UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
UNIDADE ACADÊMICA DE GRADUAÇÃO
CURSO DE ENGENHARIA ELÉTRICA

GABRIEL TEIXEIRA ZANCHIN

DESENVOLVIMENTO DE UM GATEWAY PARA AUTOMAÇÃO INDUSTRIAL

São Leopoldo
2021

GABRIEL TEIXEIRA ZANCHIN

DESENVOLVIMENTO DE UM GATEWAY PARA AUTOMAÇÃO INDUSTRIAL

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de Bacharel em Engenharia Elétrica, pelo Curso de Engenharia Elétrica da Universidade do Vale do Rio dos Sinos - UNISINOS

Orientador: Prof Samuel Lessinger

São Leopoldo

2021

RESUMO

O avanço das tecnologias de redes de comunicação nos últimos anos possibilitou o surgimento de um grande número de protocolos e equipamentos novos, que demandam o uso de hardwares intermediários para integração dos nós de uma planta. A necessidade de adequar redes ao cenário moderno, com o estabelecimento da Internet Industrial e Internet das Coisas alavancou o estudo de soluções que se proponham a solucionar esse problema. O objetivo desse trabalho foi desenvolver um *gateway* para automação industrial, capaz de realizar a leitura e escrita de valores de uma planta e disponibilizá-los para acesso de usuários através de um *software* cliente. Um caso particular desse tipo de aplicação é a existência de uma planta de automação didática no laboratório de Controle da UNISINOS e que é pouco utilizada pela comunidade acadêmica devido à dificuldade dos alunos em acessar os dados de operação da mesma com seus computadores, visto que o *software* comercializado pelo fabricante é disponibilizado aos clientes na forma de licenças pagas. Para comprovar a viabilidade do projeto, foi realizada uma pesquisa dos conceitos envolvendo gateway, plantas de controle e redes de comunicação, e que é apresentada neste trabalho. O funcionamento do hardware será descrito com base nos testes de troca de mensagens realizados ao término da pesquisa, demonstrando sua eficiência e utilidade.

Palavras-chave: Controle. Automação. Gateway. Redes. Protocolos.

LISTA DE FIGURAS

LISTA DE SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
A&E	*Alarms & Events* (Alarmes e Eventos)
CLP	Controlador Lógico Programável
COM	*Component Object Model* (Modelo de Objeto Componente)
CPU	*Central Processing Unit* (Unidade Central de Processamento)
DA	*Data Access* (Acesso de Dados)
DCOM	*Distributed Component Object Model* (Modelo de Objeto Componente Distribuído)
DP	*Decentralised Peripherals* (Periféricos Decentralizados)
E/S	Entradas e Saídas
EEPROM	*Electrically-Erasable Programmable Read-Only Memory* (Memória Somente Leitura Programável Apagável Eletricamente)
FSK	*Frequency Shift Keying* (Modulação por Chaveamento de Frequência)
GPIO	*General Purpose Input/Output* (Entrada e Saída de Uso Geral)
HART	*Highway Addressable Remote Transducer* (Via de Trandutor Remoto Endereçável)
HDA	*Historical Data Access* (Histórico de Acesso aos Dados)
HDMI	*High-Definition Multimedia Interface* (Interface Multimídia de Alta Definição)
HTTP	*Hypertext Transfer Protocol* (Protocolo de Transferência de Hipertexto)
IEC	*International Electrotechnical Commision* (Comissão Eletrotécnica Internacional)
IHM	Interface Homem Máquina
IIoT	*Industrial Internet of Things* (Internet Industrial das Coisas)
IoT	*Internet of Things* (Internet das Coisas)
IP	*Internet Protocol* (Protocolo Internet)
ISO	*International Organization for Standardization* (Organização Internacional de Padronização)
JSON	*JavaScript Object Notation* (Notação de Objetos JavaScript)
LED	*Light Emitting Diode* (Diodo Emissor de Luz)
LDR	*Light Dependent Resistor* (Resistor Dependente de Luz)

MQTT	*Message Queuing Telemetry Transport* (Enfileiramento de Mensagens de Transporte de Telemetria)
OPC	*Object Linking and Embedding for Process Control* (Vinculação e Incorporação de Objetos para Controle de Processo)
OS	*Operational System* (Sistema Operacional)
OSI	Open Systems Interconnection (Interconexão de Sistemas Abertos)
PA	*Process Automation* (Automação de Processos)
RAM	*Random Acess Memory* (Memória de Acesso Aleatório)
RTDs	*Resistance Temperature Detector* (Detector de Temperatura de Resistência)
RTU	*Remote Terminal Unit* (Unidade Remota Terminal)
SD	*Secure Digital Card* (Cartão de Memória e Armazenamento)
SOAP	*Simple Object Access Protocol* (Protocolo de Acesso de Objetos Simples)
TCP	*Transmission Control Protocol* (Protocolo de Controle de Transmissão)
UA	*Unified Architecture* (Arquitetura Unificada)
USB	*Universal Serial Bus* (Barramento Serial Universal)
XML	*Extensible Markup Language* (Linguagem de Marcação Extensível)

SUMÁRIO

1 INTRODUÇÃO

A Comunicação entre elementos de rede é um tópico da Engenharia que vem se desenvolvendo nos últimos anos, apresentando constantemente inovações. As diferentes necessidades decorrentes de cada contexto demandaram que as redes de comunicação desenvolvessem uma grande variedade de tecnologias e padrões, de maneira a atender setores distintos da sociedade, como nas telecomunicações, transmissão de energia e redes industriais. Dentro do contexto de controle de processos e automação na indústria, o grande número de equipamentos e dispositivos, de diferentes fabricantes e operando sob diferentes protocolos, demanda a utilização de dispositivos intermediários que assegurem a comunicação correta e segura entre os diversos pontos de uma planta, dispositivos estes que são conhecidos como gateways.

Esse trabalho aborda o desenvolvimento de um gateway para estabelecer a comunicação no cenário de automação industrial. Será utilizado como modelo para esse trabalho simuladores e uma planta de automação de processos SMAR PD3P, disponível no Laboratório de Controle e Automação da UNISINOS.

O objetivo geral é o desenvolvimento de um hardware que funcione como gateway OPC UA (*Open Platform Communications Unified Architecture*, ou Plataforma Aberta de Comunicação com Arquitetura Unificada) para gerenciar o acesso de uma planta por alunos e professores, de modo que o recurso possa ser utilizado por um número maior de usuários e não fique limitado ao uso através do software proprietário do fabricante. Atualmente, no caso da planta SMAR PD3P, somente um computador do laboratório de Controle e Automação possui licença para instalação dos softwares de supervisão dos dados da planta. Um fato limitador deste cenário é o controlador do equipamento possuir tecnologia OPC DA, um protocolo que utiliza tecnologias proprietárias da Microsoft e que portanto opera dependente de sistemas operacionais Windows. A utilização de um dispositivo semelhante ao descrito no trabalho contorna estas limitações.

A planta utilizada como modelo para o estudo possui recursos de integração com supervisão do protocolo OPC DA e OPC XML DA, portanto é possível gerenciar esses dados a partir de tecnologias mais recentes como o OPC UA e disponibilizá-los para o acesso de forma mais eficiente.

Uma ênfase será dada aos protocolos usados nos equipamentos presentes na planta didática estudada – Profibus DP-PA e HART -, no entanto a sua aplicação pode ser estendida a outras situações envolvendo controladores que possuam a tecnologia OPC UA já integrada.

Inicialmente, o problema será delimitado no acesso via computador pessoal de valores registrados em controladores OPC e dispositivos conectados fisicamente ao gateway. Devido a indisponibilidade do espaço físico do Laboratório de Controle e Automação da Unisinos durante o semestre e a realização deste trabalho, foram utilizados *softwares* de simuladores de servidores OPC para validação dos métodos empregados na solução do problema delimitado.

O trabalho está dividido em cinco capítulos. O capítulo um trata de fornecer uma introdução, buscando contextualizar o assunto e apresentar o objetivo. O capítulo dois apresenta a fundamentação de conceitos utilizados para o desenvolvimento do projeto, bem como uma discussão do atual estado da arte em se tratando gateways para redes inteligentes. O capítulo três descreve como foram organizados e executados os passos para a implementação do gateway. O capítulo quatro traz uma análise dos resultados obtidos durante o processo de escrita do trabalho e realização dos testes. Por fim, no capítulo cinco é feita uma avaliação final do hardware desenvolvido, seus recursos e limitações, bem como possíveis melhorias para sua implementação.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão apresentados na forma de revisão conceitos básicos referentes ao assunto do trabalho, e que foram utilizados como base para o desenvolvimento do estudo. Inicialmente, serão apresentadas as definições de Sistemas de Controle e Automação, e sua relação com a planta. Após, são descritas as características do modelo de planta disponibilizado pela UNISINOS e que é utilizada neste trabalho como objeto de interesse. Na sequência, é dado um panorama sobre protocolos de comunicação, o modelo OSI e os *softwares* de simulação que auxiliaram nos testes de implementação. Depois disso, é dada uma caracterização dos protocolos usados pelos dispositivos e controladores da planta, e as definições de um gateway. Após, é apresentado o hardware que se deseja utilizar para como plataforma para o trabalho e dada a justificativa para sua escolha. Por fim, é dado um breve contexto sobre o atual estado da arte em se tratando do uso de gateways inteligentes em aplicações envolvendo Redes Industriais e Internet das Coisas (IoT), com a apresentação de trabalhos correlatos.

2.1 Sistemas de Controle e Automação

O desenvolvimento e aperfeiçoamento das técnicas de controle moderno permitiram o surgimento de novas tecnologias dentro do contexto da terceira revolução industrial, influenciando em áreas como controle de processos, eletrônica, robótica, telecomunicações, etc. (Nise, 2017).

De acordo com a definição de Nise, um sistema de controle é caracterizado por subsistemas e processos projetados com a finalidade de, a partir de uma determinada entrada, se obter uma saída esperada com um desempenho também esperado (Nise, 2017).

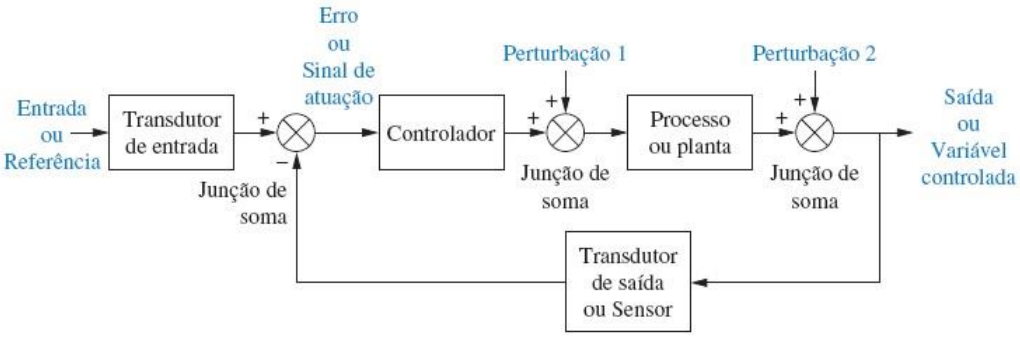
As principais vantagens para o uso de sistemas de controle podem recair sobre quatro tópicos principais: amplificação/ganho de potência, controle remoto, conveniência na hora de se determinar o tipo de sinal na entrada de um sistema, e o ajuste de ruídos ou distúrbios. Sobre sua topologia e modos de operação, podem ser caracterizados como controle de malha aberta ou de malha fechada (Nise, 2017).

2.1.1 Controle a malha aberta e malha fechada

Sistemas de controle a malha aberta são sistemas cuja entrada depende unicamente de um valor referencial. Esse tipo de controle não realiza ajustes de eventuais erros ou perturbações do sistema (Nise, 2017).

Sistemas de controle a malha fechada tem a sua entrada associada tanto a um valor referencial como também do sinal de saída deste sistema. A diferença entre a entrada de referência e o sinal de saída é conhecida como Erro. No controle de malha fechada, o Erro é usado para ajustar o sinal de entrada de modo a se compensar esses distúrbios, e se obter a saída desejada. A figura 1 apresenta o diagrama de blocos de um sistema em malha fechada (Nise, 2017).

Figura 1 - Diagrama de Blocos de um Sistema em Malha Fechada



Fonte: Nise (2017, p.7)

2.2 Planta de Controle SMAR

A planta de automação didática PD3P da Smar é um equipamento capaz de simular e reproduzir diversos processos de automação industrial e permite visualização da interação de variáveis de malhas de controle. Ela é uma solução completa composta por atuadores - bomba hidráulica, posicionadores de válvulas (reguladoras de fluxo de água) -, sensores de temperatura, pressão e vazão, e um controlador lógico programável (CLP) capaz de processar os sinais digitais. A figura 2 apresenta uma vista frontal da planta disponível no laboratório de Controle e Automação da UNISINOS (Smar PD3P, 2020).

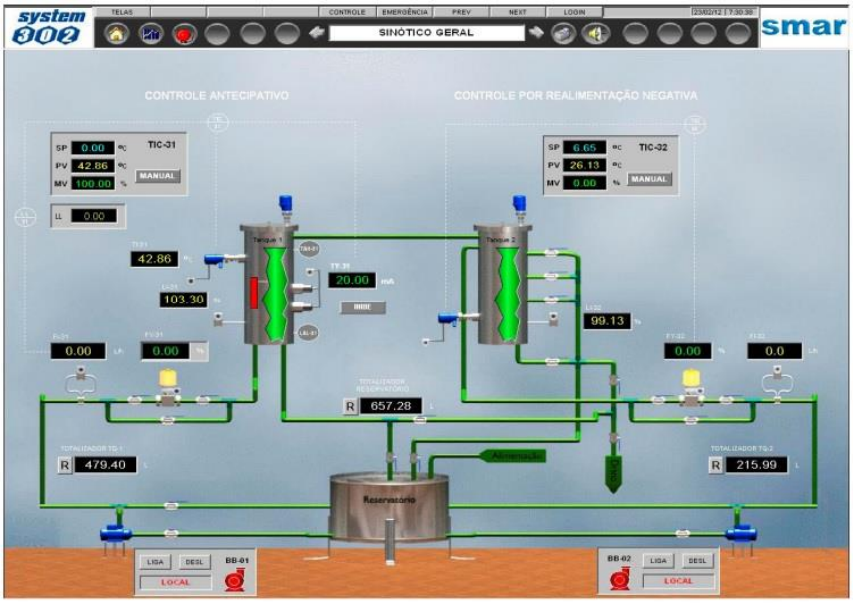
Figura 2 - Planta PD3P

Fonte: Adaptado de SMAR PD3P (2020)

O equipamento utiliza instrumentos de campo e aplicativos de software semelhantes aos utilizados em larga escala em redes de campo da indústria, permitindo que a PD3P seja usada para fins didáticos, seja em cursos de engenharia elétrica, eletrônica, controle e automação, seja na própria indústria para o treinamento de funcionários que atuam diretamente na operação de plantas similares (Smar PD3P, 2020).

Também faz parte da planta o software proprietário System302, um programa desenvolvido pela fabricante comercializado com licenças para realizar a interface visual dos processos trabalhados pela PD3P com o usuário e que é utilizado em uma estação de trabalho remota. Através do System302 é possível monitorar o status do sistema, pois ele efetua coleta de informações disponíveis dos equipamentos e apresenta os dados através de telas gráficas e animações . É também através do System302 que é realizada a configuração de operação da Planta didática e seus instrumentos, onde o usuário determina valores internos que serão usados pelos dispositivos e o modo de operação que será realizado. A figura 3 demonstra um exemplo de tela do software do fabricante e sua interface com o usuário (Smar PD3P, 2020).

Figura 3 - Interface visual System302



Fonte: Adaptado de SMAR PD3P (2020)

2.2.1 Transdutores

Na engenharia, transdutores são definidos por diversos autores e bibliografias de maneiras diferentes, havendo instituições responsáveis por buscar uma padronização no emprego de termos técnicos dentro da área de instrumentação. De acordo com Balbinot, o transdutor é definido como sendo um dispositivo que realiza a transformação de um sinal de uma forma física para um sinal correspondente de outra forma física, isto é, age como um sensor de determinada grandeza física – a palavra sensor é inclusive utilizada como sinônimo para transdutor dentro da área. Existem inúmeras classificações de sensores quanto a seu tipo de construção, modo de operação e finalidades (Balbinot, 2010).

A planta PD3P possui sensores de temperatura vazão e nível de água em um reservatório, por exemplo, que podem ser diretamente monitorados em estação remota com o aplicativo System302.

2.2.1.1 Transmissor de Pressão LD303

O transmissor de pressão LD303 da SMAR é um instrumento para medidas de pressão diferencial, absoluta e manométrica, nível e vazão, sendo da primeira geração

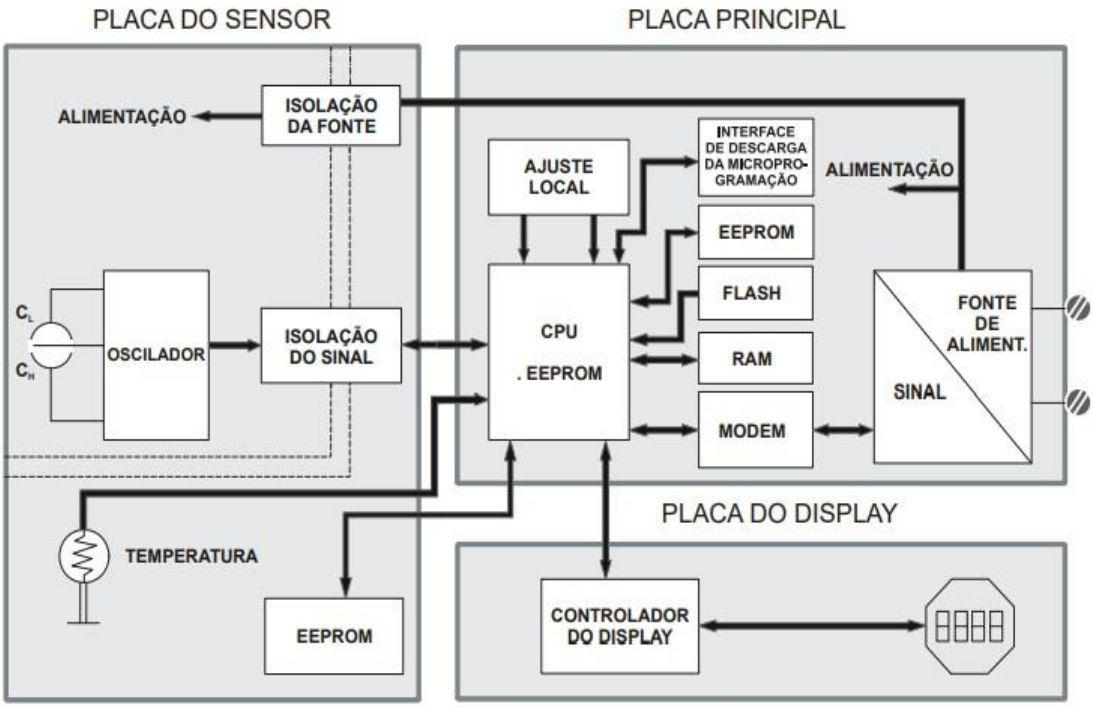
de equipamentos Profibus PA. A figura 4 apresenta o modelo do transmissor de pressão presente na PD3-P. Ele possui em seu circuito um sensor com finalidade de detectar variações de temperatura no ambiente, o que diminui a imprecisão na leitura de dados devido a fatores externos. O LD303 também possui um display para aferição de valores direto na planta (SMAR LD303, 2020).

