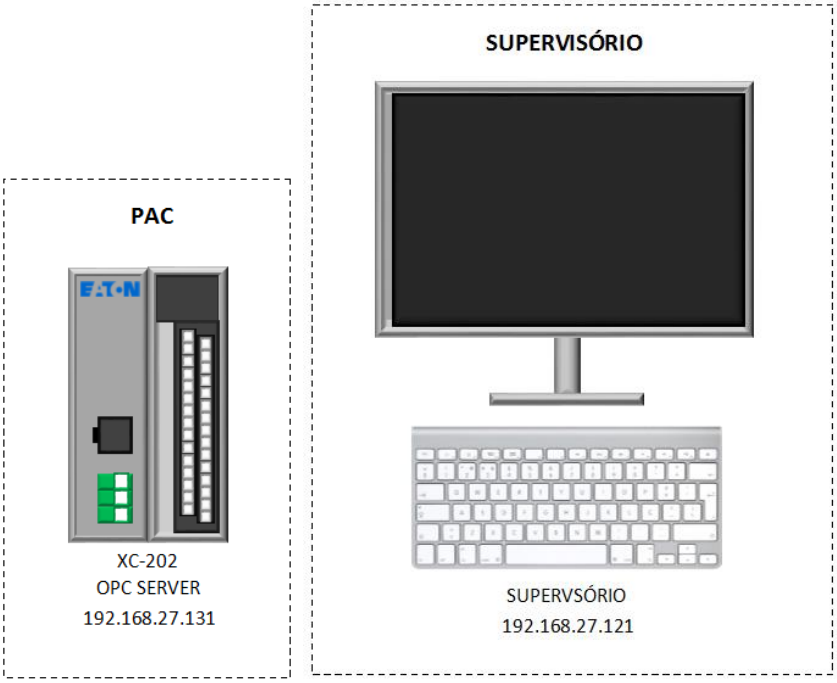
Figura 30 - Mapeamento de rede do Soprador



Fonte: Própria

Também foi mapeado com o IP: 192.168.27.131, a CPU do Painel de Automação Central, e com o IP: 192.168.27.121, o Supervisório, como pode ser visto na ***Figura 31***.

Figura 31 - Mapeamento de rede do PAC e do Supervisório

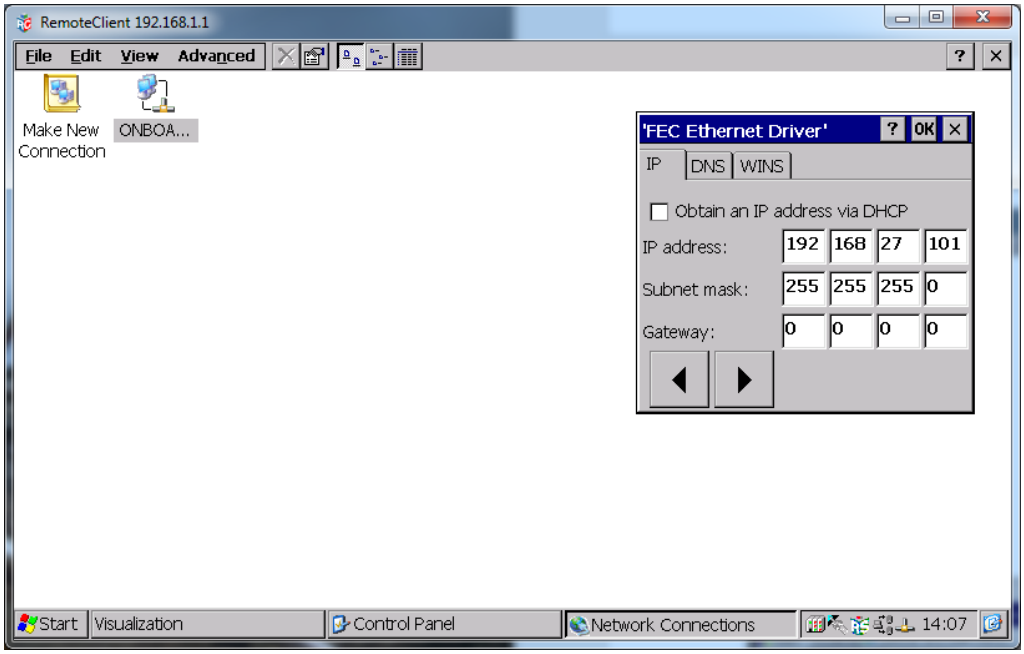


Fonte: Própria

Para o processo de mapeamento de rede das CPUs e a troca do IP original de fábrica para o novo IP foi utilizada a ferramenta "*Remote Client*", que acessa o sistema operacional (SO) das CPUs remotamente através de uma rede local e permite a configuração do driver de rede, além de outras configurações do SO. Essa ferramenta faz parte do pacote de softwares instalados juntamente com o Codesys 3.5.

A ***Figura 32*** demonstra um acesso às configurações do *driver* de rede da CPU, através do *Remote Client*.

Figura 32 - Ferramenta de acesso a CPU (*Remote Client*)

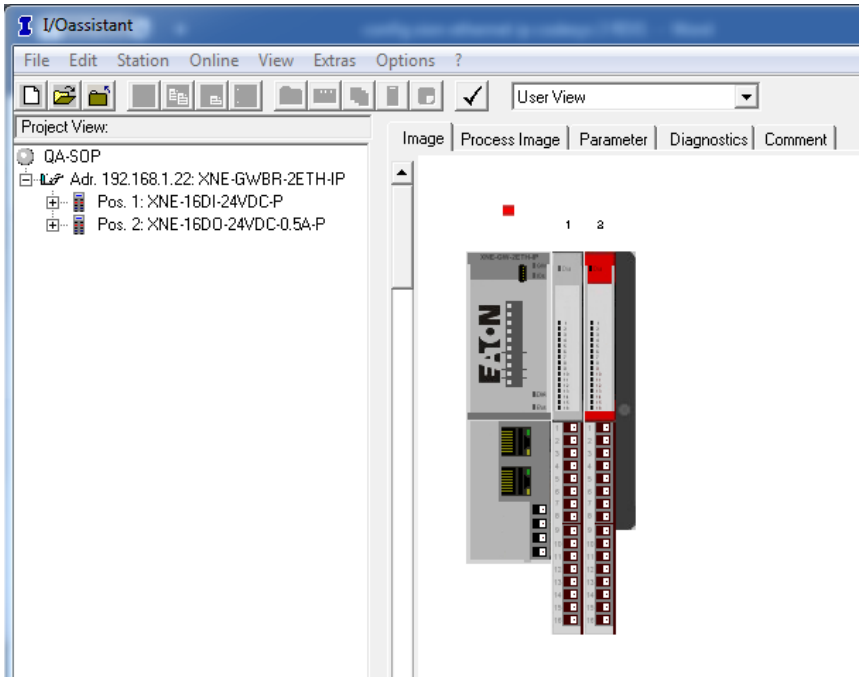


Fonte: Própria

O processo de mapeamento das remotas é realizado através de um software chamado *I/O Assistant*, da fabricante EATON, e seu download é gratuito. Iniciando o software, combina-se o gateway escolhido com as entradas e saídas do projeto e se conecta a elas, possibilitando configuração de rede e teste das entradas e saídas inseridas.

A ***Figura 33*** demonstra essa ferramenta, com a configuração de *hardware* do Soprador.

Figura 33 - Ferramenta de configuração das remotas



Fonte: Própria

3.9 Programação e testes de comunicação da CPU e da IHM

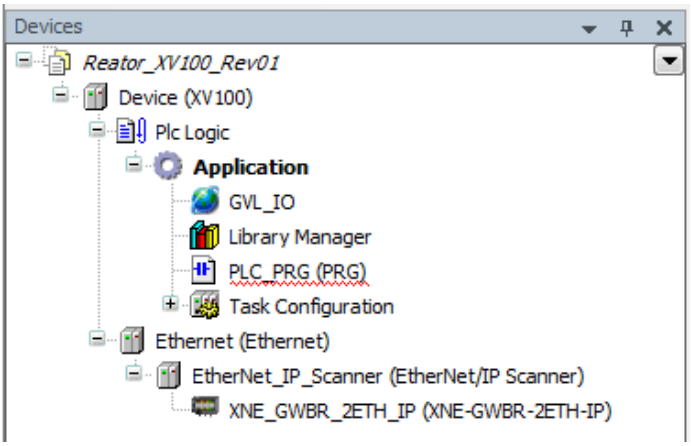
A programação da CPU foi realizada no Codesys, assim como a programação da IHM foi realizada no Galileu, ambas são descritas nas seções a seguir.

3.9.1 Configuração de Hardware no ambiente Codesys

O primeiro passo realizado foi a configuração de *hardware* no Codesys 3.5. Para isso se inseriu o "*Device*", *hardware* utilizado fisicamente no projeto, e o *firmware* foi atualizado para a última versão disponível. Na sequência foi inserido o objeto de comunicação da CPU e também o *gateway* responsável pela comunicação entre a CPU e os cartões remotos de entradas e saídas.

A ***Figura 34*** representa essa topologia de configuração no CODESYS 3.5.