Figura 4 - Transmissor de Pressão

Fonte: Adaptado de SMAR LD303 (2020)

Para realizar as medições, o LD303 usa um sensor do tipo célula capacitiva, com dois capacitores de capacitâncias variáveis sensíveis a pressão diferencial aplicada. A figura 5 apresenta o funcionamento do circuito em blocos. Os dados recolhidos são processados por um microcontrolador, que fica responsável pelo gerenciamento e monitoramento das medidas e sua comunicação. Internamente, também possui uma memória EEPROM que armazena dados que precisam ser salvos, como por exemplo perfis de calibração e operação. Para comunicação externa, os sinais de controle são modulados por um modem e enviados via um par de fios (SMAR LD303, 2020).

Figura 5 - Diagrama de Blocos LD303



Fonte: Adaptado de SMAR LD303 (2020)

2.2.1.2 Transmissor de Temperatura TT303

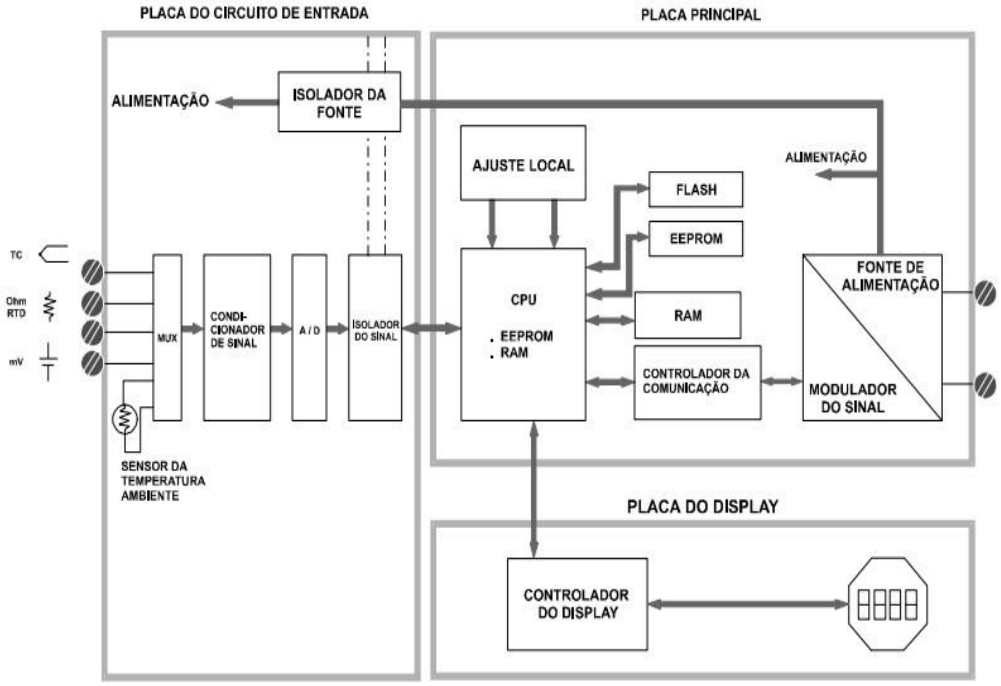
Assim como o LD303, o transmissor de temperatura TT303 também é um equipamento da primeira geração de instrumentos Profibus PA. Ele usa para modulação do sinal o modo de tensão 31,25 kbit/s e um par de fios como meio físico, sendo recomendado a utilização de cabos do tipo par trançado. No mesmo barramento devem ser usados equipamentos respeitando o mesmo tipo de modulação, podendo ser utilizados diversos instrumentos de redes de campo. Sua alimentação é realizada via barramento, devendo estar na faixa de 9 a 32 V_{DC} . A figura 6 traz uma imagem do TT303, que pode funcionar com sinais de termopares e termoresistências (RTDs), desde que estejam operando dentro da faixa aceitável (de 50 a 500 mV para geradores de tensão, e 0 a 2 kΩ para sensores resistivos) (SMAR LT303, 2020).

Figura 6 - Transmissor de Temperatura

Fonte: Adaptado de SMAR TT303 (2020)

A figura 7 apresenta o funcionamento do circuito em blocos. O bloco do circuito de entrada realiza a aquisição de leituras dos sensores e condiciona o sinal para ser enviado ao microprocessador. No bloco principal, os dados de campo são processados por uma CPU, que fica responsável pelo gerenciamento e monitoramento das medições e sua comunicação. A CPU também possui uma memória interna não volátil, onde se pode armazenar dados relevantes, como perfis de calibração. O TT303 também conta com um display para aferição de valores direto na planta.

Figura 7 - Diagrama de Blocos LT303



Fonte: Adaptado de SMAR LT303 (2020)

2.2.2 Atuadores

Os atuadores são os dispositivos ou equipamentos responsáveis por traduzirem em movimento ou algum outro tipo de ação os comandos enviados pelo sistema – geralmente sinais elétricos. Podem ser vistos como transdutores de saída, já que convertem energia elétrica em outra forma de energia, ao contrário dos sensores (transdutores de entrada) (Balbinot, 2010).

A planta PD3P possui os atuadores: bomba hidráulica, posicionadores de válvula, resistências aquecedoras, etc., que podem ser diretamente monitorados em estação remota.

2.2.2.1 Posicionador de válvula FY303

Os posicionadores de válvula FY303 da Smar são baseados no protocolo Profibus-PA e fornecem controle linear para a saída de pressão exata das válvulas, baseado em comandos recebidos por uma unidade controladora. A figura 8 ilustra o posicionador encontrado na PD3-P (SMAR FY303, 2020).

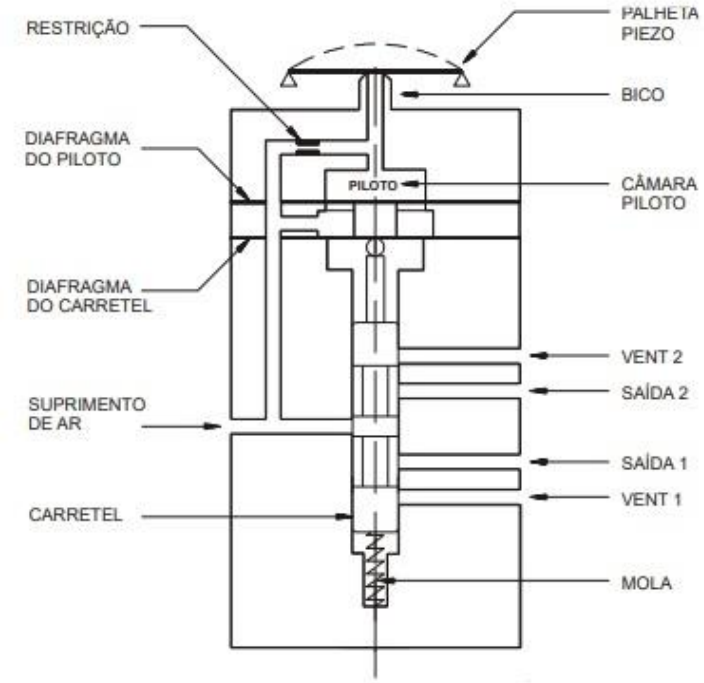
Figura 8 - Posicionador de Válvula

Fonte: Adaptado de SMAR FY303 (2020)

Seu funcionamento de controle é baseado no esquema demonstrado na figura 9. O circuito recebe um valor de referência enviado pela CPU e um sinal de realimentação advindo do sensor de efeito Hall. A palheta piezoelétrica no topo do

esquema é deslocada com a aplicação de tensão pelo driver de controle. Com a obstrução do fluxo de ar, a pressão na câmara piloto se altera, transferindo força para o diafragma da câmara piloto. Essas alterações de pressão provocam o deslocamento do carretel, levando a válvula para cima e para baixo e alterando a pressão de saída das saídas 1 e 2 até que o equilíbrio ocorra novamente (SMAR FY303, 2020).

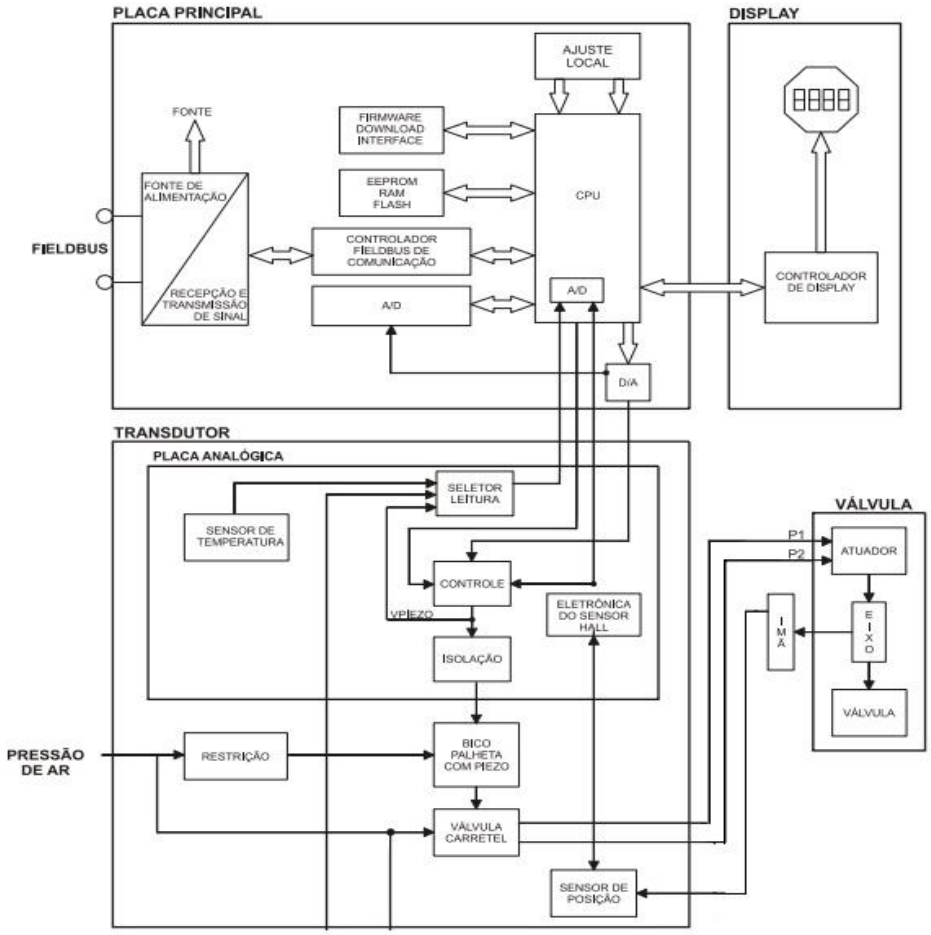
Figura 9 - Esquema de Funcionamento do Transdutor Pneumático



Fonte: Adaptado de SMAR FY303 (2020)

A figura 10 ilustra os blocos de funcionamento do equipamento. A placa principal funciona analogamente aos transmissores de pressão e temperatura. Recebe os sinais analógicos do transdutor pneumático e converte para digital, sendo processado pela CPU. Uma memória EEPROM integrada pode armazenar dados configurados do usuário (como backup), e também possui saída para envio e recebimento de sinais com outros instrumentos de redes de campo. A FY303 também conta com display para visualização das medições (SMAR FY303, 2020).

Figura 10 - Diagrama de Blocos FY303



Fonte: Adaptado de SMAR FY303 (2020)

2.2.3 Plataforma de Controle e Automação de Processos DFI302

O DFI302 é uma plataforma modular, flexível e multifunção. O controlador possui uma alta capacidade de processamento, que possibilita a comunicação e aquisição de dados dos processos, e interoperabilidade com equipamentos do protocolo FOUNDATION Fieldbus e PROFIBUS, além de possuir recursos nativos de comunicação OPC UA (SMAR DFI302, 2020).

O DFI302 é usado para a conexão da planta com o gateway, podendo ser utilizado uma gama de *link devices* compatíveis disponíveis no mercado ou uma instalação personalizada. O controlador DFI302 e seus módulos auxiliares são apresentados na figura 11 (SMAR DFI302, 2020).

Figura 11 - Plataforma DFI302

Fonte: Adaptado de SMAR FY303 (2020)

2.3 Protocolos de comunicação

Protocolo de comunicação é um conjunto de regras estabelecido para que possa existir a troca de informações entre dispositivos. Esse conjunto de regras pode apresentar variações de um protocolo para outro, no entanto sempre estabelece três conceitos fundamentais, que são sintaxe, semântica e timing. A sintaxe diz respeito ao formato dos dados que serão trocados (quantidade de bits), e que costuma ser exemplificada através do *dataframe*. A semântica trata do significado dos bits, realizando uma interpretação de cada trecho – leitura, escrita, etc. Por fim, timing se refere a quando os bits devem ser enviados ou recebidos, e também com que velocidade se dará a conexão (Forouzan, 2010).

Os protocolos de comunicação podem ser classificados como abertos ou fechados. São considerados protocolos abertos aqueles que podem ser utilizados por qualquer usuário ou fabricante de *hardware/software* na implementação de seu produto. Esses protocolos são largamente divulgados em meios públicos e possuem suas normas técnicas acessíveis a qualquer um. Exemplos: PROFIBUS, HART. São considerados protocolos fechados aqueles que não são disponibilizados ao usuário geral e nem a outros fabricantes de hardware que não sejam o proprietário (Forouzan, 2010).

2.3.1 Modelo OSI

O modelo OSI (*Open Systems Interconnection*) é um modelo de arquitetura de sistemas de comunicação criado com o objetivo de facilitar as regras de padronização

para criação de redes industriais, uma vez que a vasta quantidade de equipamentos existentes no mercado dificultava a comunicação entre dispositivos de fabricantes diferentes em uma mesma rede. Esse modelo é organizado em uma divisão de sete camadas distintas porém relacionadas entre si, conforme mostrado na figura 12 (Forouzan, 2010).

Figura 12 - Camadas do Modelo OSI

Fonte: Forouzan (2010, p. 30).

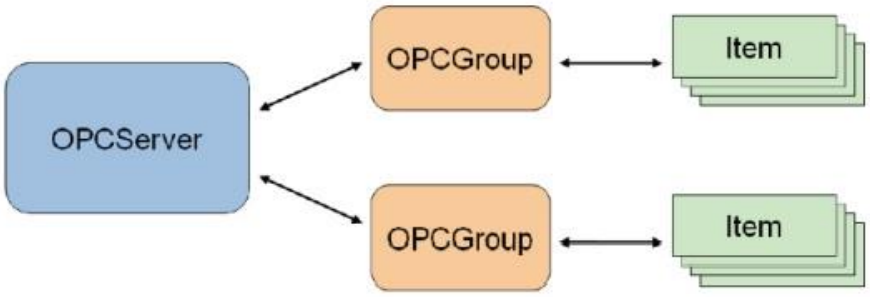
2.3.2 OPC DA

O protocolo OPC, originalmente *OLE for Process Control*, foi criado em 1996 por um grupo de companhias com o objetivo de resolver os problemas relacionados a intercomunicação de dispositivos instalados em níveis diferentes de uma rede de automação. Atualmente, o padrão OPC segue sendo disponibilizado pela OPC Foundation, uma organização sem fins lucrativos responsável por manter a padronização de dispositivos compatíveis. Para a troca de informações, o protocolo utiliza uma abordagem Cliente-Servidor, onde uma máquina Servidor centraliza os dados dos processos e os disponibiliza para o acesso de Clientes OPC que desejarem realizar consultas (Mahnke et. al., 2009).

À época, e de acordo com os requisitos dentro do contexto de aplicações industriais, foram desenvolvidas três principais especificações para o protocolo: Acesso de Dados (*Data Access,* ou *"DA"*), Alarme & Eventos (*"A&E*") e Acesso a Histórico de Dados (ou *"HDA"*, do inglês *Historical Data Access*). (Mahnke et. al., 2009).

A interface OPC Data Access permite a leitura, escrita e monitoramento de variáveis de interesse em um processo, e sua utilização se dá principalmente na aquisição de dados em tempo real de CLPs e outros dispositivos de controle ligados a sistemas IHM. O funcionamento do acesso de dados opera a partir de um cliente OPC DA, que seleciona a variável que deseja consultar. É então instanciado um objeto OPCServer, que oferece métodos de navegação ao cliente para que este possa acessar os dados de interesse e suas características, como o tipo de dado e as suas permissões de acesso. A figura 13 ilustra o instanciamento de objetos por um cliente OPC no acesso de dados, com as variáveis no servidor sendo representadas pelos Itens (Mahnke et. al., 2009).

Figura 13 - OPC UA na pirâmide de Automação



Fonte: Mahnke et. al. (2009)

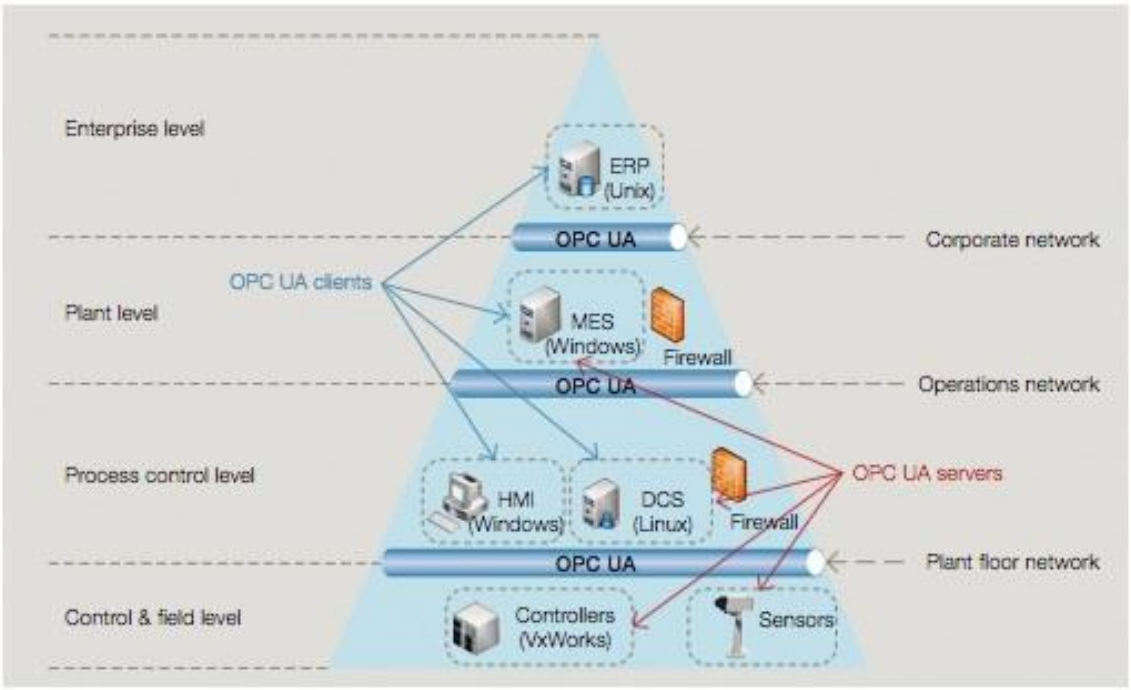
O OPC DA utilizava inicialmente para interface a tecnologia COM e DCOM, proprietárias da Microsoft, e que representavam uma desvantagem em sua utilização, uma vez que a tornava dependente de sistemas operacionais baseados em Windows. Posteriormente, foi desenvolvido o OPC XML-DA, substituindo a COM/DCOM pelo HTTP/SOAP e tecnologias Web Service. Esse modelo foi utilizado principalmente em sistemas embarcados e outros sistemas operacionais que não o Windows, mas devido as restrições que precisou implementar em comparação ao OPC DA e ao alto nível

de recursos exigidos para o funcionamento limitado fez com que o protocolo não atingisse o sucesso que era esperado (Mahnke et. al., 2009).

2.3.3 OPC UA

A partir da popularização do OPC DA Clássico e do surgimento de novas demandas, o protocolo sofreu alterações para que passasse a ser um padrão na indústria, removendo a necessidade de se utilizar tecnologias proprietárias, e se tornando uma arquitetura unificada – recebendo a descrição UA (Unified Architecture). Hoje, o OPC UA (Open Platform Communications Unified Architecture) é um protocolo de comunicação no qual diversos tipos de dispositivos e sistemas podem ser configurados para que se comuniquem, realizando troca de mensagens no formato cliente – servidor. A figura 14 ilustra as camadas da pirâmide de automação na qual o protocolo OPC UA pode integrar. Para que exista comunicação, o OPC UA define perfis nos quais o Servidor pode operar e solicitar conformidade com os Clientes, dinamicamente se adequando ao perfil a depender da necessidade (OPC FOUNDATION, 2020).

Figura 14 - OPC UA na pirâmide de Automação

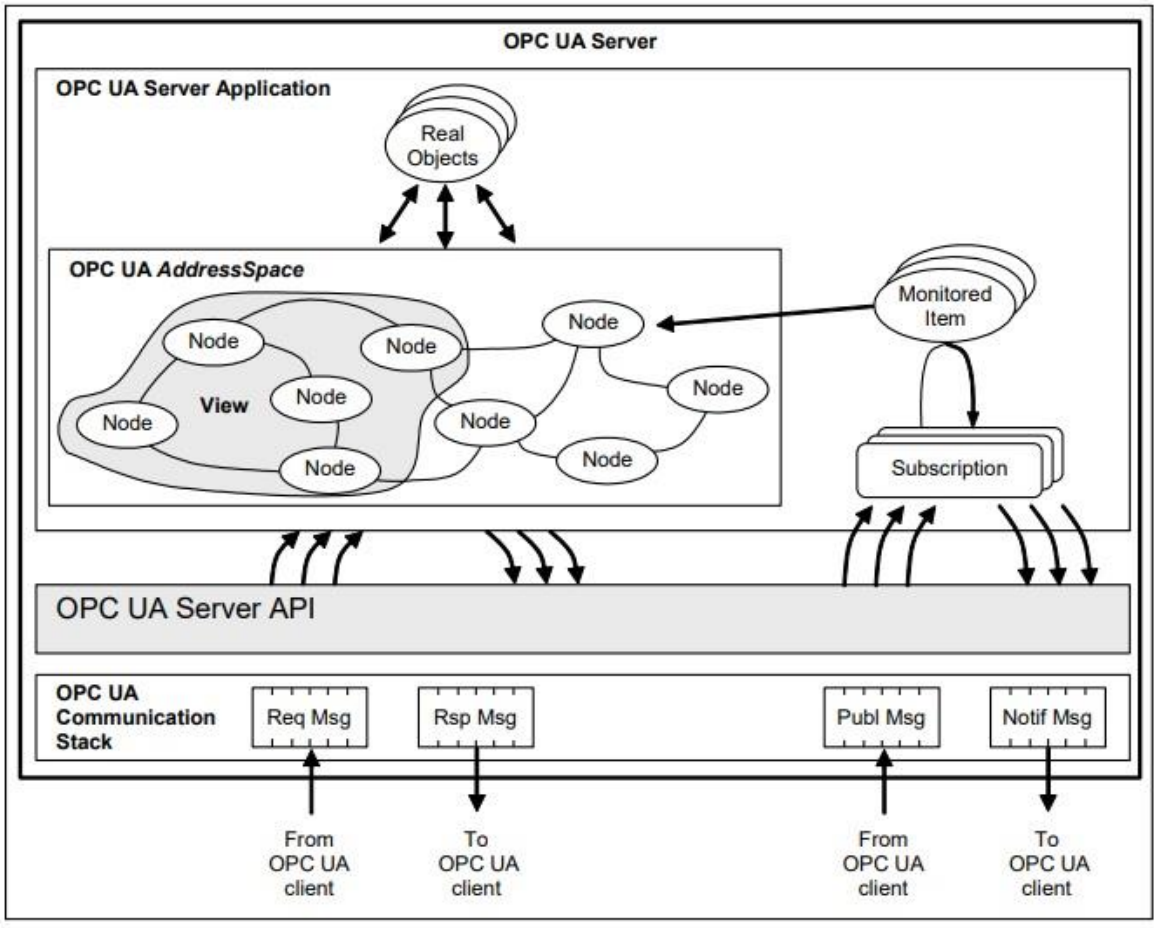


Fonte: Adaptado de www.embarcados.com.br

A arquitetura OPC UA define a existência de máquinas Servidores e Clientes trabalhando em parceria, com cada sistema podendo receber diversos Servidores e Clientes que interagem concomitantemente. O Servidor OPC UA modela o *endpoint* das interações entre cliente e servidor (OPC FOUNDATION, 2020).

A figura 15 apresenta uma ilustração dos elementos em um Servidor OPC UA e suas relações entre si. No esquema, os Objetos reais (*Real Objects*) representam objetos físicos (por exemplo, um dispositivo de campo) ou de software que podem ser consultados pela aplicação do Servidor. Esses objetos são organizados dentro de uma estrutura conhecida como AddressSpace, que por sua vez é modelada como um conjunto de Nós (Nodes); cada nó representa um objeto real, que pode ser acessado pelos Clientes (OPC FOUNDATION, 2020).

Figura 15 - Arquitetura do Servidor OPC UA

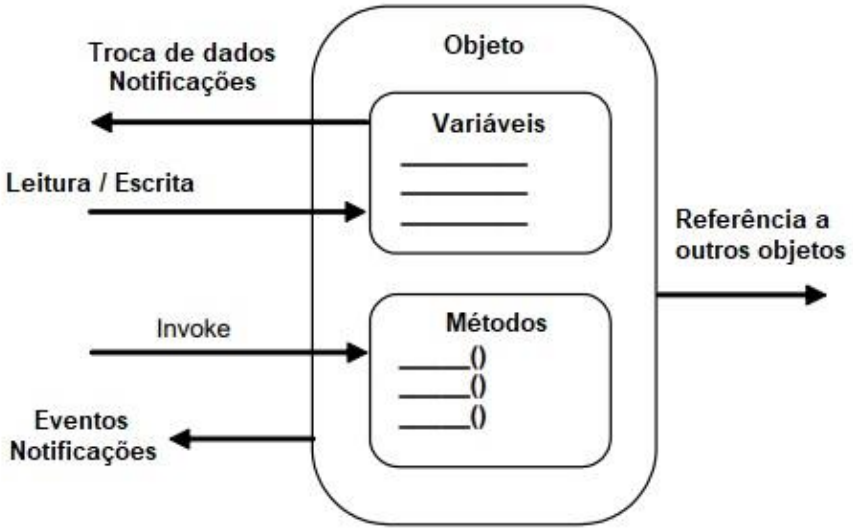


Fonte: OPC UA Specification (2020)

A função principal do AddressSpace é a de padronizar a forma com que servidores representam os Objetos para acesso dos clientes. A figura 16 ilustra o

modelo de Objeto utilizado pelo protocolo, que é descrito na forma de Variáveis e Métodos. As Variáveis representam valores armazenados atribuídos ao Objeto, que podem ser utilizadas para leitura e escrita, enquanto os Métodos são funções que podem retornar ao cliente um resultado. Esse modelo permite a referência entre Objetos, o que faz com que o Nós do AddressSpace possam ser organizados em hierarquias (OPC FOUNDATION, 2020).

Figura 16 – Modelo de Objeto OPC UA

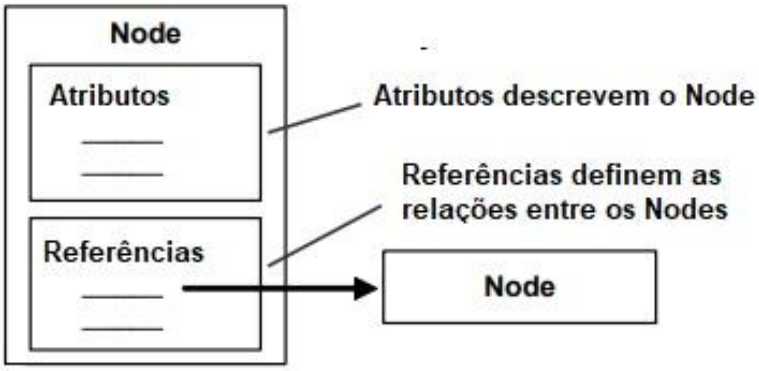


Fonte: Adaptado de OPC UA Specification (2020)

O conjunto de Nós são organizados no AddressSpace conforme o esquema da figura 17, através de Atributos e Referências. Os Atributos fornecem um nome, uma descrição e o tipo de dado que o Nó carrega, que são fornecidos no momento em que o Nó é instanciado no AddressSpace.

As Referências são definidas pelo Servidor, e cada uma é instanciada através de um *ReferenceType Node*. O Nó que contém a Referência é conhecido como *SourceNode*, enquanto o Nó que é referenciado é conhecido como *TargetNode* – que pode estar localizado tanto no mesmo AddressSpace quanto no de outro servidor OPC UA (OPC FOUNDATION, 2020).

Figura 17 - Modelo de Nó OPC UA



Fonte: Adaptado de OPC UA Specification (2020)

As principais vantagens na utilização de uma interface OPC são: autonomia dos fabricantes na manufatura de equipamentos e desenvolvimento de softwares, facilidade para configuração do tipo de dados trocados, escalabilidade, orientação a objetos e permissão para acesso simultâneo as informações guardadas pelo Servidor OPC (UNIFIED AUTOMATION, 2020).

2.3.4 Profibus DP-PA

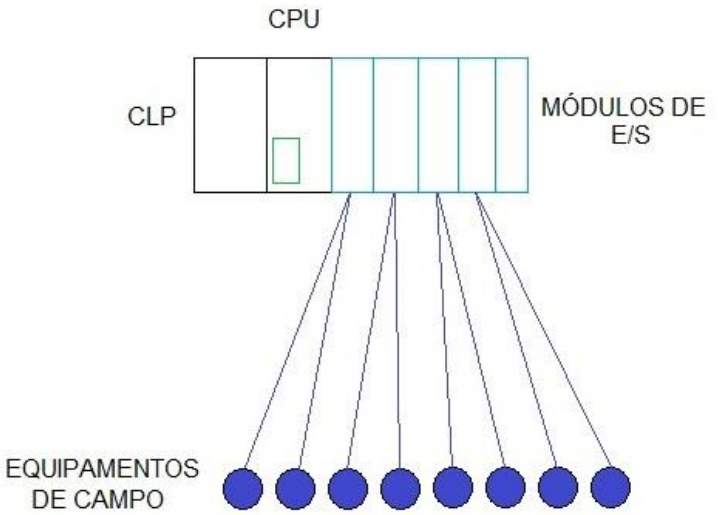
O Profibus é um padrão aberto de redes fieldbus para comunicação digital bidirecional muito utilizado em aplicações em processo e manufatura, e que é definido pelas normas EN 50170 e EN 50254. É baseado no padrão ISO/OSI, atuando nas camadas física, camada de interface de dados e interface do usuário (Lugli; Santos, 2010).

O Profibus DP (*Decentralised Peripherals*) é o protocolo utilizado na operação de sensores e atuadores via uma central de controle descentralizada. Ele utiliza somente as camadas 1 e 2 do modelo OSI, que determinam as características físicas de transmissão e o protocolo utilizado para acesso, respectivamente. Com a utilização deste protocolo, é possível instalar o bloco de entradas e saídas de um CLP (E/S) mais próximo dos instrumentos de campo e economizar os custos de cabeamento, especialmente em cenários onde o chão de fábrica fica a uma grande distância da sala de controle. Com a conexão dos equipamentos de campo no bloco de E/S, a transmissão para o CLP se dá utilizando cabo RS-485. A transmissão digital traz

benefícios de diminuição de interferências no ambiente industrial, tornando a operação mais robusta DP (Lugli; Santos, 2010).

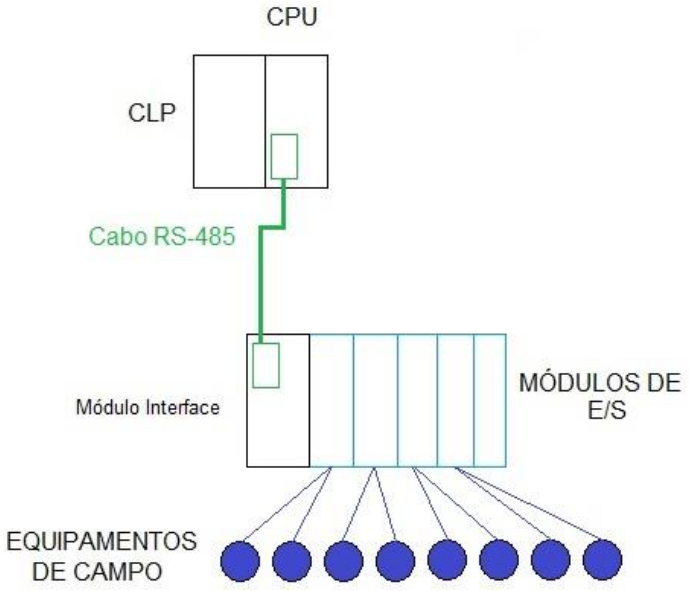
A figura 18 apresenta um modelo de rede industrial com a instalação centralizada em uma sala de controle, enquanto a figura 19 demonstra a topologia da rede em um cenário com Profibus DP.

Figura 18 - Rede Industrial com Instalação Centralizada



Fonte: Elaborado pelo Autor

Figura 19 - Rede Industrial Profibus DP



Fonte: Elaborado pelo Autor

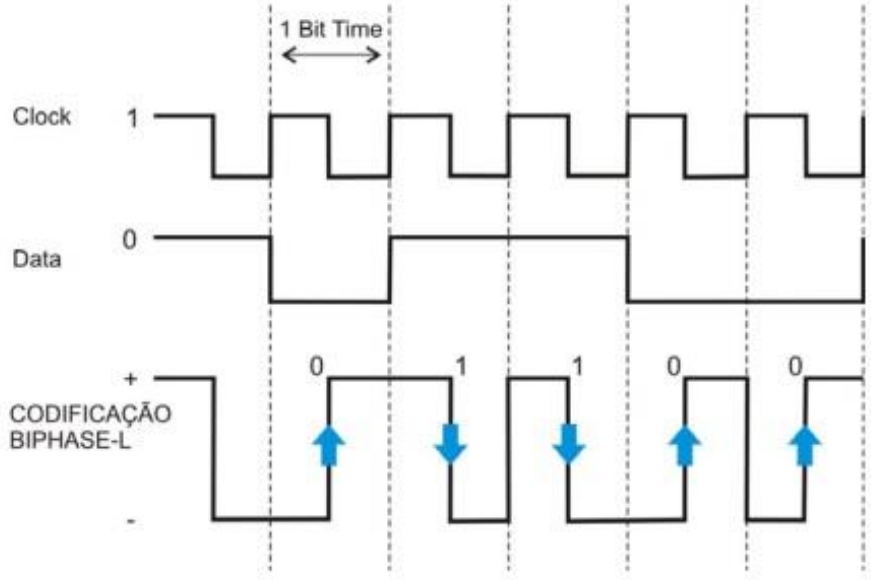
O Profibus PA (*Process Automation*) foi desenvolvido para a indústria para substituição do padrão de transmissão 4-20 mA. O protocolo utiliza cabeamento diferente do Profibus DP, no entanto o formato de dados e pacotes enviados são iguais, permitindo o uso dos dois padrões em conjunto (Lugli; Santos, 2010).

A norma que define o meio físico do protocolo PA é a IEC 61158-2. Algumas características são:

- Transferência com taxa de 31,25 kbit/s;
- Suporte a alimentação dos dispositivos via barramento, com o mesmo par de fios usado na alimentação podendo ser utilizado para transmitir o sinal de controle;
- Comprimento máximo da rede pode alcançar 9,5 km, havendo uso de repetidores;
- Capacidade para até 32 equipamentos compartilhados;
- Topologias barramento, árvore, estrela ou mista (Smar, 2009).