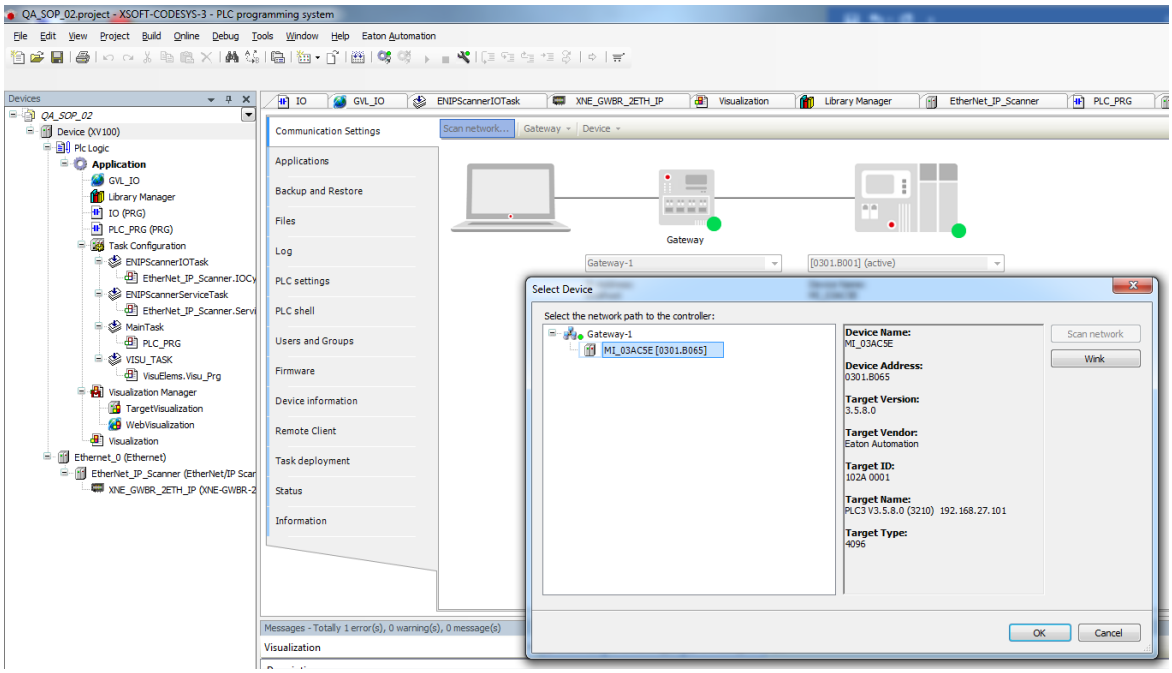
Figura 34 – Topologia de hardware no Codesys 3.5



Fonte: Própria

Após essa configuração inicial de *hardware* dentro da plataforma de programação, testou-se a comunicação entre o software e a CPU, validando o reconhecimento e a comunicação entre as entradas e saídas e o CLP, por meio do mapeamento de *IOs*. A ***Figura 35*** representa a conexão entre o PC e a CPU.

Figura 35 - Comunicação entre PC e CPU



Fonte: Própria

3.9.2 Criação das variáveis

Assim como em outras ferramentas de programação, no Codesys 3.5 as variáveis são divididas em locais e globais. As variáveis de lógica internas podem ser colocadas no grupo de locais, porém as variáveis que irão ser trocadas com outros CLPs e com outros equipamentos devem ser colocadas na lista de variáveis globais (GVL).

Estas variáveis foram distribuídas em seus respectivos *hardwares*. O reator além de utilizar entradas e saídas digitais para os sensores das válvulas motorizadas e para as indicações de falhas das motobombas, também utiliza a troca de variáveis internamente por meio de um arquivo de símbolos (.xml) entre o CLP e a IHM, como pode ser visto na ***Figura 36***.

Figura 36 - Arquivo .XML de configuração de símbolo

```
- <Node name="Application">
  - <Node name="IHM">
      <Node name="BT_Abre_Valvula_Entrada" access="ReadWrite" type="T_BOOL"/>
      <Node name="BT_Abre_Valvula_Saida" access="ReadWrite" type="T_BOOL"/>
      <Node name="BT_Desliga_Bomba_Entrada" access="ReadWrite" type="T_BOOL"/>
      <Node name="BT_Desliga_Bomba_Lodo" access="ReadWrite" type="T_BOOL"/>
      <Node name="BT_Fecha_Valvula_Entrada" access="ReadWrite" type="T_BOOL"/>
      <Node name="BT_Fecha_Valvula_Saida" access="ReadWrite" type="T_BOOL"/>
      <Node name="BT_Liga_Bomba_Entrada" access="ReadWrite" type="T_BOOL"/>
      <Node name="BT_Liga_Bomba_Lodo" access="ReadWrite" type="T_BOOL"/>
      <Node name="CS_Automatico_Bomba_Entrada" access="ReadWrite" type="T_BOOL"/>
      <Node name="CS_Automatico_Bomba_Lodo" access="ReadWrite" type="T_BOOL"/>
      <Node name="CS_Local_Reator" access="ReadWrite" type="T_BOOL"/>
      <Node name="CS_Manual_Bomba_Entrada" access="ReadWrite" type="T_BOOL"/>
      <Node name="CS_Manual_Bomba_Lodo" access="ReadWrite" type="T_BOOL"/>
      <Node name="CS_Remoto_Reator" access="ReadWrite" type="T_BOOL"/>
      <Node name="RL_Aciona_Bomba_Entrada" access="ReadWrite" type="T_BOOL"/>
      <Node name="RL_Aciona_Bomba_Lodo" access="ReadWrite" type="T_BOOL"/>
      <Node name="RL_Aciona_Valvula_Entrada" access="ReadWrite" type="T_BOOL"/>
      <Node name="RL_Aciona_Valvula_Saida" access="ReadWrite" type="T_BOOL"/>
      <Node name="RL_Bomba_Entrada_Falha" access="ReadWrite" type="T_BOOL"/>
      <Node name="RL_Bomba_Lodo_Falha" access="ReadWrite" type="T_BOOL"/>
      <Node name="RL_Desaciona_Bomba_Entrada" access="ReadWrite" type="T_BOOL"/>
      <Node name="RL_Desaciona_Bomba_Lodo" access="ReadWrite" type="T_BOOL"/>
      <Node name="RL_Desaciona_Valvula_Entrada" access="ReadWrite" type="T_BOOL"/>
      <Node name="RL_Desaciona_Valvula_Saida" access="ReadWrite" type="T_BOOL"/>
      <Node name="Sensor_Nivel" access="ReadWrite" type="T_INT"/>
      <Node name="Sensor_Oxigenio" access="ReadWrite" type="T_INT"/>
      <Node name="Sensor_PH" access="ReadWrite" type="T_INT"/>
      <Node name="T_Ligado_Bomba_Entrada" access="ReadWrite" type="T_TIME"/>
      <Node name="T_Ligado_Bomba_Lodo" access="ReadWrite" type="T_TIME"/>
    </Node>
  - <Node name="PLC_PRG">
      <Node name="S_Aberta_Valvula_Entrada" access="ReadWrite" type="T_BOOL"/>
      <Node name="S_Aberta_Valvula_Saida" access="ReadWrite" type="T_BOOL"/>
      <Node name="S_Fechada_Valvula_Entrada" access="ReadWrite" type="T_BOOL"/>
      <Node name="S_Fechada_Valvula_Saida" access="ReadWrite" type="T_BOOL"/>
    </Node>
</Node>
```

Fonte: Própria

As variáveis encontradas no arquivo .xml são geradas pelo CODESYS, inicialmente declarando-as como “Variáveis Globais (GVL)”, como pode ser vista na ***Figura 37***.