A camada física funciona recebendo mensagens da camada de interface de dados, e realiza a conversão desses sinais para que possa ser transmitido em um barramento típico de redes com equipamentos Profibus. A troca de mensagens pelos equipamentos da série 303 da PD3-P é feita usando a codificação Manchester, apresentado na figura 20 (Smar, 2009).

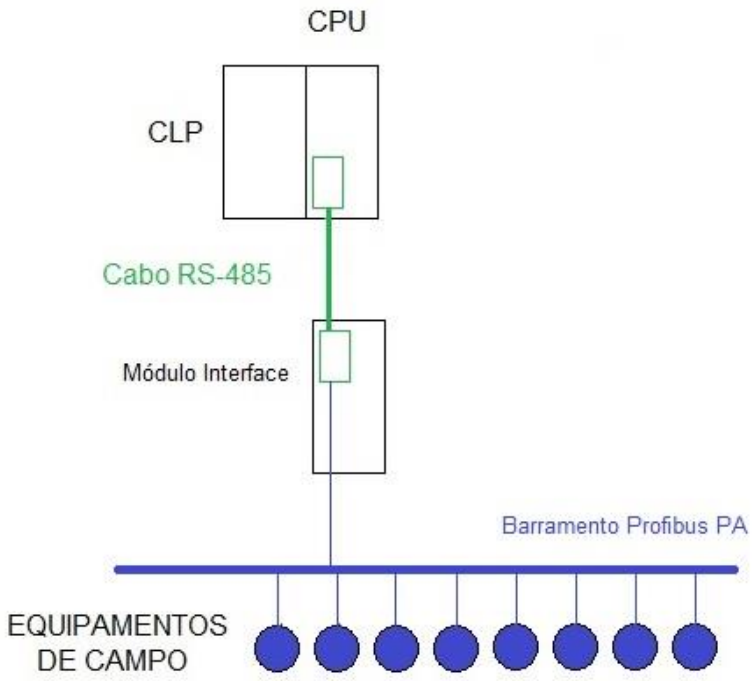
Figura 20 - Codificação Manchester



Fonte: Smar PROFIBUS PA (2009)

A utilização do Profibus PA elimina a necessidade de se utilizar um cabo para cada dispositivo de campo na conexão com os módulos de E/S do controlador, podendo ser ligados a um único barramento. A figura 21 ilustra a topologia da rede em uma instalação com Profibus PA (Smar, 2009).

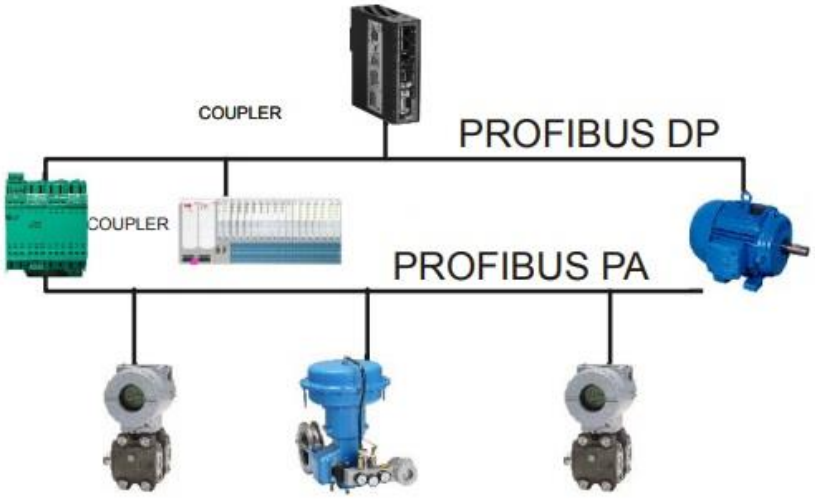
Figura 21 - Rede Industrial Profibus PA



Fonte: Elaborado pelo Autor

Para a adequação do sinal de um barramento Profibus PA para Profibus DP são utilizados acopladores DP-PA. Como meio físico, o Profibus PA pode utilizar tanto RS-485 quanto a fibra ótica, para instalações suscetíveis a perturbações ou que precisem se estender por uma faixa de distância muito grande. A figura 22 demonstra uma rede de equipamentos Profibus DP-PA com acopladores (Smar, 2009).

Figura 22 - Rede Profibus DP-PA



Fonte: Smar PROFIBUS PA (2009)

As principais vantagens na utilização de redes de campo Profibus DP-PA são: transmissão segura de informações, tratamento de erros, alta resolução de medição dos equipamentos, alta rangeabilidade, comunicação de alta velocidade, além de economicamente ser mais enxuto – tanto na instalação quanto manutenção de uma rede do tipo (Smar, 2009).

2.4 Gateway

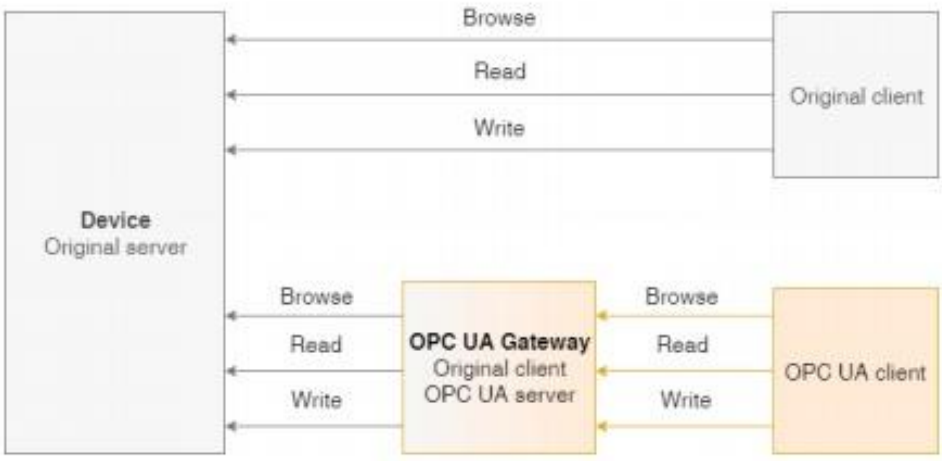
Para que exista a comunicação entre os dispositivos instalados na planta didática – que se utilizam dos protocolos descritos anteriormente – e estações remotas de controle, é necessário o uso de um hardware intermediário, que realize a conversão e tradução dos diferentes padrões de transmissão de mensagens envolvidos na operação, conhecido como gateway. O gateway é um dispositivo responsável por realizar a conexão entre duas ou mais redes de comunicação e que atua para que não ocorram problemas de comunicação ocasionados por arquiteturas, padrões ou protocolos de comunicação diferentes. Ele é o agente que garante a interoperabilidade dos elementos de rede conectados entre si (Arc Advisory Group, 2020).

No caso de um gateway OPC UA, sua função é a de coletar dados de algum dispositivo de controle compatível com o protocolo, e instanciá-los no *AddressSpace* do Servidor, permitindo que outros Clientes realizem a leitura e escrita dos valores de variáveis através dos serviços OPC UA. Conforme explicado por Ausberger e Stetina

(2019), o gateway precisa ter implementado em sua aplicação funções que realizem tarefas de comunicação (tais como conexão, reconexão, tratamento de erros, etc.), bem como procura de estruturas de dados e verificação da integridade dessas estruturas.

A figura 23 ilustra um dispositivo de controle recebendo requisições de forma direta por um cliente e também através do uso de um gateway OPC UA. A operação do gateway ocorre de maneira que, iniciada a conexão com a estrutura de dados que se tem interesse, a aplicação passa a atualizar com o dispositivo alvo de forma sincronizada. Ao receber a requisição de um Cliente, o gateway registra a requisição e a coloca em uma pilha de sincronizações, que resolvida irá retornar ao Cliente seu pedido (Ausberger; Stetina, 2019).

Figura 23 - Interface Gateway OPC UA



Fonte: Ausberger; Stetina (2019)

O espectro de funções que um gateway pode suportar dentro da automação industrial vem aumentando, junto do estabelecimento de novas tecnologias voltadas para Internet Industrial (IIoT). O conceito moderno compreende dispositivos que realizam interface de instrumentos que se encontram na base da pirâmide de automação com o armazenamento dos dados em nuvem. Esses produtos são capazes de ir além de sua função de traduzir protocolos e oferecem aplicações completas, tais como análise em tempo real de dados unificados em um servidor rodando em nuvem (Arc Advisory Group, 2020).

2.5 Raspberry Pi 3

A Raspberry Pi é um computador compacto, com tamanho próximo de um cartão de crédito, desenvolvido pelo britânico Eben Upton, inicialmente com o objetivo de ser utilizado no aprendizado de crianças dentro do ambiente de programação e robótica. Os primeiros protótipos da Raspberry se tratavam de placas com capacidade limitada, de modo que Upton e demais colegas fundaram a Raspberry Pi Foundation, e em meados de 2012 lançaram o primeiro modelo da placa com processamento semelhante aos conhecidos hoje (Upton; Halfacree, 2017).

O modelo escolhido para ser utilizado é a Raspberry Pi 3 Modelo B, que roda diversas distribuições Linux e compatível com Windows 10, além de outros sistemas operacionais desenvolvidos especialmente para plataformas de IoT. A figura 24 demonstra o modelo escolhido para uso neste trabalho.

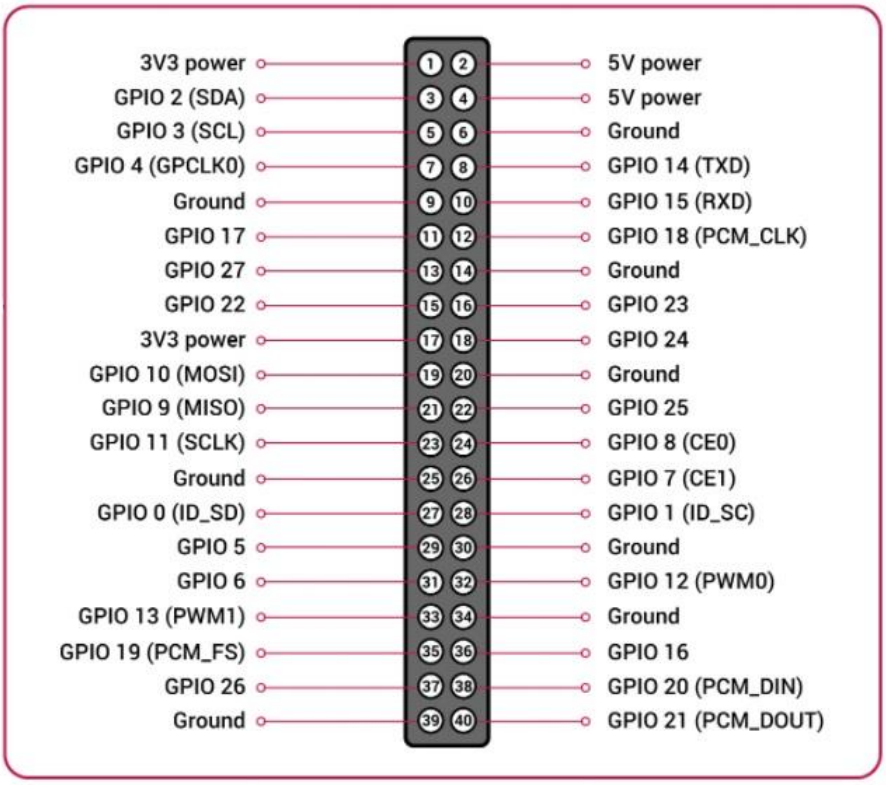
O processador embutido na placa é um Broadcom BCM2837 arquitetura 64 bits, com memória RAM de 1 GB. Ela conta com portas USB que permitem a conexão com dispositivos genéricos (teclado, mouse, pen drives), saída HDMI para televisão e monitores, cartão de memória SD e alimentação via cabo micro USB. Conta com conector de entradas e saídas de 40 pinos, podendo ser expandida com o uso de diversos módulos extras, rádio wireless, Wi-Fi e Bluetooth, tornando uma alternativa conveniente em aplicações sem fio (Upton; Halfacree, 2017).

Figura 24 - Placa Raspberry Pi 3

Fonte: Upton; Halfacree (2017)

O conector de 40 pinos da placa permite a conexão com diversos dispositivos de interface para o usuário, como teclados e displays. Para este trabalho, será utilizada apenas uma saída para o acionamento de um LED simples, a fim de sinalizar a escrita de uma variável no Espaço de Endereçamento do gateway. A figura 25 apresenta a legenda dos pinos de GPIO da Raspberry Pi 3.

Figura 25 - Legenda GPIOs Raspberry Pi 3



Fonte: Upton; Halfacree (2017)

2.6 Trabalhos Correlatos

Nos dias de hoje, a evolução da tecnologia em redes Industriais e wireless aponta a tendência de se trabalhar com sistemas que utilizam da tecnologia sem fio na comunicação entre dispositivos eletrônicos inteligentes.

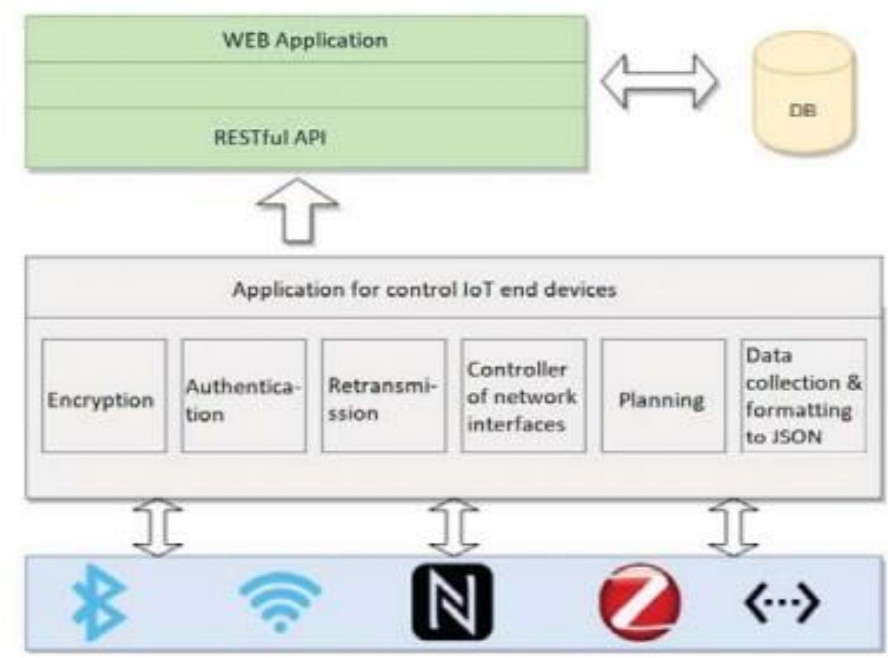
As vantagens para isso passam desde a uma redução nos custos de cabeamento, maior confiabilidade na troca de mensagens, até a robustez necessária para que opere em conjunto com outros dispositivos IoT, com aplicação em diversos novos setores, como os sistemas de distribuição elétrica inteligentes.

No uso aplicado em IoT, os *smart gateways* podem agir para suprir o problema que surge ao se trabalhar com tantos dispositivos heterogêneos (muitos de baixo

custo), que trocam dados em vários formatos. Um dos objetivos em IoT é criar casas inteligentes para que coletivamente formem cidades inteligentes; isso somente será possível se todos os nós dentro da rede conseguirem decodificar as mensagens trocadas de maneira correta (Mastilak et. al., 2018).

O funcionamento de um hardware desse tipo é demonstrado na figura 26, proposto por Mastilak et. al. (2018), um smart gateway voltado para comunicação de dispositivos de baixo custo. Uma aplicação web fornece interface para comunicação com dispositivos externos, informações de instrumentos de campo armazenados em um banco de dados. O formato de arquivo escolhido para empacotamento pelos autores foi o JSON, mas este conceito pode ser sofrer alterações em outros ambientes.

Figura 26 - Arquitetura Smart Gateway



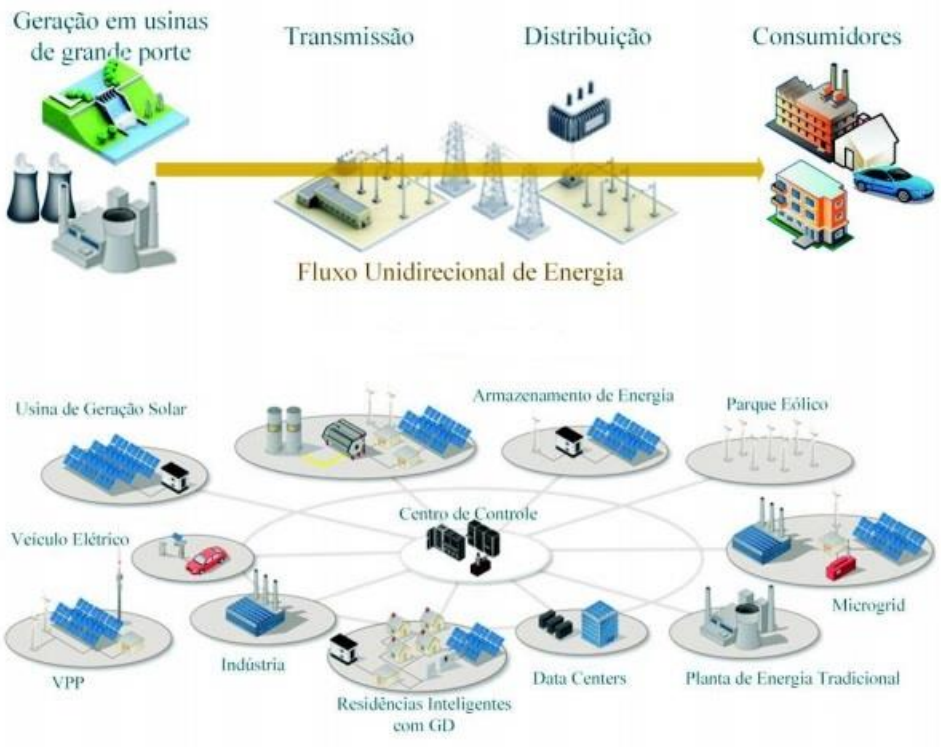
Fonte: Mastilak et. al (2018)

A aplicação que controla os dispositivos de campo é composta por seis módulos: encriptação, autenticação, retransmissão (que faz o tratamento de eventuais pacotes perdidos), controlador de interface de rede, planejamento e coleta de dados/formatação. O módulo de autenticação verifica se os dispositivos disponíveis na conexão estão aptos a criarem sessão com o smart gateway; em caso afirmativo, este dá o sinal e permite que o equipamento acesse com uma senha. Uma etapa adicional de troca hash é efetuada (Mastilak et. al., 2018).

Outro trabalho que apresenta proposta de um gateway para uso com aplicação no cenário de IoT é feito por Keller (2016). O trabalho aborda o desenvolvimento de um *middleware* para integração de redes de automação distribuídas que operam sob o protocolo Modbus RTU, utilizando o protocolo de IoT MQTT (*Message Queing Telemetry Transport*).

Esse estudo levou em consideração o sistema de geração de energia solar fotovoltaica, geograficamente espalhado numa região, para centralizar o monitoramento numa única planta de controle virtual. A figura 27 apresenta uma comparação entre redes elétricas convencionais e redes elétricas inteligentes (também conhecidas por *smart grids*), e que ilustra o papel centralizador do dispositivo desenvolvido (Keller, 2016).

Figura 27 - Comparação Redes de Geração Convencionais e Inteligentes



Fonte: Keller (2016)

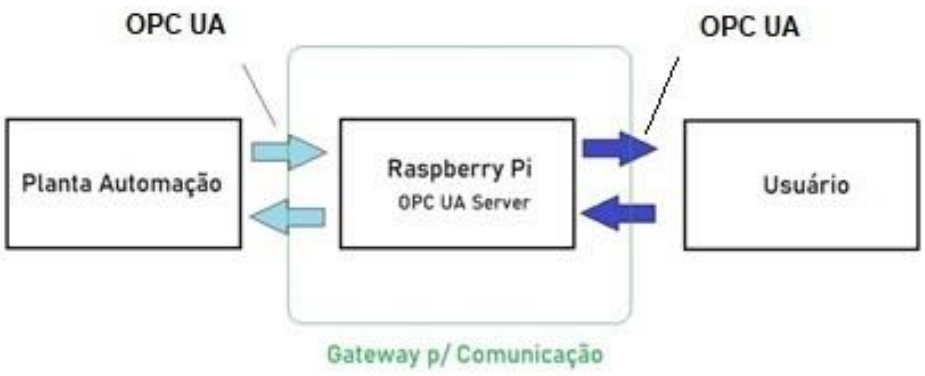
Atualmente, a geração de energia distribuída é um dos temas de ponta nas discussões acerca do planejamento de um ambiente mais sustentável. As vantagens para o uso vão desde a economia em transmissão e perdas reduzidas a diversidade na matriz energética da planta (Keller, 2016).

3 METODOLOGIA

Neste capítulo será apresentado a metodologia utilizada para a implementação de um gateway para uma planta de automação didática, equipada com dispositivos que se comunicam utilizando Profibus DP-PA. O gateway deverá permitir o acesso aos dados de simulação da planta através dos computadores dos alunos que estiverem conectados na rede local da universidade.

Para a fase inicial de projeto, foram definidos alguns hardwares e ferramentas de software que serão utilizados na solução do problema. A justificativa para o uso e características dos elementos escolhidos serão discutidos nesse capítulo. A figura 28 apresenta uma ilustração da rede que se busca implementar, definindo a Raspberry Pi 3 como plataforma utilizada com vista nas facilidades de implementação expostas no capítulo anterior.

Figura 28 - Diagrama de Rede do Gateway



Fonte: Elaborado pelo Autor

3.1 Sistema Operacional Raspberry Pi OS

Para implementação do gateway utilizando a Rasberry Pi 3 é necessária a instalação de um Sistema Operacional adequado, que suporte os softwares e ferramentas responsáveis pela troca de mensagens entre os elementos da rede. Dentre as opções viáveis, foi escolhido o Raspberry Pi OS.

O Raspberry Pi OS, anteriormente Raspbian, é um sistema operacional de código aberto otimizado para rodar em placas Raspberry Pi. Ele é baseado no Debian, uma distribuição Linux, e conta com instalação simples, interface amigável e mais de 35.000 pacotes já pré compilados (Raspbian, 2020).

A versão escolhida para utilização é Raspberry Pi OS with desktop, de agosto de 2020. Este sistema conta com ambiente gráfico já preparado para rodar aplicações em Python 3.

3.2 Ferramenta OPC UA Server-Client

O condicionamento e gerenciamento dos pacotes de mensagens gerados pela planta didática PD3P será realizado através da implementação de um gateway OPC UA na placa Raspberry Pi 3. Esse gateway será visualizado pelo CLP como um cliente, e pelos computadores de alunos como um servidor, ou seja, irá realizar a coleta de dados e permitir o tráfego das mensagens para um número superior de máquinas.

Um Servidor OPC UA é composto por três elementos principais: Objetos, Aplicação Servidor e o Espaço de Endereçamento. Os objetos dizem respeito a objetos físicos que podem ser consultados pela aplicação Servidor; exemplos de objetos são os próprios dispositivos terminais de redes *fieldbus*, como um timer. A aplicação é o software que roda a ferramenta Servidor OPC, uma interface para atuar em conjunto das saídas OPC. O espaço de endereçamento guarda as informações de quais pontos da rede são acessíveis pelo Servidor e quais referências uns possuem com os outros (OPC FOUNDATION, 2020).

O escopo deste trabalho está limitado no desenvolvimento da aplicação Servidor. Isso se dará através da implementação de um código Python que irá ser executado na Raspberry Pi 3. Para simular o Servidor original e o Cliente OPC UA da figura 22 serão utilizados softwares disponibilizados gratuitamente, que são discutidos nos próximos tópicos.

3.2.1 Prosys OPC UA Simulation Server

O Prosys OPC UA Simulation Server é um servidor gratuito, *standalone* e multiplataforma, desenvolvido pela finlandesa Prosys, que permite a simulação de

estruturas de dados geradas por dispositivos que possuem OPC UA integrados, como o CLP da planta que se deseja ter acesso. O software permite a configuração de diversos ambientes de simulação, e oferece uma série de sinais predefinidos que podem ser vinculados a objetos e variáveis dentro do contexto OPC (Prosys OPC Ltd., 2021).

A instalação do software no Windows é feita através de instalador disponibilizado no site do fabricante. Ao executar o programa, uma janela inicial é aberta contendo informações sobre Status do Servidor, Endereço de Conexão UA TCP e UA HTTPS e horário do Servidor. A tela inicial é exibida na figura 29. No exemplo, o servidor está sendo executado no computador CosmosII (que na rede local representa IP estático 192.168.1.3) utilizando a porta 53530.

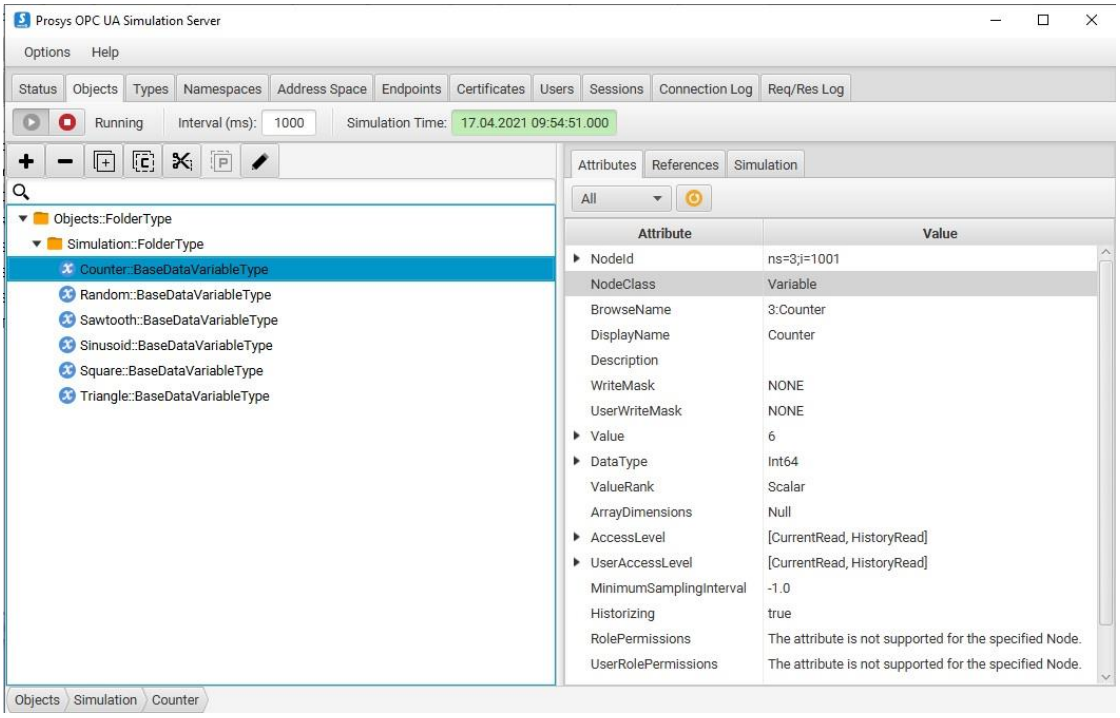
Figura 29 - Tela Inicial Prosys OPC UA Simulation Server

Fonte: Capturado pelo Autor

A aba Objects possibilita a criação de Objetos e Variáveis personalizados. Para testes iniciais, foi criado um Objeto contador com uma variável que armazena a contagem de 1 a 20. Também foram utilizados os sinais de simulação de uma senoide e um gerador de números aleatórios. É possível consultar na interface do servidor os atributos relacionados ao Nó que representa o Objeto, conforme a figura 30. Essas informações do *AddressSpace* serão utilizadas posteriormente pela aplicação cliente

ao solicitar a leitura da variável que se busca conhecer. É comum que servidores e CLPs com OPC UA integrados possuam uma interface gráfica semelhante a da figura, que permitem a consulta dos atributos. Em situações onde se é necessário descobrir os endereços, o OPC UA possui serviços que possibilitam a busca de dados pela aplicação cliente sem que necessariamente se conheça as informações dos atributos dos nós no servidor.

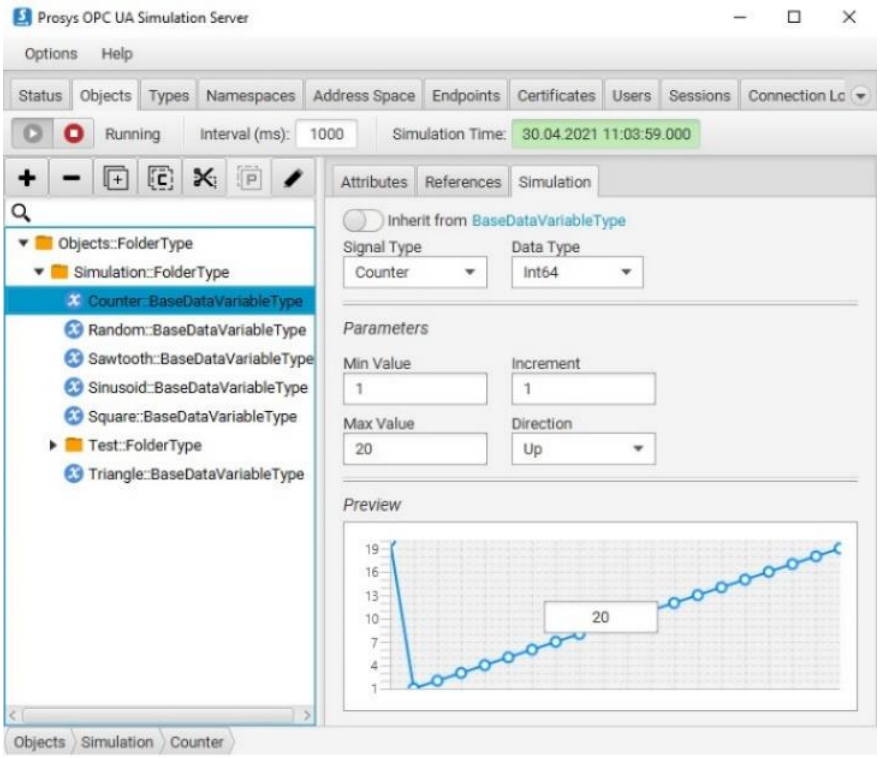
Figura 30 - Tela de Objetos e Atributos do *AddressSpace*



Fonte: Capturado pelo Autor

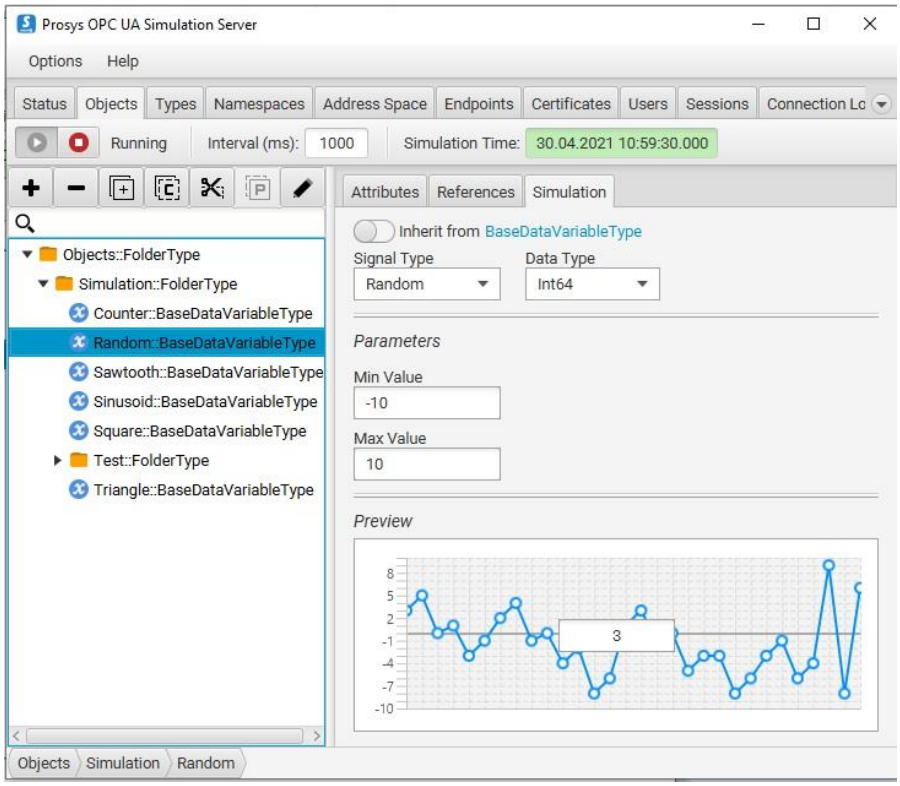
As figuras 31, 32 e 33 apresentam os sinais que são gerados pelo simulador e que serão utilizados para leitura do servidor.

Figura 31 - Simulação Contador



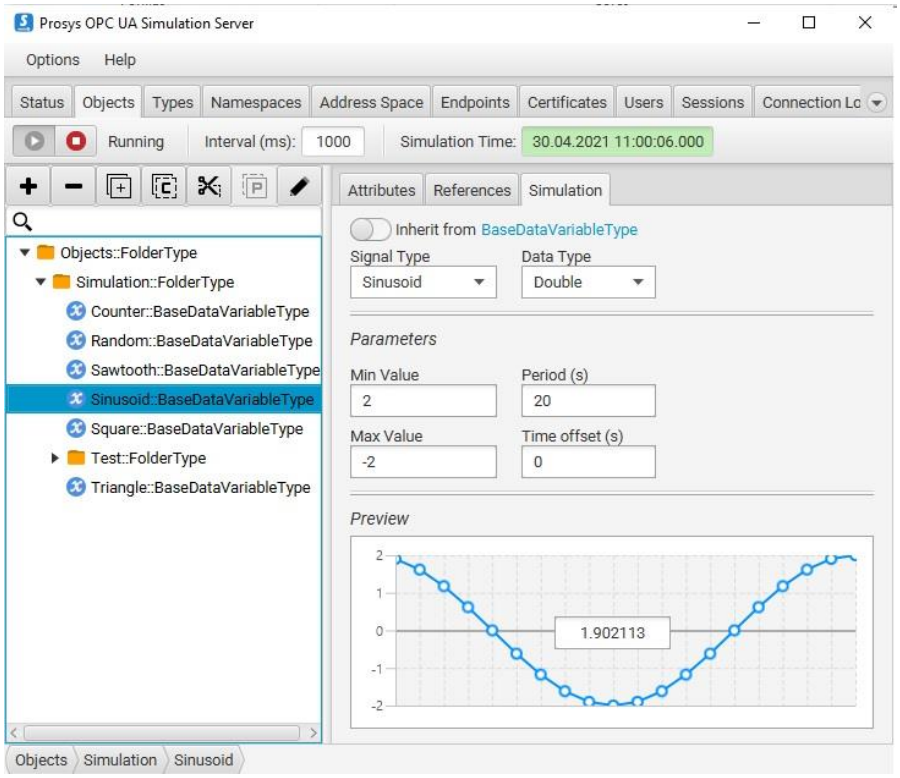
Fonte: Capturado pelo Autor

Figura 32 - Simulação Número Aleatório



Fonte: Capturado pelo Autor

Figura 33 - Simulação Senoide



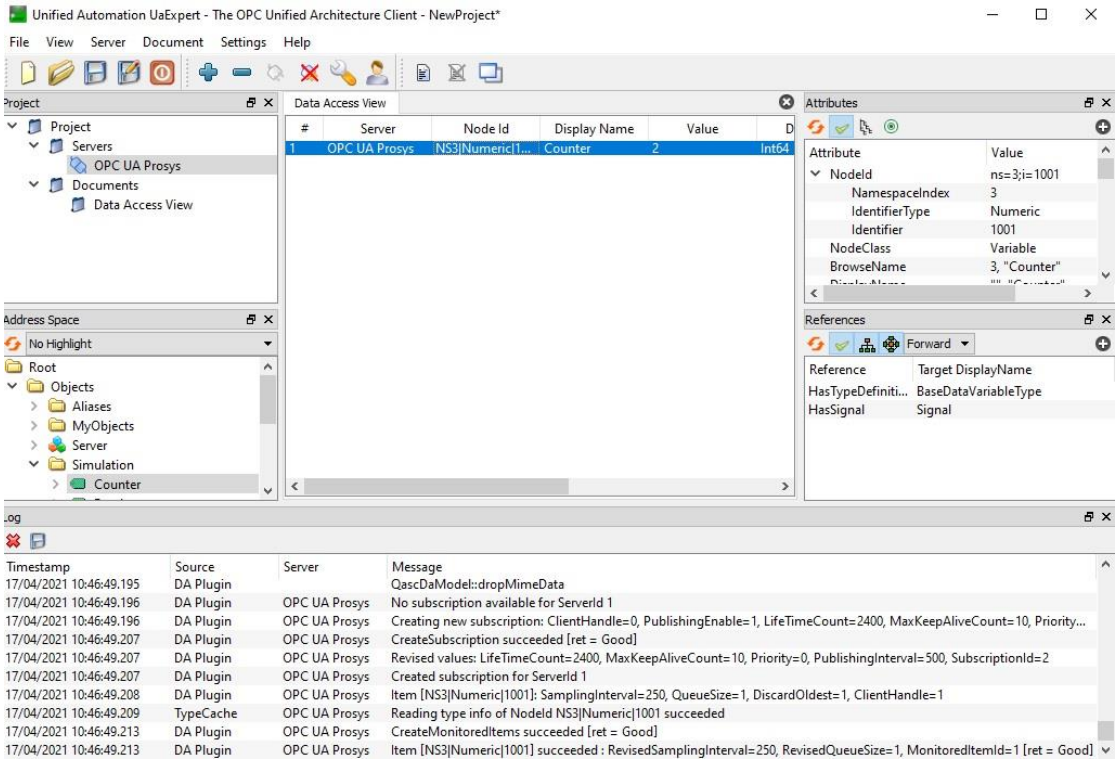
Fonte: Capturado pelo Autor

3.2.2 UaExpert

O acesso aos dados pelos alunos será realizado através do software UaExpert. Ele é um programa gratuito disponível para Windows e Linux e que foi projetado para ser um cliente teste em redes OPC UA. Possui recursos de acesso a dados em tempo real ou armazenados em histórico, alarmes e condições, bem como funções de invocar métodos.