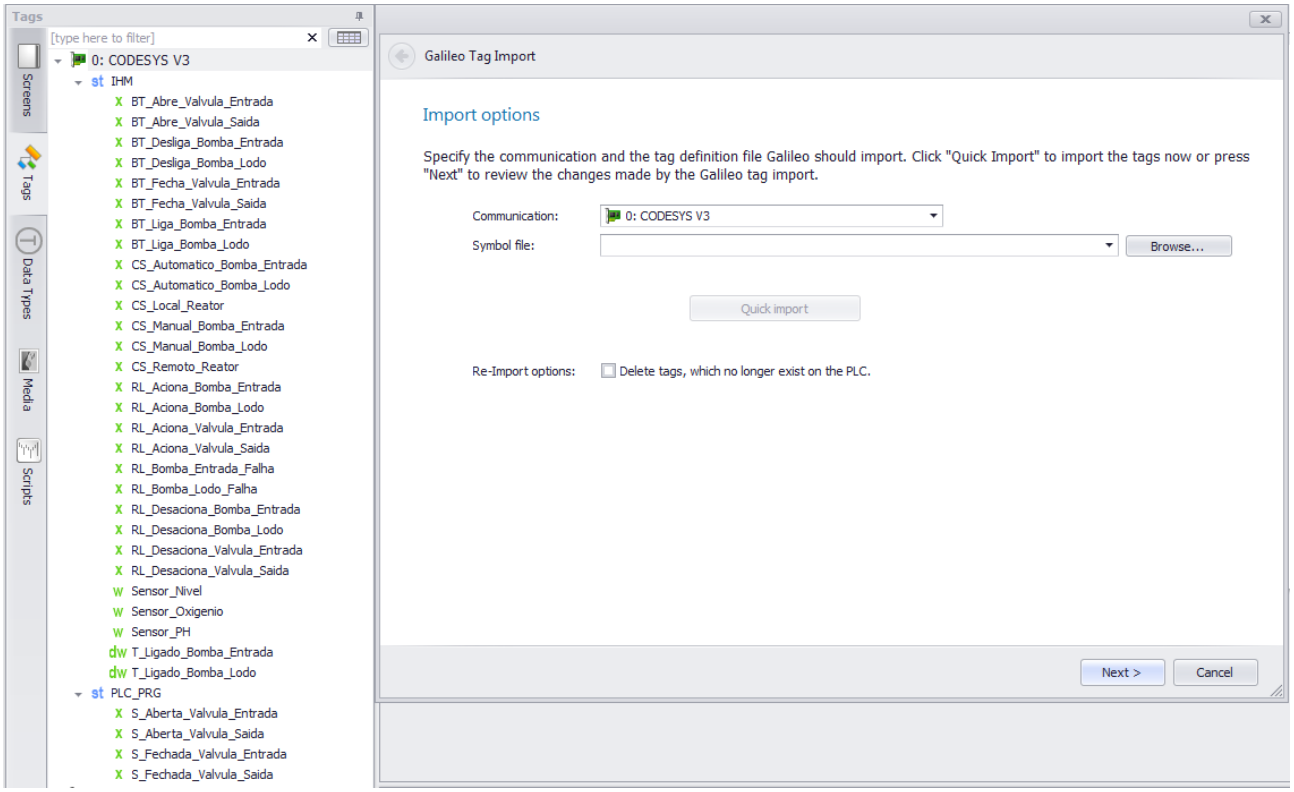
Figura 37 - Variáveis de controle do Reator para exportação ao Galileo

```
{attribute 'qualified_only'}
VAR_GLOBAL

    //Status
    CS_Local_Reator                 : BOOL;
    CS_Remoto_Reator                : BOOL;

    //Válvula de Entrada
    BT_Abre_Valvula_Entrada         : BOOL;
    BT_Fecha_Valvula_Entrada        : BOOL;
    RL_Aciona_Valvula_Entrada       : BOOL;
    RL_Desaciona_Valvula_Entrada    : BOOL;

    //Válvula de Saída
    BT_Abre_Valvula_Saida           : BOOL;
    BT_Fecha_Valvula_Saida          : BOOL;
    RL_Aciona_Valvula_Saida         : BOOL;
    RL_Desaciona_Valvula_Saida      : BOOL;

    //Bomba de Lodo
    BT_Liga_Bomba_Lodo              : BOOL;
    BT_Desliga_Bomba_Lodo           : BOOL;
    CS_Automatico_Bomba_Lodo        : BOOL;
    CS_Manual_Bomba_Lodo            : BOOL;
    RL_Aciona_Bomba_Lodo            : BOOL;
    RL_Desaciona_Bomba_Lodo         : BOOL;
    RL_Bomba_Lodo_Falha             : BOOL;
    T_Ligado_Bomba_Lodo             : TIME;

    //Bomba de Entrada
    BT_Liga_Bomba_Entrada           : BOOL;
    BT_Desliga_Bomba_Entrada        : BOOL;
    CS_Automatico_Bomba_Entrada     : BOOL;
    CS_Manual_Bomba_Entrada         : BOOL;
    RL_Aciona_Bomba_Entrada         : BOOL;
    RL_Desaciona_Bomba_Entrada      : BOOL;
    RL_Bomba_Entrada_Falha          : BOOL;
    T_Ligado_Bomba_Entrada          : TIME;

    //Sensores
    Sensor_Nivel                    : INT; //Nível
    Sensor_PH                       : INT; //PH
    Sensor_Oxigenio                 : INT; //Oxigênio

END_VAR
```

Fonte: Própria

O arquivo de símbolos após criado no Codesys 3 (ambiente de programação do CLP) foi importado para o Galileo 10 (ambiente de programação da IHM) para a associação aos objetos de interface criados, através do arquivo de símbolos (.xml). A ***Figura 38*** demonstra a tela de importação de tags no software Galileo.

Figura 38 - Importação de tags no Galileo



Fonte: Própria

3.9.3 Telas da IHM

As telas de controle e operação da IHM foram criadas obedecendo à necessidade de comandos e sinalizações presentes no memorial de operação da ETE Jarivatuba, fornecido pela Águas de Joinville.

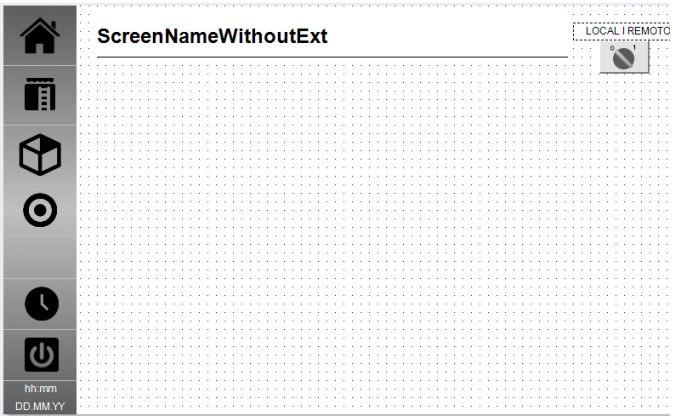
A ***Figura 39*** demonstra a tela inicial da IHM com o menu de navegação de telas à esquerda, onde as seguintes telas podem ser acessadas:

1. Tela inicial
2. Tela do reator
3. Tela das bombas
4. Tela das válvulas

5. Tela de configuração de data e hora
6. Sair

Nessa primeira tela pode se controlar o modo de trabalho do reator, modo “Remoto” ou em modo “Local”, através do clique no botão do canto superior a direita.

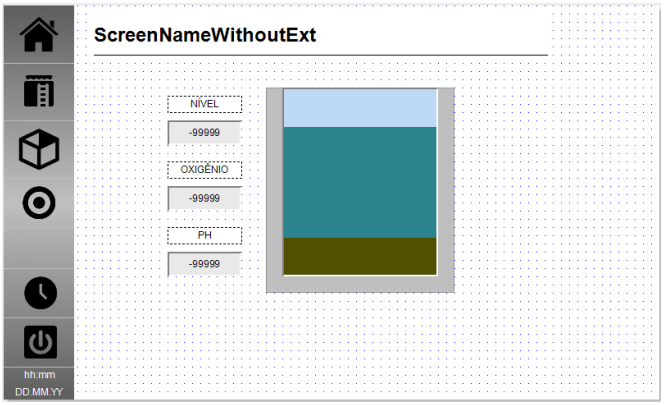
Figura 39 – Tela inicial de controle da IHM do reator



Fonte: Própria

Na “Tela do Reator” encontra-se variáveis analógicas de leitura, tais como nível, oxigênio e PH. A ***Figura 40*** demonstra essas variáveis.

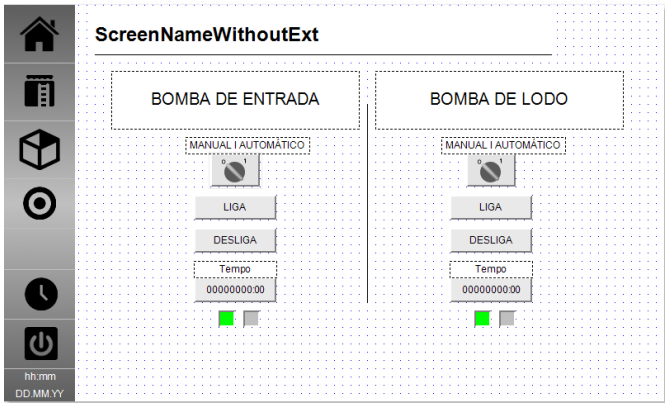
Figura 40 - Tela do Reator



Fonte: Própria

Na “Tela das bombas” é possível controlar o estado de operação, “manual” e “automático”, os botões de “liga” e “desliga” e também se encontra a variável de tempo que pode ser configurada pelo operador caso a operação esteja em automático. A ***Figura 41*** demonstra esses controles.

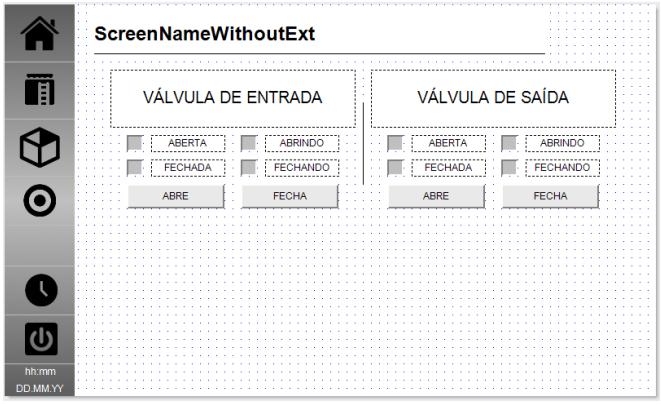
Figura 41 - Tela das bombas



Fonte: Própria

Na “Tela das válvulas” é possível executar apenas dois comandos, o comando de “Abrir” e o comando de “fechar” a válvulas. Também nessa tela se encontram as sinalizações de estado das válvulas, “Aberta”, “Fechada”, “Abrindo” e “Fechando”. A ***Figura 42*** demonstra esses controles e sinalizações.

Figura 42 - Tela das válvulas



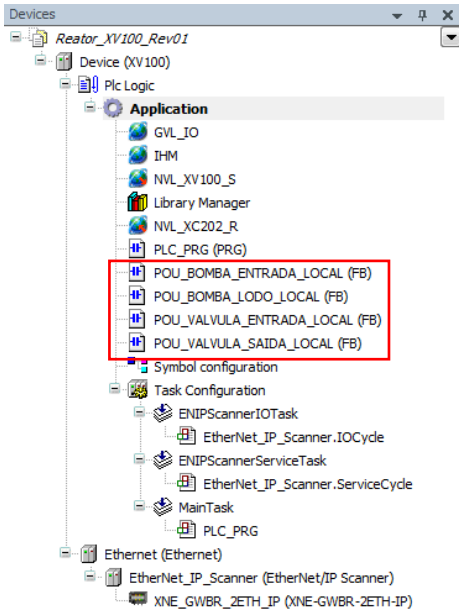
Fonte: Própria

3.9.4 Programando as CPUs

Através da modelagem, iniciou-se a lógica de programação do CLP, primeiramente com o controlador do reator e em um segundo momento com o controlador do soprador. Foi utilizado a linguagem *ladder* na programação e os modelos foram implementados em blocos de funções POU (*Program Ogarnization Unit*).