A figura 34 demonstra um exemplo da interface do UaExpert, conectado no endereço TCP do servidor Prosys OPC UA. Realizada a conexão, é possível navegar pelos Nós do *AddressSpace* do Servidor na janela da esquerda, “AddressSpace”. Os objetos são organizados na árvore em pastas, e encontrado o objeto de interesse é possível arrastá-lo até a janela “Data Access View”, onde se passa a monitorar a variável em tempo real. Selecionar o objeto permitirá também ao usuário verificar as informações de endereço no *AddressSpace*, na janela “Attributes”.

Figura 34 - Tela Cliente UaExpert



Fonte: Capturado pelo Autor

3.3 Aplicação Servidor OPC UA

Como explicado anteriormente, o escopo deste trabalho está resumido na criação de uma aplicação Servidor OPC UA que seja capaz de realizar comunicação com um controlador preparado para o protocolo, e que permita o acesso aos dados de leitura e escrita por outros computadores na rede local. A aplicação será feita utilizando um código em Python desenvolvido para o trabalho.

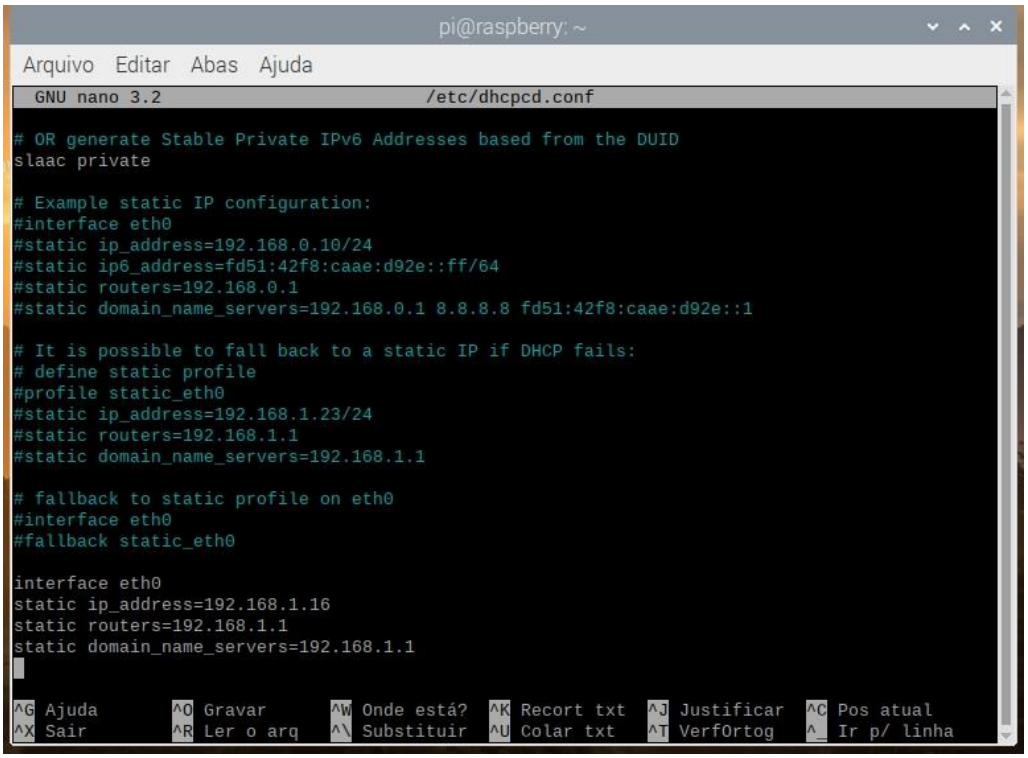
3.3.1 Configurações Iniciais

Com o Raspberry Pi OS já instalado na placa, foi primeiro configurado o IP estático na máquina para que inicie usando sempre o mesmo endereço e facilite a localização e conexão entre os dispositivos. Em sistemas operacionais baseados em Linux, as configurações de endereço do computador estão armazenadas no arquivo dhcpcd.conf, localizado no diretório etc. Para acessar o arquivo, foi aberto o Terminal e digitado o comando “sudo nano /etc/dhcpcd.conf”. Dentro do arquivo, foram adicionadas as linhas que definem o IP fixo e o roteador

interface eth0
static ip_address=192.168.1.16
static routers=192.168.1.1
static domain_name_servers=192.168.1.1

onde “static ip_address” determina o endereço IPv4, arbitrariamente escolhido, que será utilizado pelos clientes para se conectar ao gateway e “static routers” determina o endereço IPv4 do roteador na rede local. A figura 35 mostra as linhas acrescentadas para fixação do IP.

Figura 35 - IP Estático



Fonte: Capturado pelo Autor

3.3.2 Biblioteca Python-OPCUA

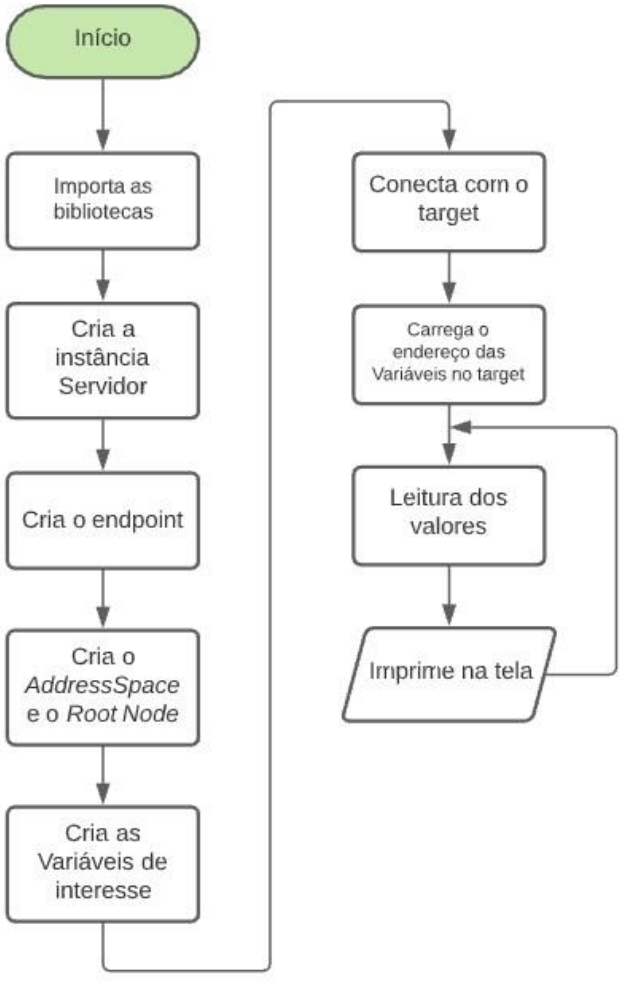
O FreeOPCUA é um projeto de código aberto que foi desenvolvido para prover aos usuários de Python uma série de recursos na implementação de servidores e clientes OPC UA. A instalação da biblioteca na Raspberry é feita com o pacote instalador pip, pelo Terminal do Linux com o comando “sudo pip 3 install freeopcua”.

Por dependência de outras bibliotecas, é necessário fazer antes a instalação do pacote cryptography, digitando no Terminal “sudo pip3 install cryptography”.

3.3.3 Algoritmo Servidor

O algoritmo que será executado na Raspberry Pi 3 foi escrito em Python. A figura 36 apresenta um croqui do funcionamento dele.

Figura 36 - Croqui Aplicação Servidor



Fonte: Elaborado pelo Autor

O código do programa escrito para realizar a comunicação com o target OPC UA é demonstrado nas figuras 37 e 38. O algoritmo inicia com a importação das bibliotecas Server e Client OPC UA, e realizando a criação da instância servidor na Raspberry Pi 3, disponibilizando um *endpoint* para clientes se conectarem. É feita

também a importação da biblioteca RPi.GPIO, responsável por habilitar os pinos de entrada e saída do módulo Raspberry Pi 3.

No código da figura 37, nas linha 7 e 8, o *endpoint* foi arbitrariamente alocado na porta 4840 do endereço 192.168.1.36, que é o IP estático do sistema configurado anteriormente. As linhas de código 10 e 11 criam, respectivamente, o *AdressSpace* do gateway e o *Root Node* ao qual os demais nodes estarão vinculados.

Para leitura e armazenamento dos dados dentro do espaço de endereçamento do gateway foi criado um Objeto de nome “Parâmetros”, que receberá as variáveis “Contador”, “Random”, “Senoide” e “Setpoint”, além da variável LDR que armazenará os valores lidos pelo sensor montado na protoboard. Esses comandos são exibidos nas linhas 14 a 19 do algoritmo do servidor. A linha 20 dá as permissões de escrita na variável Setpoint, que permitirá que usuários alterem seu valor de acordo com o desejado.

A conexão do gateway com o target é definida nas linhas 25 e 26 do código. Nelas, é atribuída a variável “sysopc” com o comando Client() o endereço de rede do servidor OPC UA que se busca ler. No caso deste trabalho, o simulador foi instalado e executado no endereço opc.tcp://192.169.1.8:53530/OPCUA/SimulationServer. Nas linhas 30, 31 e 32 é utilizado o comando get_node() para informar o *namespace* e o índice das variáveis que se busca realizar a leitura no simulador Prosys OPC UA.

Os comandos das linhas 37 a 47 realizam uma parametrização inicial dos pinos GPIOs do gateway, definindo os pinos 10, 18 e 36 da placa como saídas dos LEDs e inicialmente desligadas. O pino 7 é definido como uma entrada, que recebe o sinal do sensor LDR e que será utilizado para controle do acionamento do LED no pino 36.

Nas linhas 49 a 57 é definida a função “rc_time()” que recebe como parâmetro o pino onde o sensor LDR está conectado. A Raspberry Pi 3 não possui GPIO analógica, portanto é utilizado um contador que incrementa 1 a cada centésimo de segundo até que o pino da placa mude para o estado alto, o que ocorre quando o capacitor se carrega. A função então retorna o valor armazenado na variável count, que é utilizado para acionar o LED de sinalização.

O gateway começa a funcionar na linha 59, com o comando “server.start()”.

Figura 37 - Algoritmo Servidor

```
from opcua import Server, Client
from datetime import datetime
import time
import RPi.GPIO as GPIO

#CRIA O SERVIDOR QUE RODA NA RASPBERRY PI=================
server = Server() #Cria o Servidor
url="opc.tcp://192.168.1.36:4840"
server.set_endpoint(url)
name = "RASPBERRY_OPCUA_SERVER"
addspace = server.register_namespace(name)
node = server.get_objects_node() # CRIA O ROOT NODE NO SERVER

Param = node.add_object(addspace, "Parametros")
isrvr_Contador = Param.add_variable(addspace, "Contador", 0)
isrvr_Random = Param.add_variable(addspace, "Random", 0)
isrvr_Senoide = Param.add_variable(addspace, "Senoide", 0)
isrvr_Setpoint = Param.add_variable(addspace, "Set Point", 8)
isrvr_LDR = Param.add_variable(addspace, "LDR", 0)
isrvr_Setpoint.set_writable()
isrvr_LDR.set_writable()
#=========================================================

#CRIA A CONEXÃO COM O SYSOPCUA============================
sysopc=Client("opc.tcp://192.168.1.8:53530/OPCUA/SimulationServer")
sysopc.connect()
now = datetime.now().time()
print ("Conexão estabelecida com o target as {}".format(now))

iContador = sysopc.get_node("ns=3;i=1001") #Armazena em iContador o end. no PróSy
iRandom = sysopc.get_node("ns=3;i=1002")
iSenoide= sysopc.get_node("ns=3;i=1004")
#=========================================================
GPIO.setmode(GPIO.BOARD) #Utiliza a numeração de pinos da placa física
GPIO.setwarnings(False)

GPIO.setup(10, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(18, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(11, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.output(18, False)
#=========================================================
delayt = .1
vLDR = 0
ldr = 7
led = 36
GPIO.setup(led, GPIO.OUT)
GPIO.output(led, False)

def rc_time (ldr):
    count = 0
    GPIO.setup(ldr, GPIO.OUT)
    GPIO.output(ldr, False)
    time.sleep(delayt)
    GPIO.setup(ldr, GPIO.IN)
    while (GPIO.input(ldr) == 0):
        count += 1
    return count
#=========================================================
server.start()
```

Fonte: Elaborado pelo Autor

Figura 38 - Algoritmo Servidor

```
60  try:
61      while True:
62          now = datetime.now().time()
63          vContador = iContador.get_value()
64          vRandom = iRandom.get_value()
65          vSetpoint = isrvr_Setpoint.get_value()
66          vSenoide = iSenoide.get_value()
67
68          isrvr_Contador.set_value(vContador), isrvr_Random.set_value(vRandom)
69          isrvr_Senoide.set_value(vSenoide)
70          isrvr_Setpoint.set_value(vSetpoint)
71          isrvr_LDR.set_value(vLDR)
72          GPIO.output(18, not GPIO.input(11))
73          ## AQUI VAI A PARTE DO LED DE SETPOINT==================
74          if vSetpoint == 10:
75              GPIO.output(10, GPIO.HIGH)
76          else:
77              GPIO.output(10, GPIO.LOW)
78          ## PARTE DO LDR=============================
79          vLDR = rc_time(ldr)
80          if (vLDR <= 8000 ):
81                  GPIO.output(led, False)
82          if (vLDR > 8000):
83                  GPIO.output(led, True)
84          ## ENCERRA A PARTE DO LDR=========================
85          time.sleep(1)
86  finally:
87      GPIO.output(11, False)
88      GPIO.cleanup()
```

Fonte: Elaborado pelo Autor

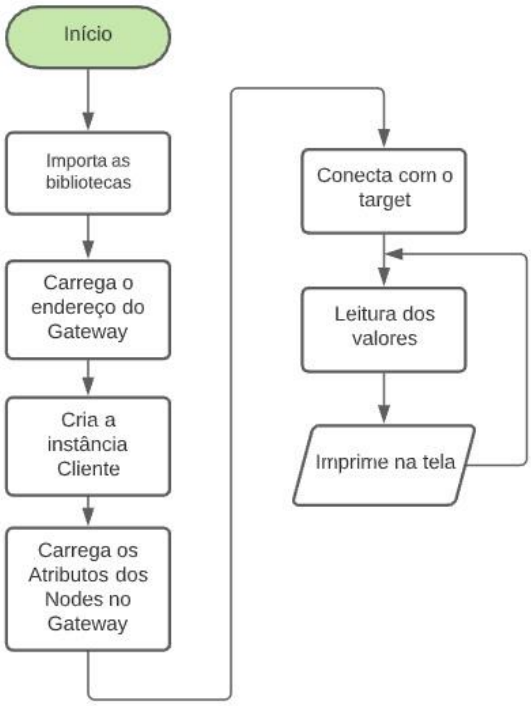
A partir da linha 60 da figura 38, foi criado um laço de repetição para que a cada segundo as variáveis alocadas no *AddressSpace* do gateway realizem a leitura de sua respectiva variável no simulador Prosys OPC UA Server. O laço de repetição está implementado no código entre as linhas 61 e 85. Nas linhas 74 e 76 é feita a comparação do valor de Set Point para determinar o acionamento da saída no pino 10, com os comandos de acionamento e desligamento escritos nas linhas de código 75 e 77, respectivamente. Nas linhas 80 e 82 é feita a comparação do valor de armazenamento do sensor LDR, com o acionamento do LED de sinalização sendo implementado nas linhas 81 e 83.

3.4 Cliente OPC UA

Para auxiliar o desenvolvimento do trabalho, foi escrito um algoritmo que implementa a aplicação cliente. Esse cliente pode ser utilizado em computadores com Linux ou Windows que tenham o Python e as bibliotecas OPC UA previamente

instalados. A figura 39 apresenta um croqui do funcionamento dele. A figura 40 apresenta o código que implementa a aplicação cliente.

Figura 39 - Croqui Aplicação Cliente



Fonte: Elaborado pelo Autor

Figura 40 - Algoritmo Cliente

gateway4.py | client_gateway_opcua1.py*

```
1  from opcua import  Client
2  from datetime import datetime
3  import time
4
5  uri = "opc.tcp://192.168.1.16:4840"
6  client = Client(uri)
7  client.connect()
8  print ("Conexão estabelecida em {}".format(uri))
9  root = client.get_root_node()
10
11 #children = root.get_children()
12 #objects = client.get_objects_node()
13
14 iContador = client.get_node("ns=2;i=2")
15 iRandom = client.get_node("ns=2;i=3")
16 iSenoide = client.get_node("ns=2;i=4")
17 iSetpoint = client.get_node("ns=2;i=5")
18
19 while True:
20     now = datetime.now().time()
21     vContador = iContador.get_value()
22     vRandom = iRandom.get_value()
23     vSetpoint = iSetpoint.get_value()
24     vSenoide = iSenoide.get_value()
25     print ("TIME: {}".format(now),"\t CONTADOR: {}".format(vContador))
26     print ("\t RANDOM: {}".format(vRandom),"\t\tSENO: {}".format(vSenoide))
27     print ("\t SETPOINT: {}".format(vSetpoint))
28     time.sleep(1)
```

Fonte: Elaborado pelo Autor

O algoritmo da figura 40 inicia com a importação da biblioteca Client OPC UA. As linhas de código 5, 6 e 7 são responsáveis por determinarem o endereço de conexão OPC TCP do gateway e solicitarem a conexão.