A ***Figura 43*** representa as POUs criadas para a motobomba de entrada, bomba de lodo, válvula de entrada e válvula de saída.

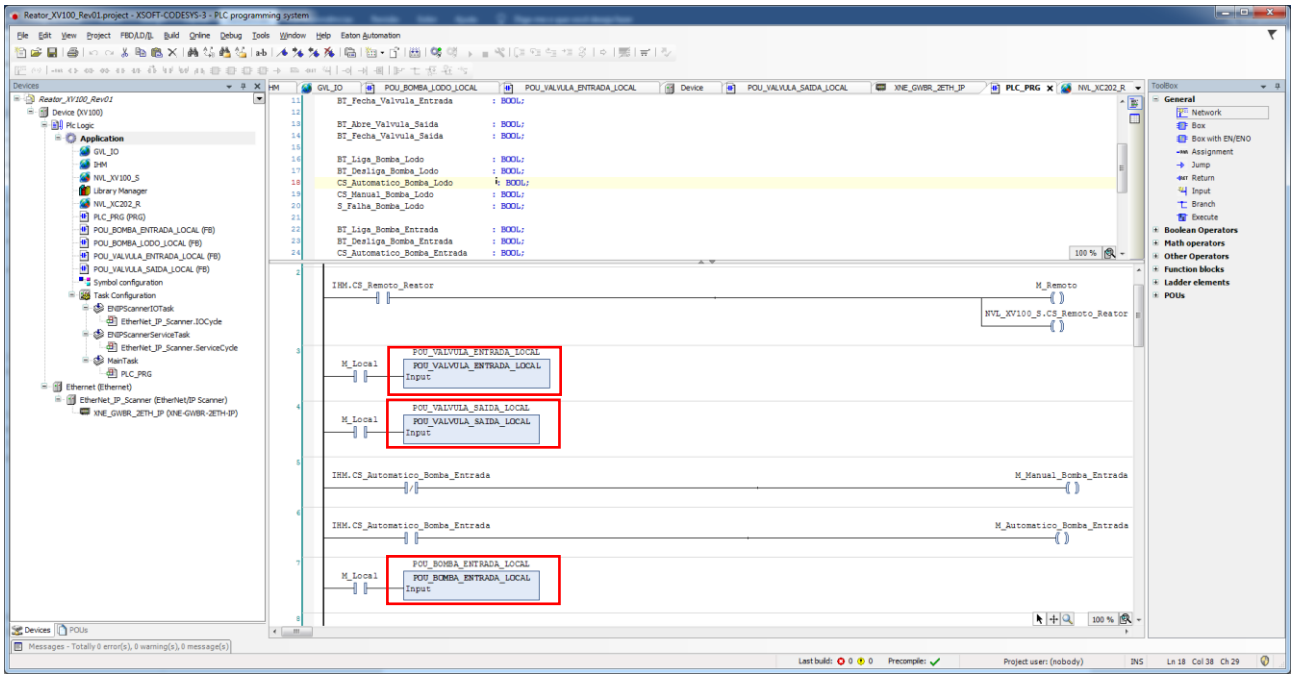
Figura 43 - Conjunto de POUs implementadas no controlador do reator



Fonte: Própria

Essas POUs são visualizadas dentro da lógica de programação por caixas, e internamente outras lógicas com avaliações de entradas e comandos em saídas acontecem. As POUs ajudam a replicar uma mesma lógica em outra aplicação ou mesmo para um equipamento que se repete na planta. Na ***Figura 44*** encontram-se as POUS das válvulas e das motobombas controladas pelo CLP do Reator.

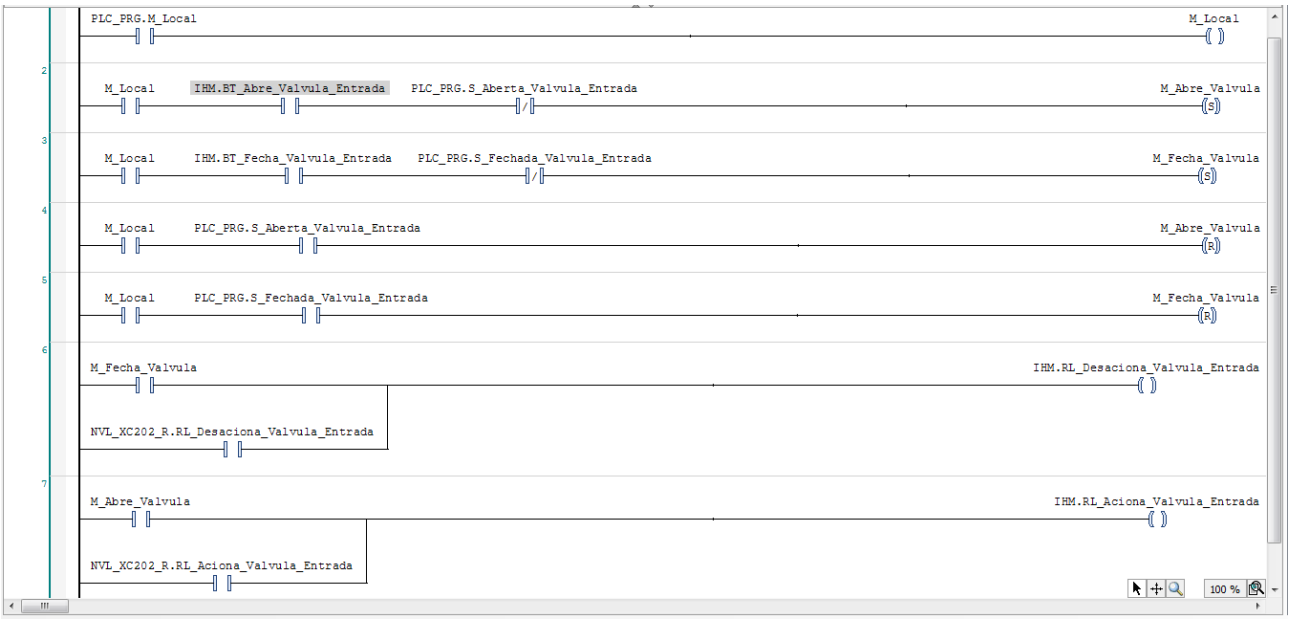
Figura 44 - POUs representadas pelos blocos de funções



Fonte: Própria

A ***Figura 45*** representa a programação interna do bloco de programação (POU) da válvula de entrada do Reator.

Figura 45 - Maximização do bloco de função "POU_VALVULA_DE_ENTRADA"



Fonte: Própria

3.10 Implementação do middleware

O *middleware*, objeto de estudo dessa dissertação, é responsável por coletar as informações do processo, por meio de diversos protocolos de comunicação e centralizá-los no padrão de comunicação OPC *Classic*. Também é sua função disponibilizar as variáveis de controle e operação de acordo com a seleção de grupos e usuários. Para isso, o *hardware* centralizador necessita ser um servidor OPC DA para oferecer e gerenciar as informações que serão entregues aos clientes.

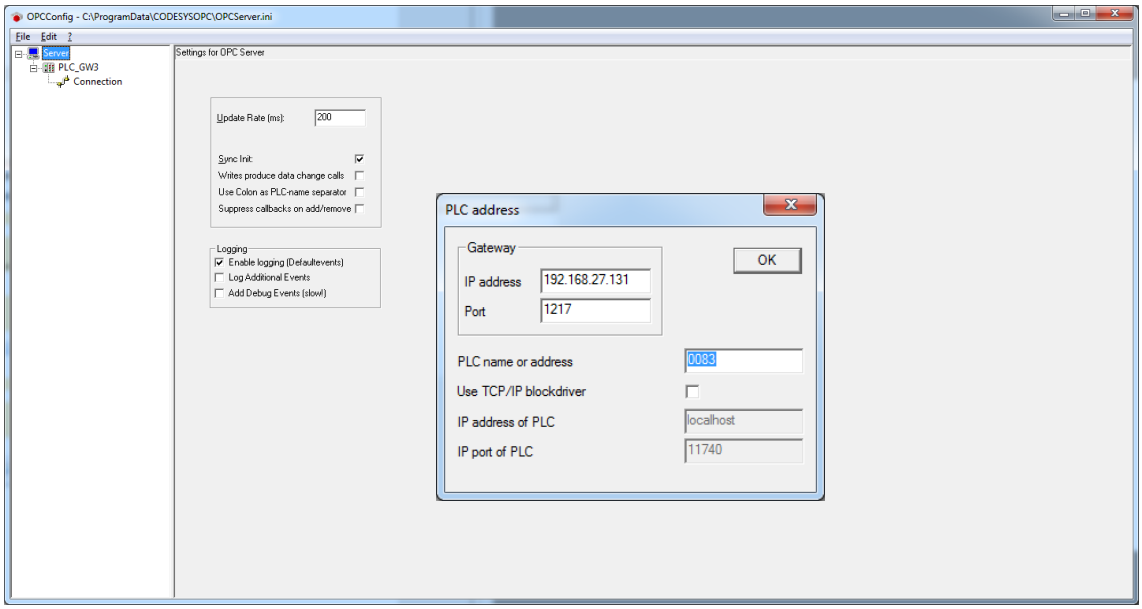
3.10.1 Servidor OPC DA

Inicialmente voltou-se para a implantação do servidor OPC DA. Como apresentado na seção 3.4 (Arquitetura dos *hardwares*), a CPU XC-202 foi utilizada como servidor OPC DA e responsável pela interoperabilidade do sistema.

O servidor OPC DA foi implementado utilizando a ferramenta *OPC Configuration*, que compõe o pacote de softwares do CODESYS 3.5. Nessa ferramenta foi configurada uma nova conexão com

a rede desejada, permitindo o acesso do CLP XC-202. A ***Figura 46*** demonstra essa nova configuração da conexão do servidor OPC DA que será carregada no CLP XC-202.