Na linha 9, com o comando “client.get_rood_node()”, é feita uma consulta dos atributos do Nó Root no gateway, que contém os objetos que se busca monitorar. Como os Nodes já são conhecidos por via das ferramentas já descritas, seus endereços são parametrizados nas linhas de código 14, 15, 16 e 17.

A partir desse ponto, foi criado um laço de repetição para que a cada segundo fossem realizadas leituras nos Objetos solicitados: Contador, Random, Seno e Valor de Set Point. Os resultados são exibidos no terminal.

4 ANÁLISE DOS RESULTADOS

No decorrer do desenvolvimento deste trabalho, foram realizados testes de comunicação entre os elementos e de validação do projeto, descritos no capítulo 3. No presente capítulo, será demonstrado e discutido os resultados obtidos mais pertinentes.

4.1 Leitura dos Valores

Com o Prosys OPC UA Server sendo executado na rede, o gateway realizou uma série de leituras dos sinais e valores gerados pelo simulador. Para validação inicial, foi instanciado no *AdressSpace* do gateway um Nó que recebe e armazena os valores de um contador, um gerador de números aleatórios, uma senoide e um valor de referência (*setpoint*). Com outro computador na rede, é possível acessar os dados lidos pelo gateway para visualização. A figura 41 apresenta os valores lidos pelo gateway e impressos na tela do Terminal.

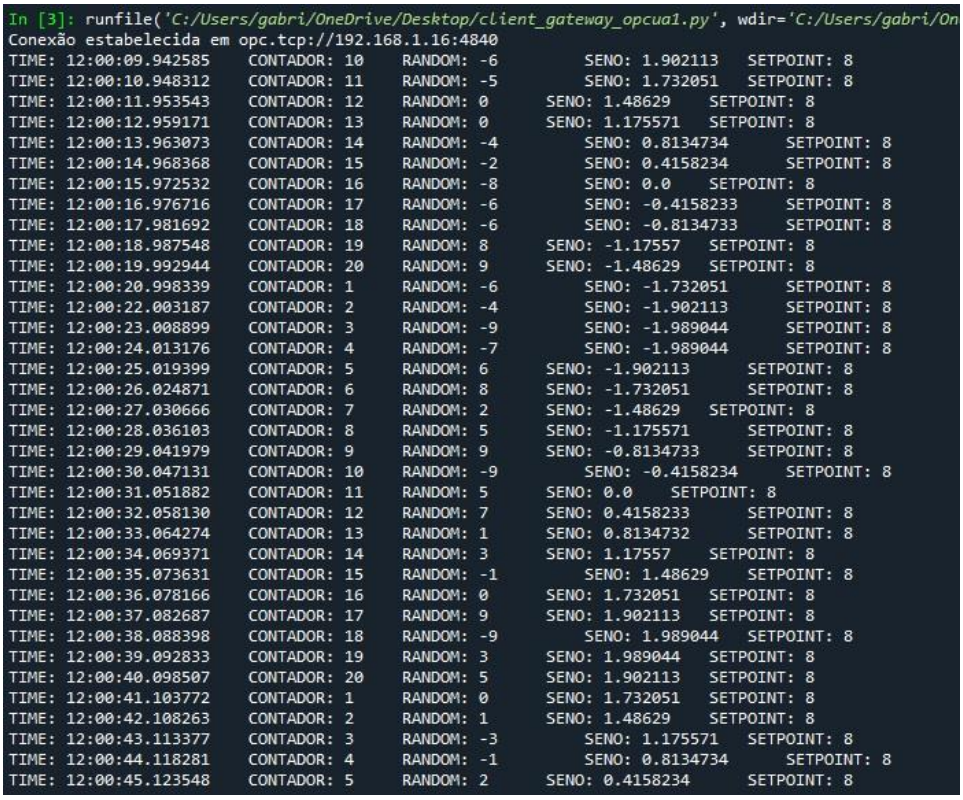
Figura 41 - Leitura de Valores no Gateway

```
Python 3.7.3 (default, Jan 22 2021, 20:04:44)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: /home/pi/Desktop/gateway4.py ==========
Conexão estabelecida com o target as 12:00:00.221847
Endpoints other than open requested but private key and certificate are not set.
Listening on 192.168.1.16:4840
TIME:12:00:00.226497    CONTADOR:1      RANDOM:3        SENO:0.0                SETPOINT:8
TIME:12:00:01.241636    CONTADOR:2      RANDOM:-1       SENO:0.4158233          SETPOINT:8
TIME:12:00:02.264315    CONTADOR:4      RANDOM:3        SENO:1.17557            SETPOINT:8
TIME:12:00:03.280950    CONTADOR:5      RANDOM:5        SENO:1.48629            SETPOINT:8
TIME:12:00:04.308992    CONTADOR:6      RANDOM:0        SENO:1.732051           SETPOINT:8
TIME:12:00:05.325520    CONTADOR:7      RANDOM:2        SENO:1.902113           SETPOINT:8
TIME:12:00:06.338971    CONTADOR:8      RANDOM:-3       SENO:1.989044           SETPOINT:8
TIME:12:00:07.353863    CONTADOR:9      RANDOM:-1       SENO:1.989044           SETPOINT:8
TIME:12:00:08.369797    CONTADOR:10     RANDOM:-6       SENO:1.902113           SETPOINT:8
TIME:12:00:09.384994    CONTADOR:11     RANDOM:-5       SENO:1.732051           SETPOINT:8
TIME:12:00:10.401189    CONTADOR:12     RANDOM:0        SENO:1.48629            SETPOINT:8
TIME:12:00:11.416156    CONTADOR:13     RANDOM:0        SENO:1.175571           SETPOINT:8
TIME:12:00:12.431133    CONTADOR:14     RANDOM:-4       SENO:0.8134734          SETPOINT:8
TIME:12:00:13.446627    CONTADOR:15     RANDOM:-2       SENO:0.4158234          SETPOINT:8
TIME:12:00:14.464691    CONTADOR:16     RANDOM:-8       SENO:0.0                SETPOINT:8
TIME:12:00:15.479896    CONTADOR:17     RANDOM:-6       SENO:-0.4158233                 SETPOINT:8
TIME:12:00:16.493973    CONTADOR:18     RANDOM:-6       SENO:-0.8134733                 SETPOINT:8
TIME:12:00:17.508649    CONTADOR:19     RANDOM:8        SENO:-1.17557           SETPOINT:8
TIME:12:00:18.521300    CONTADOR:20     RANDOM:9        SENO:-1.48629           SETPOINT:8
TIME:12:00:19.543586    CONTADOR:1      RANDOM:-6       SENO:-1.732051          SETPOINT:8
TIME:12:00:20.563464    CONTADOR:2      RANDOM:-4       SENO:-1.902113          SETPOINT:8
TIME:12:00:21.577103    CONTADOR:3      RANDOM:-9       SENO:-1.989044          SETPOINT:8
TIME:12:00:22.590616    CONTADOR:4      RANDOM:-7       SENO:-1.989044          SETPOINT:8
TIME:12:00:23.610755    CONTADOR:5      RANDOM:6        SENO:-1.902113          SETPOINT:8
TIME:12:00:24.635069    CONTADOR:6      RANDOM:8        SENO:-1.732051          SETPOINT:8
TIME:12:00:25.650740    CONTADOR:7      RANDOM:2        SENO:-1.48629           SETPOINT:8
TIME:12:00:26.673940    CONTADOR:8      RANDOM:5        SENO:-1.175571          SETPOINT:8
TIME:12:00:27.690058    CONTADOR:9      RANDOM:9        SENO:-0.8134733                 SETPOINT:8
TIME:12:00:28.704575    CONTADOR:10     RANDOM:-9       SENO:-0.4158234                 SETPOINT:8
TIME:12:00:29.717211    CONTADOR:11     RANDOM:5        SENO:0.0                SETPOINT:8
TIME:12:00:30.730205    CONTADOR:12     RANDOM:7        SENO:0.4158233          SETPOINT:8
TIME:12:00:31.747084    CONTADOR:13     RANDOM:1        SENO:0.8134732          SETPOINT:8
TIME:12:00:32.761205    CONTADOR:14     RANDOM:3        SENO:1.17557            SETPOINT:8
TIME:12:00:33.776062    CONTADOR:15     RANDOM:-1       SENO:1.48629            SETPOINT:8
TIME:12:00:34.800422    CONTADOR:16     RANDOM:0        SENO:1.732051           SETPOINT:8
```

Fonte: Capturado pelo Autor

A figura 42 apresenta os valores lidos pela aplicação Cliente no console do Spyder, uma IDE gratuita para Python , rodando em Windows na rede local.

Figura 42 - Leitura de Valores no Cliente



Fonte: Capturado pelo Autor

O mesmo acesso pode ser feito com o UaExpert. A figura 43 ilustra a leitura dos valores de objetos no gateway.

Figura 43 - Leitura de Valores no UaExpert

Data Access View

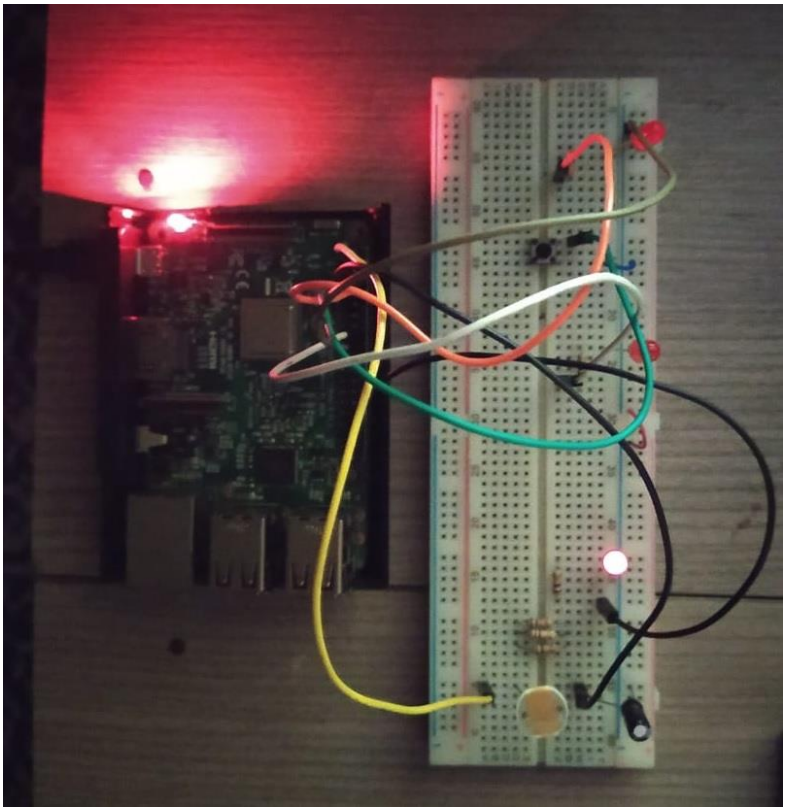
#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp
1	Raspberry Pi 3	NS2|Numeric|2	Contador	3	Int64	17:50:03.381
2	Raspberry Pi 3	NS2|Numeric|3	Random	6	Int64	17:50:03.382
3	Raspberry Pi 3	NS2|Numeric|4	Senoide	1.618034	Double	17:50:03.382
4	Raspberry Pi 3	NS2|Numeric|5	Set Point	8	Int64	21:00:00.000
5	Raspberry Pi 3	NS2|Numeric|6	LDR	4004	Int64	17:50:03.383

Fonte: Capturado pelo Autor

Conforme explicado previamente, um resistor dependente de luminosidade (LDR) foi utilizado para demonstração da leitura de valores do gateway. A figura 44 apresenta a sinalização de funcionamento do sensor, que na baixa incidência de luz

aciona o pino de saída da placa, ligando o LED na parte inferior da protoboard (conectado fisicamente a GPIO número 36 da Raspberry).

Figura 44 – Sinalização de Funcionamento do LDR



Fonte: Capturado pelo Autor

4.2 Escrita dos Valores

Para o teste de escrita dos clientes no gateway, foi criado no servidor uma variável de controle chamada “Set Point” (linha de código 17 da figura 35), que arbitrariamente inicia com o valor 8. Essa variável foi utilizada para fazer o acionamento de um LED com as GPIOS da Raspberry Pi 3. Esse teste foi uma maneira de demonstrar visualmente o procedimento de escrita no Espaço de Endereçamento do gateway, além de também demonstrar o uso de recursos de I/Os oferecidos pela plataforma. Dentro do código da aplicação gateway, foi criada uma condição para que o LED acendesse quando o usuário escrevesse o valor 10 na variável de Set Point.

Para alterar o valor de Set Point, o usuário utiliza o UaExpert. Na mesma janela de visualização, é dado um duplo clique em cima do valor que se deseja escrever. O valor é atualizado no gateway ao pressionar Enter no cliente. As figuras 45 e 46

registram a troca de valor armazenado sendo feita pelo cliente, fazendo a troca do valor e comandando o acionamento do LED. A figura 47 apresenta o terminal localizado no gateway, com os valores armazenados sendo alterados.

Figura 45 – Escrita de Valores no UaExpert

Data Access View

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp
1	Raspberry Pi 3	NS2|Numeric|2	Contador	14	Int64	12:45:33.982
2	Raspberry Pi 3	NS2|Numeric|3	Random	0	Int64	12:45:33.983
3	Raspberry Pi 3	NS2|Numeric|4	Senoide	-1.618034	Double	12:45:33.984
4	Raspberry Pi 3	NS2|Numeric|5	Set Point	8	Int64	21:00:00.000

Fonte: Capturado pelo Autor

Figura 46 – Escrita de Valores no UaExpert

Data Access View

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp
1	Raspberry Pi 3	NS2|Numeric|2	Contador	5	Int64	12:44:44.718
2	Raspberry Pi 3	NS2|Numeric|3	Random	7	Int64	12:44:44.719
3	Raspberry Pi 3	NS2|Numeric|4	Senoide	1.902113	Double	12:44:44.720
4	Raspberry Pi 3	NS2|Numeric|5	Set Point	10	Int64	21:00:00.000

Fonte: Capturado pelo Autor

Figura 47 – Valores Escritos no Gateway

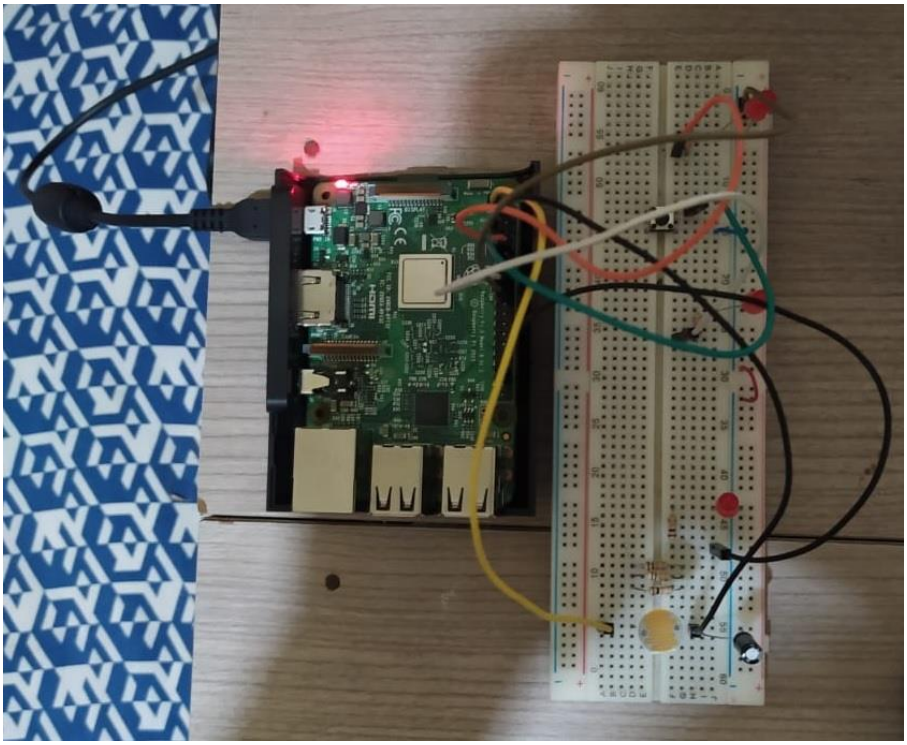
Shell

```
LED LOW
TIME:12:48:06.565457    CONTADOR:7      RANDOM:0        SENO:1.902113          SETPOINT:8
LED LOW
TIME:12:48:07.608798    CONTADOR:8      RANDOM:-5       SENO:1.618034          SETPOINT:8
LED LOW
TIME:12:48:08.655977    CONTADOR:9      RANDOM:-3       SENO:1.175571          SETPOINT:8
LED LOW
TIME:12:48:09.700619    CONTADOR:10     RANDOM:-8       SENO:0.618034          SETPOINT:8
LED LOW
TIME:12:48:10.750394    CONTADOR:11     RANDOM:-7       SENO:0.0               SETPOINT:8
LED LOW
TIME:12:48:11.800394    CONTADOR:12     RANDOM:-2       SENO:-0.618034         SETPOINT:10
LED HIGH
TIME:12:48:12.843924    CONTADOR:13     RANDOM:-1       SENO:-1.175571         SETPOINT:10
LED HIGH
TIME:12:48:13.994322    CONTADOR:15     RANDOM:-4       SENO:-1.902113         SETPOINT:10
LED HIGH
TIME:12:48:15.057878    CONTADOR:16     RANDOM:9        SENO:-2.0              SETPOINT:10
LED HIGH
TIME:12:48:16.105981    CONTADOR:17     RANDOM:-8       SENO:-1.902113         SETPOINT:10
LED HIGH
TIME:12:48:17.144736    CONTADOR:18     RANDOM:4        SENO:-1.618034         SETPOINT:10
LED HIGH
TIME:12:48:18.242361    CONTADOR:19     RANDOM:6        SENO:-1.175571         SETPOINT:10
LED HIGH
TIME:12:48:19.320955    CONTADOR:20     RANDOM:-9       SENO:-0.618034         SETPOINT:10
LED HIGH
TIME:12:48:20.374066    CONTADOR:1      RANDOM:-8       SENO:0.0               SETPOINT:10
```