Figura 46 - Configuração de conexão do servidor OPC



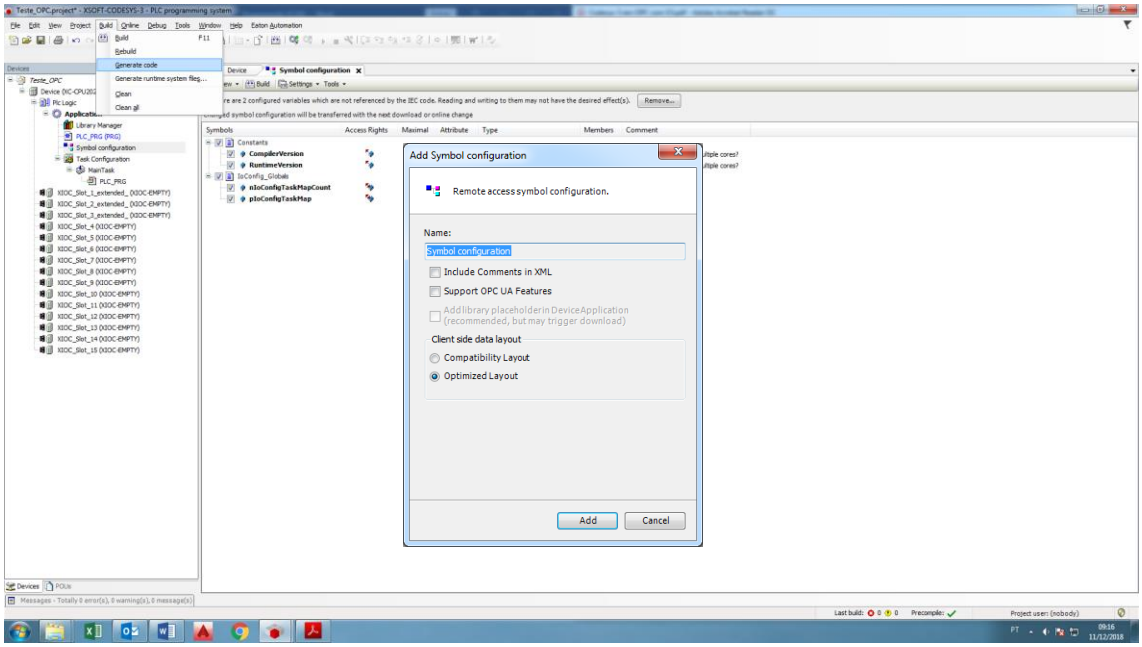
Fonte: Própria

O *OPC Configuration* é uma ferramenta que gera um arquivo com o formato .ini e ao se conectar ao CLP para a realização do download da programação esse arquivo é carregado para a CPU do CLP, transferindo a configuração de comunicação do servidor OPC.

3.10.2 Variáveis disponibilizadas pelo middleware

No padrão de comunicação OPC *Classic*, a troca de variáveis realizada entre o servidor e os clientes é disponibilizada através de um arquivo de símbolos. Esse arquivo contém as propriedades, os grupos e os itens, componentes característicos da modelagem de um *middleware* OPC. A ***Figura 47*** demonstra a implementação do arquivo de símbolos do servidor OPC DA.

Figura 47 - Arquivo de símbolos



Fonte: Própria

3.10.3 Modelagem do middleware em OPC *Classic*

A modelagem do *middleware* utilizando o OPC *Classic* tem uma abordagem bastante simplista, uma vez que os dados são acessados a partir de uma estrutura em árvore, onde cada dispositivo carrega suas variáveis expansíveis.

Essa estrutura em árvore aborda dois conceitos, primeiramente o conteito de *Group*, e posteriormente o conceito de *Item*. Na modelagem do *middleware* os grupos são os nós NVL_XC152_R, NVL_XV100_R e Supervisorio_OPC, e os itens são propriamente as variáveis a serem trocadas entre os dispositivos. Essa estrutura pode ser observada na ***Figura 48***.

Figura 48 - Estrutura de modelagem do middleware OPC *Classic*

```
<Node name="NVL_XC152_R">
  <Node name="Local_Soprador" type="T_BOOL" access="ReadWrite" />
  <Node name="Remoto_Soprador" type="T_BOOL" access="ReadWrite" />
</Node>
<Node name="NVL_XV100_R">
  <Node name="CS_Local_Reator" type="T_BOOL" access="ReadWrite" />
  <Node name="CS_Remoto_Reator" type="T_BOOL" access="ReadWrite" />
  <Node name="S_Aberta_Valvula_Entrada" type="T_BOOL" access="ReadWrite" />
  <Node name="S_Aberta_Valvula_Saida" type="T_BOOL" access="ReadWrite" />
  <Node name="S_Falha_Bomba_Entrada" type="T_BOOL" access="ReadWrite" />
  <Node name="S_Falha_Bomba_Lodo" type="T_BOOL" access="ReadWrite" />
  <Node name="S_Fechada_Valvula_Entrada" type="T_BOOL" access="ReadWrite" />
  <Node name="S_Fechada_Valvula_Saida" type="T_BOOL" access="ReadWrite" />
  <Node name="Sensor_Nivel" type="T_INT" access="ReadWrite" />
  <Node name="Sensor_Oxigenio" type="T_INT" access="ReadWrite" />
  <Node name="Sensor_PH" type="T_INT" access="ReadWrite" />
```

```
<Node name="Supervisorio_OPC">
  <Node name="BT_Abre_Valvula_Entrada" type="T_BOOL" access="ReadWrite" />
  <Node name="BT_Abre_Valvula_Saida" type="T_BOOL" access="ReadWrite" />
  <Node name="BT_Automatico_Soprador" type="T_BOOL" access="ReadWrite" />
  <Node name="BT_Desliga_Bomba_Entrada" type="T_BOOL" access="ReadWrite" />
  <Node name="BT_Desliga_Bomba_Lodo" type="T_BOOL" access="ReadWrite" />
  <Node name="BT_Desliga_Soprador" type="T_BOOL" access="ReadWrite" />
  <Node name="BT_Fecha_Valvula_Entrada" type="T_BOOL" access="ReadWrite" />
  <Node name="BT_Fecha_Valvula_Saida" type="T_BOOL" access="ReadWrite" />
  <Node name="BT_Liga_Bomba_Entrada" type="T_BOOL" access="ReadWrite" />
  <Node name="BT_Liga_Bomba_Lodo" type="T_BOOL" access="ReadWrite" />
  <Node name="BT_Liga_Soprador" type="T_BOOL" access="ReadWrite" />
  <Node name="BT_Manual_Soprador" type="T_BOOL" access="ReadWrite" />
  <Node name="BT_Tempo_Ligado_Soprador" type="T_TIME" access="ReadWrite" />
  <Node name="CS_Automatico_Bomba_Entrada" type="T_BOOL" access="ReadWrite" />
  <Node name="CS_Automatico_Bomba_Lodo" type="T_BOOL" access="ReadWrite" />
  <Node name="CS_Manual_Bomba_Entrada" type="T_BOOL" access="ReadWrite" />
  <Node name="CS_Manual_Bomba_Lodo" type="T_BOOL" access="ReadWrite" />
  <Node name="RL_Abrindo_Valvula_Entrada" type="T_BOOL" access="ReadWrite" />
  <Node name="RL_Abrindo_Valvula_Saida" type="T_BOOL" access="ReadWrite" />
  <Node name="RL_Deligada_Bomba_Entrada" type="T_BOOL" access="ReadWrite" />
  <Node name="RL_Deligada_Bomba_Lodo" type="T_BOOL" access="ReadWrite" />
  <Node name="RL_Desligado_Soprador" type="T_BOOL" access="ReadWrite" />
  <Node name="RL_Falha_Bomba_Entrada" type="T_BOOL" access="ReadWrite" />
  <Node name="RL_Falha_Bomba_Lodo" type="T_BOOL" access="ReadWrite" />
  <Node name="RL_Fechando_Valvula_Entrada" type="T_BOOL" access="ReadWrite" />
  <Node name="RL_Fechando_Valvula_Saida" type="T_BOOL" access="ReadWrite" />
  <Node name="RL_Ligada_Bomba_Entrada" type="T_BOOL" access="ReadWrite" />
  <Node name="RL_Ligada_Bomba_Lodo" type="T_BOOL" access="ReadWrite" />
  <Node name="RL_Ligado_Soprador" type="T_BOOL" access="ReadWrite" />
  <Node name="T_Ligado_Bomba_Entrada" type="T_TIME" access="ReadWrite" />
  <Node name="T_Ligado_Bomba_Lodo" type="T_TIME" access="ReadWrite" />
</Node>
```

Fonte: Própria

3.10.4 Segurança dos dispositivos

O padrão de comunicação OPC *Classic* utiliza as ferramentas de segurança do Windows, uma vez que são baseadas em DCOM/COM. Para garantir a segurança e interoperabilidade do sistema utilizou-se o pacote CODESYS *Security* que ao ser implementado permite criptografia do código-fonte e também o gerenciamento de usuários e autenticação.

O pacote CODESYS *Security* da desenvolvedora 3S-Smart Software Solutions, atende à IEC 62443 nos níveis um e dois, dos quatros níveis de proteções definidos pela norma:

Nível 1 – ameaças ocasionais e acidentais

Nível 2 – ameaças intencionais por meios simples

Nível 3 – ameaça intencional por meios altamente desenvolvidos

Nível 4 – ameaça intencional por meios altamente desenvolvido e recursos estendidos

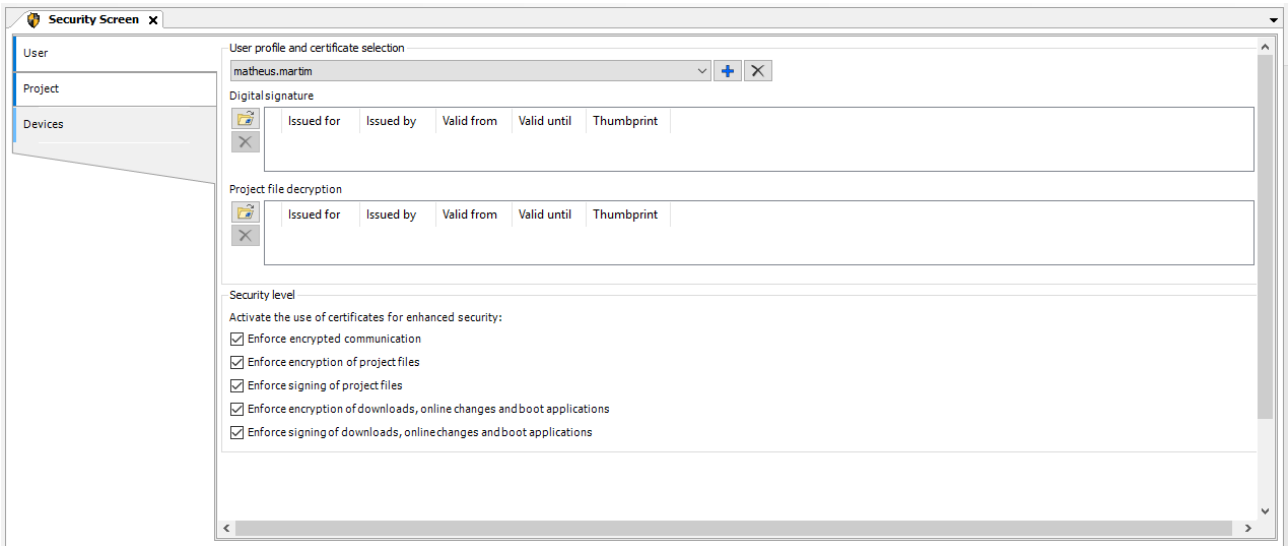
Criptografia do código-fonte

O código-fonte da aplicação contém as informações e a lógica de operação da planta, ou seja, toda a propriedade intelectual desenvolvida pela empresa, portanto possui alta prioridade de segurança, uma vez que contém informações confidenciais.

O CEDESYS 3.5.12 oferece a proteção do código fonte utilizando encriptação através de certificados digitais. As configurações de segurança são encontradas na tela de configuração *Security Screen*.

Incialmente foi selecionado os níveis de segurança que iriam possuir encriptação e em seguida o certificado a ser utilizado. A ***Figura 49*** mostra a tela de seleção do nível de segurança do usuário *matheus.martim*, outros usuários podem ser configurados com diferentes permissões e também diferentes certificações podem ser utilizadas.

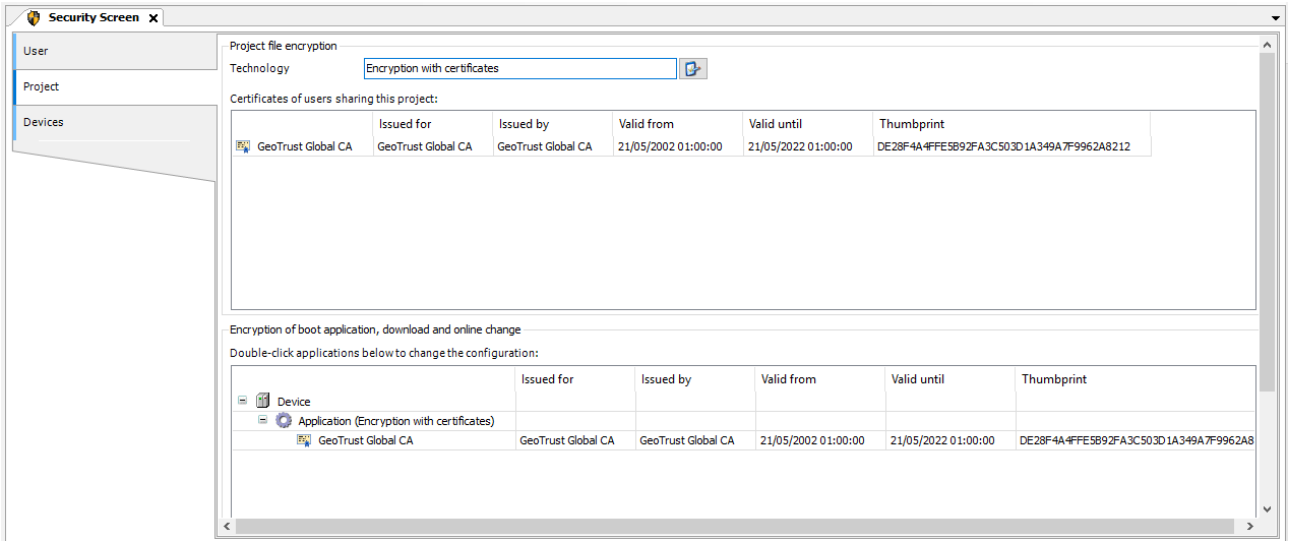
Figura 49 - Security Screen - User



Fonte: Própria

A ***Figura 50*** demonstra que a tecnologia de segurança utilizada para o projeto foi a encriptação com certificado. O certificado para os usuários compartilharem esse projeto foi o GeoTrustGlobalCA e abaixo seleciona a aplicação que se deseja encriptar.

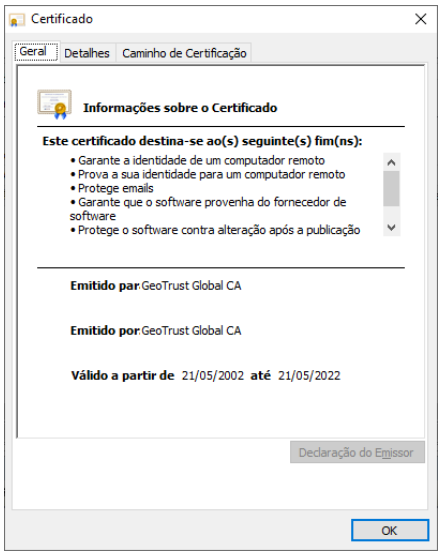
Figura 50 - Security Screen - Project



Fonte: Própria

A ***Figura 51*** apresenta o certificado de encriptação e suas informações de destinação, emissão e validade.

Figura 51 - Certificado de encriptação GeoTrustGlobalCA

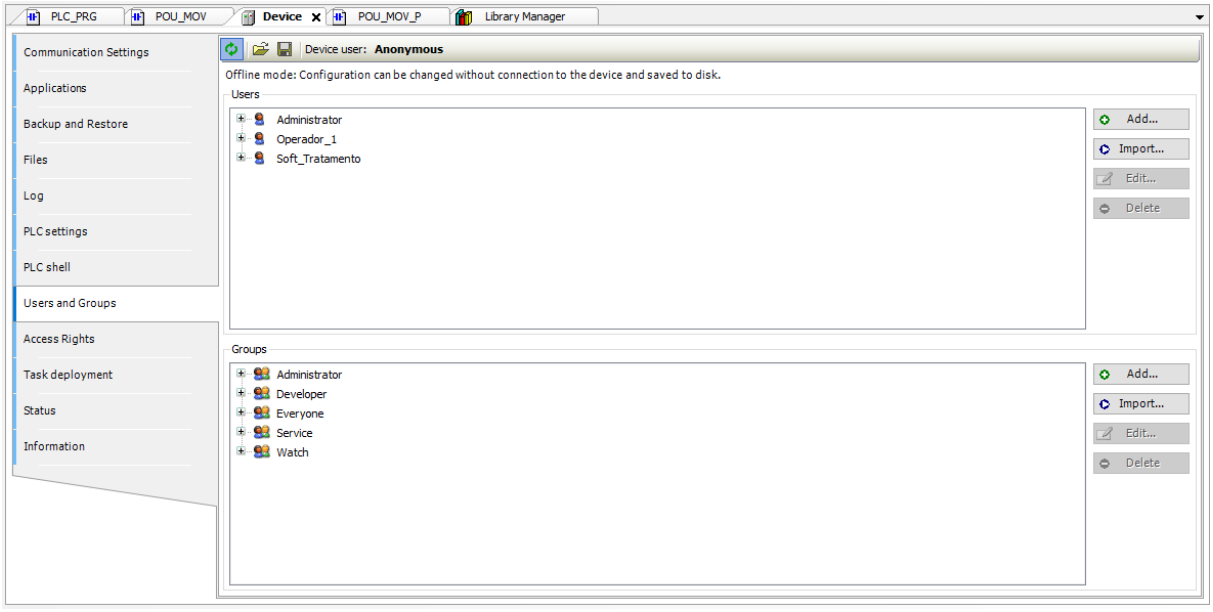


Fonte: Própria

Gerenciamento de usuários e autenticação

O gerenciamento de usuários foi realizado por meio da configuração de *Groups* e *Users*. Os seguintes grupos foram criados: *Administrator*, *Developer*, *Service* e *Watch*. A ***Figura 52*** demonstra o gerenciamento dos grupos e dos usuários.