Fonte: Capturado pelo Autor

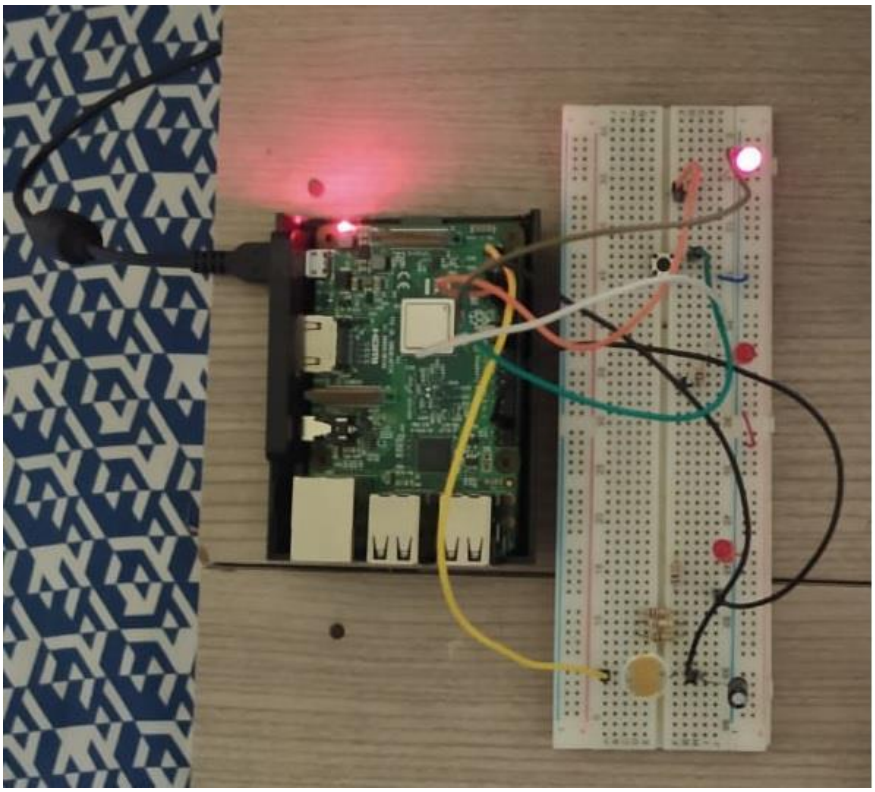
As figuras 48 e 49 apresentam a sinalização do LED com a troca dos valores. O LED que indica o valor de Set Point é o da parte superior da protoboard, conectado fisicamente no pino 10 da placa.

Figura 48 – Sinalização de Escrita



Fonte: Capturado pelo Autor

Figura 49 – Sinalização de Escrita



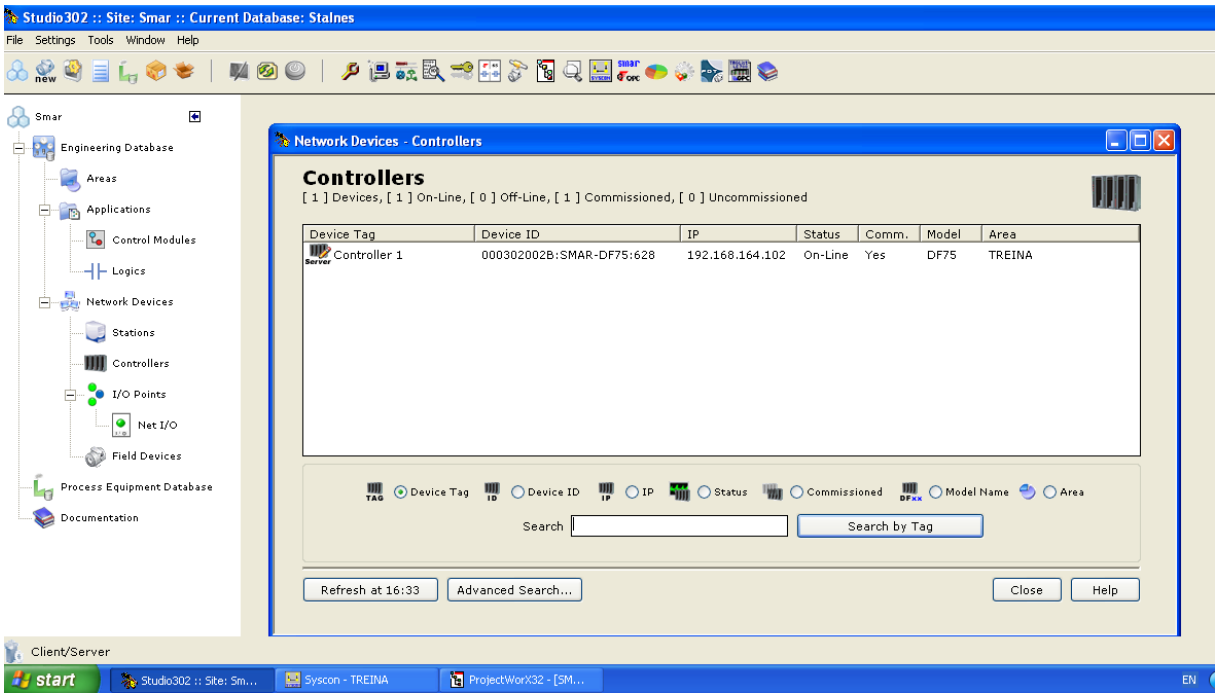
Fonte: Capturado pelo Autor

4.3 Comunicação com a Planta Smar PD3P

Como parte da etapa de testes, foi realizada uma visita ao laboratório de Controle e Automação da Unisinos para utilização da planta Smar PD3P, a fim de se verificar a comunicação entre a planta e o gateway descrito neste trabalho. Para visualização do controlador da planta foi utilizado o programa Studio302, do próprio fabricante Smar. Por motivos de restrições da rede local, tais como conflitos com o firewall, o programa teve de ser instalado em um ambiente virtual desconectado da internet, sendo utilizado para conexão na rede local um cabo Ethernet entre o computador e o *switch* do painel da planta.

A figura 50 apresenta a tela de estabelecimento da comunicação entre controlador do equipamento e o *software* Studio302, que realiza o gerenciamento e configuração do dispositivo. Na janela é exibida informações de Tag do dispositivo, número de identificação, endereço IP, modelo e Status de conexão.

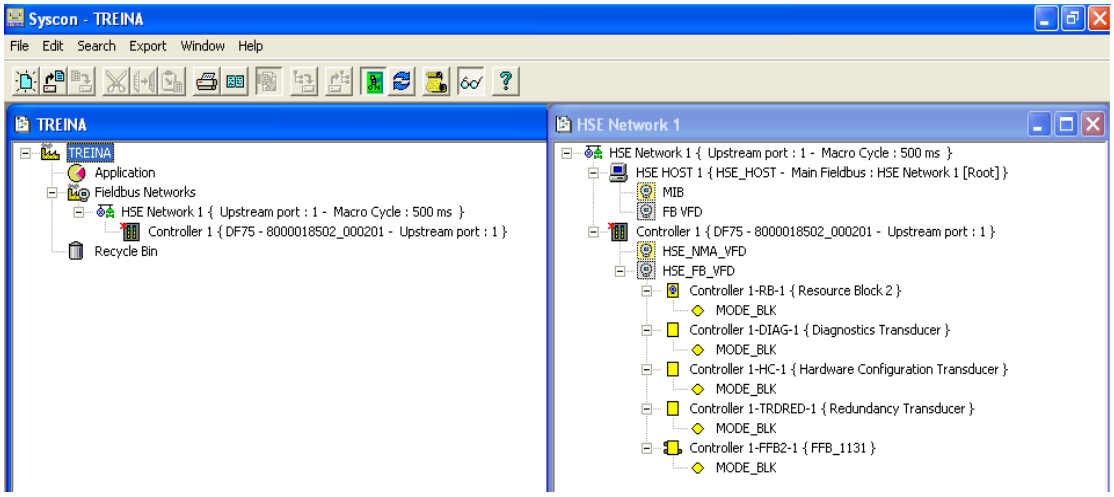
Figura 50 – Tela de Conexão Estabelecida com o Controlador no Studio 302



Fonte: Capturado pelo Autor

A figura 51 apresenta a tela do Syscon, *software* que integra o Studio302 e permite a configuração dos instrumentos Fieldbus do fabricante.

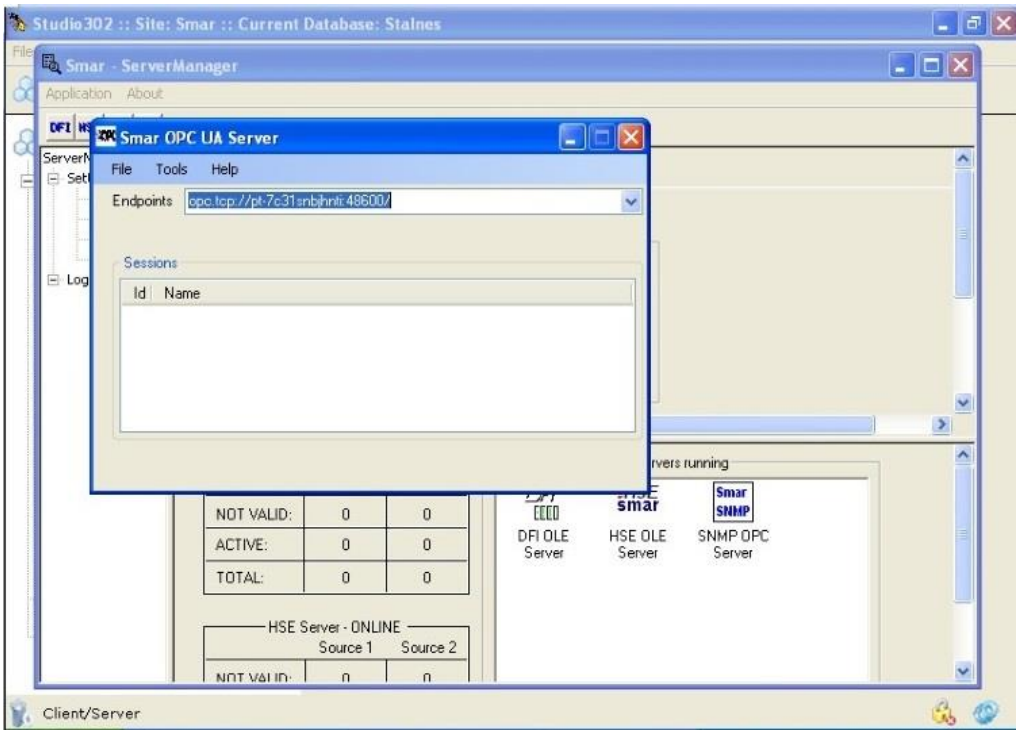
Figura 51 - Tela de Configuração dos Instrumentos SMAR



Fonte: Capturado pelo Autor

Como estratégia para comunicação entre os dispositivos da planta e o gateway, foi habilitado no Studio302 o servidor OPC UA. A figura 52 apresenta essa função habilitada e sendo executada, com o *endpoint* para comunicação sendo exibida na tela.

Figura 52 – Tela de Habilitação do OPC UA Server no Studio 302



Fonte: Capturado pelo Autor

5 CONCLUSÃO

O trabalho apresentado tratou nos capítulos iniciais de contextualizar o problema que se busca solucionar, no desenvolvimento de um gateway para acesso de dados no âmbito de automação industrial, mais especificamente em plantas que possuam recursos OPC UA. O problema foi delimitado na solução primeira de se obter dados de leitura de simuladores e componentes conectados fisicamente ao gateway, emulando dispositivos de campo de uma planta didática, e segundo na disponibilização desses dados para acesso com seus computadores pessoais de alunos, professores e pesquisadores da universidade, via cliente. Após, buscou-se realizar testes com os instrumentos que compõem a planta disponível no laboratório da UNISINOS,. Foram desenvolvidas todas as informações, baseadas em pesquisa, das tecnologias utilizadas pela planta disponível na universidade, e de alternativas para implementação do *hardware* proposto. A realização da pesquisa mostrou ser uma tarefa viável, econômica e funcional.

Os exemplos demonstrados no capítulo 2 na seção de trabalhos correlatos mostram a relevância do assunto nos dias de hoje, com o avanço das tecnologias de redes *fieldbus* cada vez mais voltadas para aplicação em conjunto da Internet 4.0 e de sistemas distribuídos. Isso faz com que a aplicação do hardware abordado nesse trabalho tenha possibilidades de escalar para uso em outros ambientes, inclusive com outros protocolos de comunicação utilizados na indústria e que não foram mencionados.

A metodologia descrita no capítulo 3 foi utilizada para o desenvolvimento do gateway dentro do prazo. Os resultados obtidos no ambiente de simulação, com *softwares* que emulam um controlador OPC UA transmitindo seus dados numa rede local, possibilitaram constatar a viabilidade do projeto do gateway para ser utilizado em conjunto da planta Smar PD3P, conforme exposto no capítulo 4. Adicionalmente, os dispositivos conectados as GPIOs do gateway exemplificaram funcionalidades que podem ser utilizadas para expandir o leque de aplicações.

De forma geral, o gateway atendeu os requisitos de coletar dados de automação via OPC e disponibilizá-los para acesso simultâneo de clientes. Não foi possível verificar com testes a leitura de instrumentos da planta Smar PD3P por conta da dificuldade na configuração dos equipamentos e dos parâmetros de rede do

laboratório. Futuramente, podem ser realizadas melhorias para o uso do gateway conforme as sugestões:

- Utilização de um circuito adicional para possibilitar a leitura de dados via barramento CAN;
- Implementação no código de ferramentas que disponibilizem os dados de leitura via Web, eliminando a restrição da rede local;
- Desenvolver uma aplicação que registre o histórico de dados lidos pelo gateway para acesso posterior.

REFERÊNCIAS

Arc Advisory Group. **What are Industrial Network Gateways?**. Disponível em <https://www.arcweb.com/blog/what-industrial-network-gateways>. Acesso em 26 de nov. 2020.

Ausberger, T; Stetina, M. **General methodology for Building OPC UA gateways. IFAC 2019**. 16th IFAC Conference on Programmable Devices and Embedded Systems PDES 2019. Disponível em < https://www.sciencedirect.com/science/article/pii/S2405896319326291>. Acesso em 16 abr. 2020.

Balbinot, A.; Brusamarello, V. **Instrumentação e Fundamentos de Medidas**. LTC, 2010. ISBN 978-85-216-1754-9.

Embarcados. **Introdução ao OPC UA**. Disponível em <https://www.embarcados.com.br/introducao-ao-opc-ua/>. Acesso em nov. 2020.

Forouzan, B. **Comunicação de Dados e Redes de Computadores**. AMGH, 2010. ISBN 978-85-63308-47-4.

Keller, A. **Internet das coisas aplicada à indústria: dispositivo para interoperabilidade de redes industriais**. Dissertação (Mestrado em Engenharia Elétrica) – UNISINOS. São Leopoldo, p. 56. 2016

Lugli, A.; Santos, M. **Redes industriais para automação industrial:** AS-I, PROFIBUS e PROFINET. Érica, 2010. 174 p. ISBN 978-85-365-0328-8.

Mahnke, W; Leitner, S; Damm, M. **OPC Linfei Architecture**. Springer, 2009. ISBN 978-12-83945-30-1

Mastilak, L.; Galinski, M.;Kotuliak, I.; Ries, M.**Improved smart gateway in IoT**. **ICETA 2018**. 16th International Conference on Emerging eLearning Technologies and Applications. Disponível em < https://www.researchgate.net/publication/329645004_Improved_Smart_Gateway_in_IoT>. Acesso em nov. 2020.

Nise, N. **Engenharia de Sistemas de Controle**. LTC, 2017. p. 1-3. ISBN 978-85-21634-35-5.

OPC FOUNDATION. **OPC UA SPECIFICATION.** Disponível em < https://opcfoundation.org/developer-tools/specifications-unified-architecture> Acesso em 10 de nov. 2020.

Prosys OPC Ltd. **OPC UA Simulation Server**. Disponível em <https://www.prosysopc.com/products/opc-ua-simulation-server/>. Acesso em 16 de abr. 2021.

Raspbian. **Welcome to Raspbian**. Disponível em <https://www.raspbian.org/>. Acesso em nov. 2020.

Smar PROFIBUS PA. **Manual dos Procedimentos de Instalação, Operação e Manutenção – Profibus PA**. Disponível em: < https://www.smar.com/pdfs/manuals/geral-pamp.pdf>. Acesso em nov. 2020.

Smar. DFI 302. **Fieldbus Universal Bridge**. Disponível em < https://www.smar.com/pdfs/manuals/dfi302mp.pdf>. Acesso em nov. 2020.

Smar. FY303. **Posicionador de Válvulas Profibus PA**. Disponível em < https://www.smar.com/pdfs/manuals/fy303mp.pdf>. Acesso em nov. 2020.

Smar. LD303. **Transmissor de Pressão Profibus PA**. Disponível em < https://www.smar.com/pdfs/manuals/ld303mp.pdf>. Acesso em 20 out. 2020.

Smar. PD3-P. **Manual de Instruções Operação e Manutenção** – **Plantas Didáticas**. Disponível em: <https://www.smar.com/pdfs/manuals/piloplpmp.pdf>. Acesso em 15 out. 2020.

Smar. PROFIBUS PA. **PROFIBUS PA**. Disponível em <https://www.smar.com/pdfs/manuals/geral-pamp.pdf>. Acesso em nov. 2020.

Smar. TT303. **Transmissor de Temperatura Profibus PA**. Disponível em < https://www.smar.com/brasil/produto/tt303-transmissor-de-temperatura-profibus-pa>. Acesso em 20 out. 2020.

UNIFIED AUTOMATION. **Motivation for the OPC UA**. Disponível em <http://documentation.unified automation.com/uasdkdotnet/2.0.0/L2OpcUaMotivation.html>. Acesso em 23 de nov. 2020.

Upton, E.; Halfacree, G. **Raspberry Pi: Guia do Usuário**. Alta Books, 2017. ISBN 978-85-508-0441-5 (Ebook).