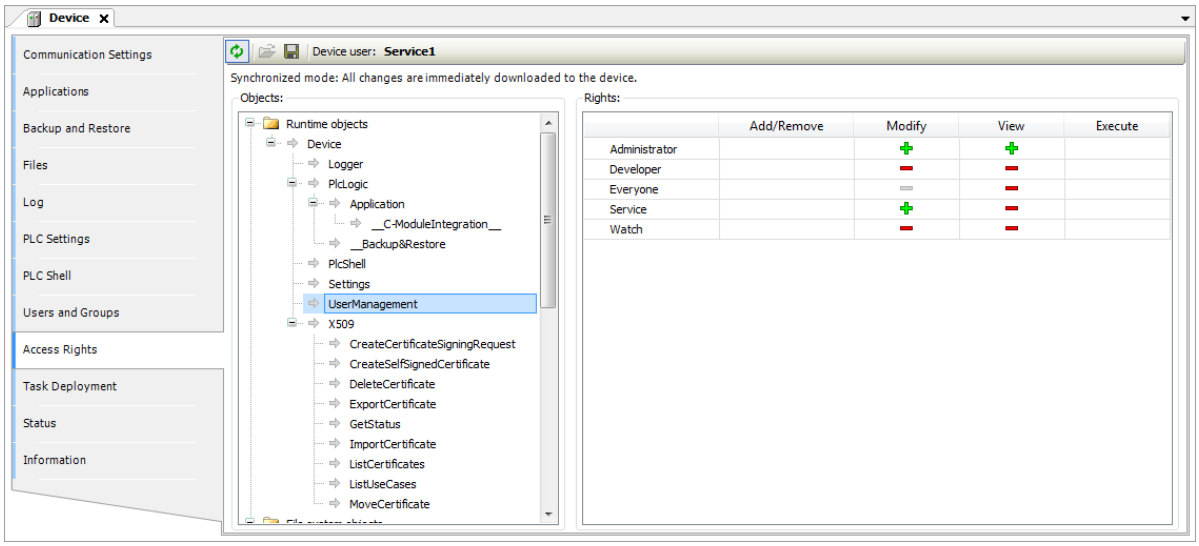
Figura 52 - Gerenciamento de grupos



Fonte: Própria

Através da aba de *Access Rights* na opção *UserManagement* foram configuradas as permissões de modificação e visualização do gerenciador de usuários. Outras configurações, tais como dispositivos, aplicações, comunicação, podem ser alteradas as permissões de modificação e visualização. A ***Figura 53*** demonstra essas configurações.

Figura 53 - Configuração de permissões de modificação e visualização



Fonte: Própria

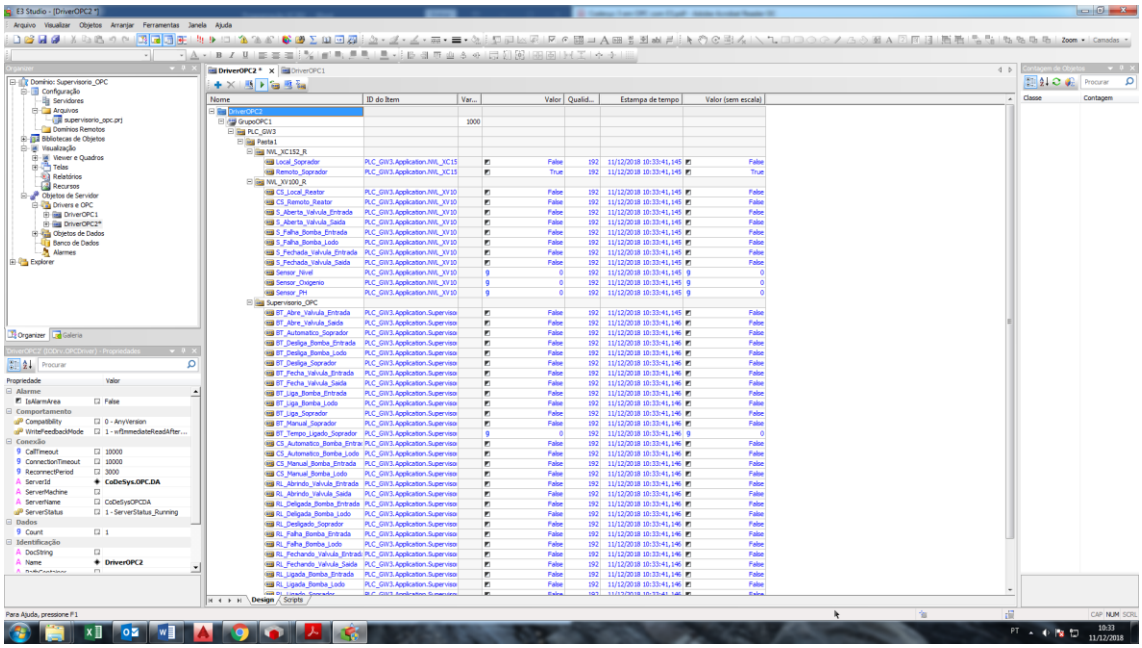
3.10.5 Testes de comunicação entre o servidor e o cliente OPC DA

Após a implementação do servidor OPC DA, voltou-se para o desenvolvimento de um cliente OPC para validar o conceito de permissão de acesso e troca de variáveis entre os dispositivos OPC.

A plataforma SCADA foi utilizada para se implementar um cliente OPC. Utilizando o software E3 da desenvolvedora Elipse inseriu-se o driver de comunicação OPC e encontrou-se o servidor OPC DA conectado à rede, denominado CoDeSys.OPC.DA. Através do processo de escaneamento importou-se as *tags* para acesso do cliente conforme suas permissões de usuário.

A ***Figura 54*** demonstra essa importação para um usuário do grupo D*eveloper*, e pode-se verificar a mesma estrutura de arvore apresentada na seção 643.10.3 (Modelagem do middleware em OPC) onde se trata da modelagem através do nós NVL_XC152_R, NVL_XV100_R e Supervisorio_OPC.

Figura 54 - Cliente OPC implementado na plataforma SCADA da Elipse com o software E3



Fonte: Própria

O cliente OPC foi configurado para realizar leitura assíncrona e por mudança de estado, isso significa que a leitura de novos dados não necessita de confirmação de execução e que o servidor é responsável por enviar aos clientes os itens que sofrerem alteração em seu estado.

A ***Tabela 10*** demonstra as configurações de permissões de cada grupo criado.

Tabela 10 - Permissões de modificação e visualização dos grupos

	ADMINISTRATOR	**DEVELOPER**	**SERVICE**
DEVICE	Modificação e Visualização	Visualização	
APPLICATION	Modificação e Visualização	Modificação e Visualização	Visualização
USERMANAGEMENT	Modificação e Visualização	Modificação e Visualização	Modificação e Visualização

Fonte: Própria

Para validação dessas configurações e confirmação da troca de dados e segurança foram realizados três testes. O primeiro teste utilizou o cliente OPC *Operador_1* que possui permissões do grupo de *Service*. Este grupo permite somente modificações e visualização dos usuários e também a visualização das variáveis da aplicação. Dois parâmetros foram conferidos nesse teste, o primeiro foi efetivamente a permissão das alterações das variáveis e confirmação desses valores e o segundo foi se somente a variável alterada era enviada do servidor OPC DA para o cliente OPC. Como esse usuário só poderia visualizar as variáveis da aplicação acompanhou-se a resposta mediante ao processo de visualização conforme as variáveis eram alteradas no servidor OPC DA. Também foi testada a modificação e visualização de usuários, como permitido para esse grupo.

O segundo teste utilizou o cliente OPC *Soft_Tratamento* que possui permissões do grupo de *Developer*. Este grupo permite modificações e visualizações das variáveis da aplicação e do gerenciamento dos usuários, porém os dispositivos (*hardwares*) somente são visualizados por este grupo. Lembra-se que o estudo de caso dessa dissertação justamente é a disponibilização das variáveis para este grupo de clientes, portanto o funcionamento das características de modificação e visualização descritas para esse grupo foram insistentemente testadas e conferidas para que o projeto garantisse a interoperabilidade e a segurança necessárias e prometidas pela empresa Pesco.

O terceiro e último utilizou o cliente OPC *Administrator*, que possui permissões do grupo *Administrator*, onde todos os dispositivos (*hardwares*), as variáveis da aplicação, e os gerenciamentos dos usuários podem ser modificados e visualizados. Esse grupo foi criado para testes e somente é utilizado pela empresa desenvolvedora do sistema. Esse grupo não é compartilhado com os clientes.

4. Conclusões

O presente trabalho de pesquisa partiu-se da necessidade de interfaceamento, por meio de um *middleware*, da empresa Pesco para a comunicação com outros softwares proprietários de tratamento de efluentes adquiridos pela CAJ. Para solucionar a necessidade foram estudadas tecnologias que pudessem englobar a solução de padronização e interfaceamento por completo.

Inicialmente a tecnologia que compreendia por completo essa necessidade levantada foi o padrão de comunicação OPC UA, que é um padrão orientado a serviços, possui equivalência funcional ao padrão OPC *Classic*, garante segurança através de criptografia, autenticação e auditoria, e permite modelagem da informação.

A solução utilizando OPC UA seria capaz de encontrar disponibilidade de servidores OPC em PCs ou em redes locais, também poderiam encontrar os dados modelados representados hierarquicamente sendo visualizados e alterados por níveis de acesso. O padrão possui independência de plataforma, ou seja, permite que seja embarcado em PCs, servidores em nuvem, CLPs e microcontroladores, além de rodar em diversos sistemas operacionais como Microsoft Windows, Apple OSX, Android ou Linux, fornecendo interoperabilidade para toda a infraestrutura.

Em relação à segurança, o padrão OPC UA é compatível com firewall e garante controle de transporte de dados, além de possuir criptografia em vários níveis. Também possui assinatura de mensagem, que permite o destinatário verificar a origem e integridade das mensagens. Para se evitar exposição a ataques de repetição, os pacotes são sequenciados e os clientes e servidores possuem certificados de autenticação X509, além de executar controle de usuários onde as atividades são registradas, fornecendo material para auditoria.

A Pesco Automação & Controle responsável pelas compras e fornecimento dos softwares e *hardwares*, inviabilizou a solução primeiramente pensada, devido a custos de implementação da solução, porém uma segunda hipótese foi levantada juntamente com a fornecedora de equipamentos de automação (EATON), onde utilizou-se o padrão de comunicação OPC *Classic*.

O padrão de comunicação OPC C*lassic* é baseado em DCOM, portanto todas as ferramentas facilitam à implementação para o sistema operacional Windows. Assim como no OPC UA, o OPC *Classic* também pode ser configurado para fornecer assinatura de dados e criptografia o que garante maior segurança de dados. Além da encriptação a identificação e segurança de aplicações também pode ser configurada no OPC *Classic*, o que exige um alto nível de trabalho de configuração de itens diferentes do DCOM, como firewalls e RPC.

A utilização de firewalls é dificilmente implementada para o padrão OPC *Classic*, uma vez que se utiliza o RPC por padrão, e o mesmo faz uso de alocação dinâmica de porta, o que dificulta a configuração de um firewall, pois uma grande variedade de portas precisaria ser aberta e configurada.

Para garantir também registros de auditoria que permitem uma análise das ações em um sistema após a solução de algum problema, como fornecido pelo padrão OPC UA é necessário que os usuários habilitem o log para solicitações de conexão DCOM, acesso a objetos do sistema e *logons* SMB, permitindo que o sistema operacional gere eventos que possam ser usados para rastrear as atividades.

São notáveis as vantagens de implementação do padrão de comunicação OPC UA mediante a comparação com seu antecessor tecnológico OPC *Classic*, porém nada impede que medidas de segurança possam ser implementadas garantindo a robustez e estabilidade ao sistema. O tempo de desenvolvimento se torna maior mediante a muitas configurações necessárias para se garantir o alto nível de segurança, além de que o sistema só pode ser utilizado em plataformas Windows, inviabilizando completa interoperabilidade, como garantido pelo padrão de comunicação OPC UA.

O presente trabalho garantiu um *middleware* capaz de suprir as necessidades de interfaceamento e padronização de comunicação entre o sistema de controle da planta da estação de tratamento de esgotos de Joinville e softwares proprietários que necessitarem de variáveis internas do gerenciamento e da qualidade do esgoto tratado.

4.1 Trabalhos futuros

Deixa-se como sugestões de trabalhos futuros a implementação do *middleware* utilizando o padrão de comunicação OPC UA, validando as comparações realizadas acima e garantindo a interoperabilidade para diversos sistemas operacionais.

5. Referências

Araújo, M. R. (2016). *Implementação e análise de um middleware baseado em RPC utilizando Erlang.* Universidade Federal de Pernanbuco, Recife.

Azevedo, M. T. (2017). *Transformação Digital na Indústria: Indústria 4.0 e a Rede de Água Inteligente no Brasil.* Tese (Tese em Ciências) - Escola Politécnica da USP, São Paulo.

Bakken, D. E. (2003). *Middleware.* Washington State University, Pullman.

Branquinho, M. A., Moraes, L. C., Seidl, J., Junior, J. A., & Branquinho, T. B. (2014). *Segurança de Automação Industrial e SCADA* (1ª ed.). Elsevier Brasil.

Costa, D. d. (2009). *Sistemas de supervisão e controlo integrados.* Dissertação - Universidade de Aveiro, Aveiro. Portugal.

Filho, M. d. (2009). *Aplicação de Conceitos de Middleware em Redes de Controle Lonworks/EIA 709.1 para Automação de Ambientes Pervasivos.* Tese (Tese em Engenharia Elétrica) - Escola Politécnica da Universidade de São Paulo, São Paulo.

Fonseca, M. d. (9 de outubro de 2002). VI Seminário de Automação de Processos, Associação Brasileira de Metalurgia e Materiais. *Comunicação OPC - Uma abordagem prática*, p. 12.

Gonçalves, R. N. (2012). *Desenvolvimento de servidores OPC DA, OPC UA e wrappers para aplicação em automação.* Dissertação (Dissertação em Engenharia Elétrica) - Universidade Federal de Itajubá - UFMG, Itajubá.

Lugli, A. B., & Santos, M. M. (2009). *Sistemas Fieldbus para Automação Industrial. Devicenet, CANOpen, SDS e Ethernet* (1ª ed.). São Paulo: Érica.

Lugli, A. B., Souza, J. E., Pessoa, L. d., Rodrigues, R. L., & Tarifa, T. H. (Fevereiro de 2017). *Redes Ethernet Industriais e Aplicações.* Instituto Nacional de Telecomunicações - INATEL, Santa Rita do Sapucaí, Minas Gerais, Brasil.

Mackay, S., Wright, E., Reynders, D., & Park, J. (2004). *Practical Industrial Data Networks: Design, Installation and Troubleshooting* (1ª ed.). Burlington: Elsevier.

Mahnke, W., Leitner, S.-H., & Damm, M. (2009). *OPC Unified Architecture* (1ª ed.). Alemanha: Springer.

Metcalf, E. E. (2003). *Waste Enineering: teatment, diposal and reuse.* McGraw Hill: 4ª Ed.

Pressman, R. S. (2010). *Engenharia de software* (6ª ed.). Porto Alegre: MCGrawHill.

Rocha, D. J. (2013). *Sistemas de supervisão e controles de autómatos: soluções baseadas em OPC e IEC 60870-5-104.* Instituto Superior de Engenharia do Porto.

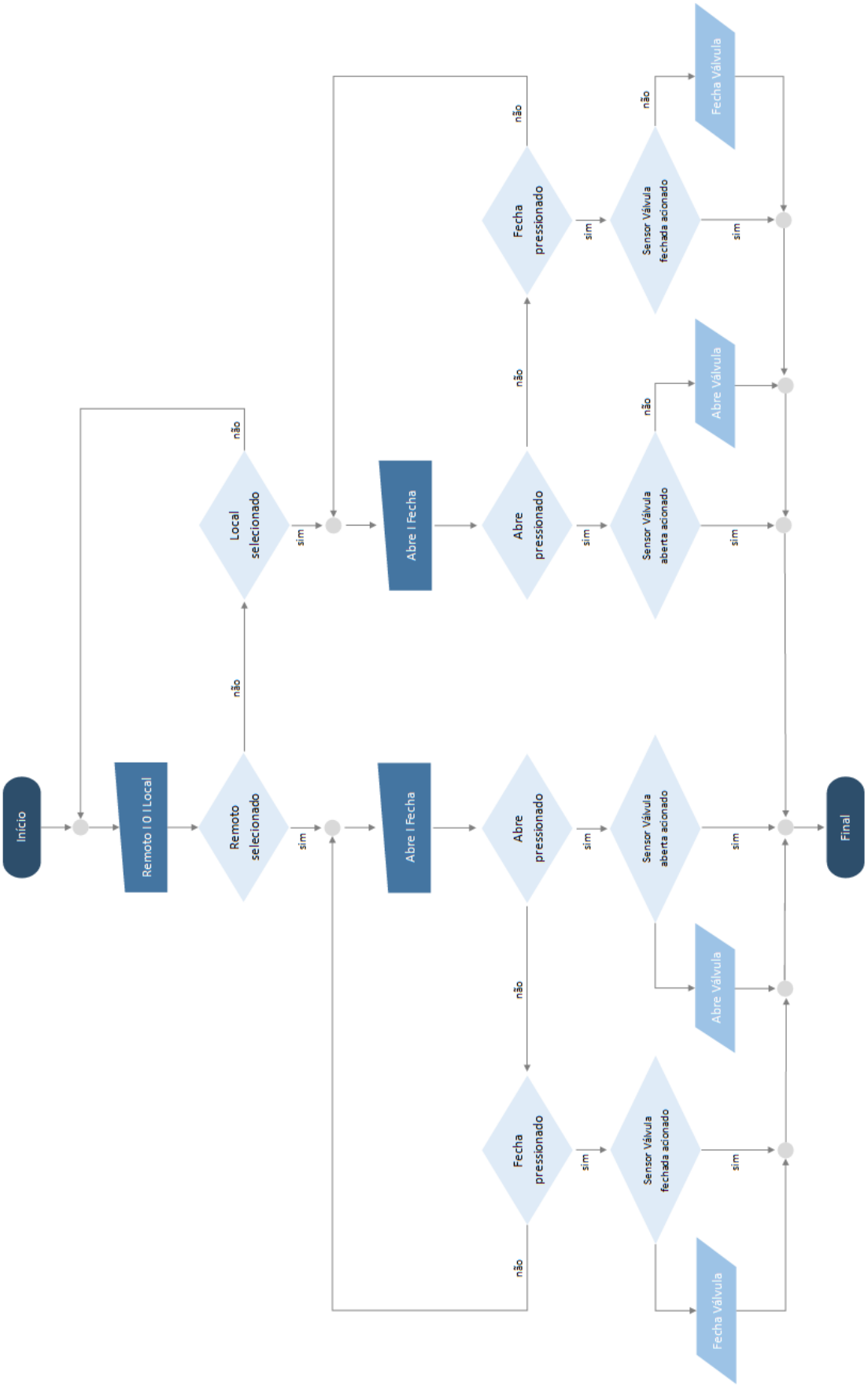
Schwab, K. (2017). *The Fourth Industrial Revolution* (First ed.). Nova York: Crown Business.

Thans, F. C. (2008). *Controle operacional de reator em bateladas sequenciais (RBS): ajuste na concentração de oxigênio dissolvido visando a remoção de nutrientes.* Universidade Federal de Santa Catarina, Florianópolis.

Viegas, V. M. (2012). *Plataformas de Interoperabilidade para sistemas de instrumentação, medida e controle.* Universidade Técnica de Lisboa.

6. Anexos

6.1 ANEXO I - Fluxograma de modelagem das válvulas motorizadas



6.2 ANEXO II - Fluxograma de modelagem das motobombas