Mayra Rafaella Galvincio Silva

Instrumentação de uma planta de manufatura flexível utilizando padrão OPC-UA embarcado

Campina Grande, PB

2023

Mayra Rafaella Galvincio Silva

Instrumentação de uma planta de manufatura flexível utilizando padrão OPC-UA embarcado

Trabalho de Conclusão de Curso (TCC) submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Graduada em Engenharia Elétrica.

Universidade Federal de Campina Grande – UFCG

Centro de Engenharia Elétrica e Informática

Departamento de Engenharia Elétrica

Orientador: Rafael Bezerra Correia Lima

Campina Grande, PB

2023

Mayra Rafaella Galvincio Silva

Instrumentação de uma planta de manufatura flexível utilizando padrão OPC-UA embarcado

Trabalho de Conclusão de Curso (TCC) submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Graduada em Engenharia Elétrica.

Trabalho aprovado. Campina Grande, PB, _____/_____/_____.

Rafael Bezerra Correia Lima
Orientador

George Acioli Júnior
Convidado

Campina Grande, PB
2023

Agradecimentos

A minha família, por terem me proporcionado uma boa educação e por me oferecerem uma rede de apoio nos momentos os quais estava muito atarefada.

Ao meu orientador, professor Rafael, pela atenção e apoio para a realização deste trabalho, e pelos ensinamentos dados nas disciplinas acadêmicas.

A minha equipe de trabalho no VIRTUS, em especial Fabiano, Fernando e Lucas, que sempre se preocuparam com o andamento do meu trabalho de conclusão de curso, além de estarem dispostos a me ajudar com minhas dúvidas e problemas sem medir esforços.

Ao meu namorado, Ronaldo, por sempre acreditar que posso conquistar meus sonhos, por estar comigo em todos os momentos, além de me ajudar a tomar as decisões mais adequadas para o desenvolvimento e escrita do trabalho.

Aos amigos da graduação, que me proporcionaram muitos momentos felizes e me acompanharam durante o andamento do curso, o que facilitou bastante a passagem por jornadas difíceis, pois estava na companhia deles.

Aos meus personagens fictícios favoritos, os quais me ensinaram a não desistir, mesmo que tudo pareça muito difícil, não há nada que não possa ser conquistado com perseverança e dedicação.

E a todos que de alguma forma puderam me ajudar no desenvolvimento deste trabalho.

Muito obrigada!

"Quando vocês acham que as pessoas morrem?
Quando elas levam um tiro de pistola bem no coração?
Não.
Quando são vencidas por uma doença incurável?
Não!
Quando bebem uma sopa de cogumelo venenoso?
Não!
Elas morrem... Quando são esquecidas."
-Doutor Hiluluk (One Piece)

Lista de ilustrações

Lista de abreviaturas e siglas

MQTT	*Message Queuing Telemetry Transport*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
HTTP	*HyperText Transfer Protocol*
OPC	*Open Platform Communications*
SCADA	*Supervisory Control and Data Acquisition*
HMI	*Human-Machine Interface*
COM	*Component Object Model*
DCOM	*Distributed Component Object Model*

Resumo

OPC-UA é um padrão de comunicação industrial que permite a transferência segura e confiável de dados entre diferentes sistemas e dispositivos em setores industriais. Ele é amplamente utilizado para monitorar e controlar processos de produção, incluindo a automação de máquinas, a monitoração de qualidade e a gestão de energia. Neste trabalho foi desenvolvido um sistema de medição para uma planta de manufatura flexível utilizando padrão OPC-UA embarcado na placa de desenvolvimento ESP32. Além disso, foi estabelecida a comunicação entre servidores OPC-UA utilizando a ferramenta *Node-RED* visando realizar a transferência de dados do sistema de medição para o controlador lógico programável (CLP) presente na planta industrial.

Palavras-chave: Industria, OPC-UA, Planta de manufatura, ESP32, Node-RED, CLP.

Abstract

OPC-UA is an industrial communication standard that enables secure and reliable data transfer between different systems and devices in industrial sectors. It is widely used to monitor and control production processes, including machine automation, quality monitoring and energy management. In this work, a measurement system for a flexible manufacturing plant was developed using the OPC-UA standard embedded in the ESP32 development board. In addition, communication between OPC-UA servers was established using the *Node-RED* tool in order to transfer data from the measurement system to the programmable logic controller (PLC) present in the industrial plant.

Keywords: Industry, OPC-UA, Manufacturing plant, ESP32, Node-RED, PLC.

Sumário

1 Introdução

A indústria 4.0 tem como principal objetivo transformar os mecanismos de manufatura de modo a trazer maior eficiência e produtividade em seus processos. Para isso, tecnologias como Internet das Coisas, *Big Data* e Computação em Nuvem constituem a base para que esse objetivo possa ser alcançado [Xu, Xu e Li 2018]. Além dessas tecnologias, padrões utilizados para estabelecer comunicação entre sistemas de controle industriais, como o OPC Clássico, também foram aprimorados [Hannelius, Salmenpera e Kuikka 2008].

A tecnologia OPC-UA (*Open Platform Communications Unified Architecture*) possibilita uma arquitetura unificada e de padrão aberto, além de trazer agilidade, segurança de dados e escalabilidade, para otimizar o padrão OPC Clássico e atender os requisitos da Indústria 4.0. Aliado a sistemas de automação industrial, essa tecnologia traz facilidade na detecção de anomalias em processos, devido a sua caraterística de interconexão inteligente entre dispositivos [Hannelius, Salmenpera e Kuikka 2008] [Souza 2022], possibilitando a coleta e a análise de dados em tempo real, além de oferecer recursos de segurança, incluindo autenticação de usuários e criptografia de dados para proteger a privacidade dos dados industriais [Mahnke, Leitner e Damm 2009].

A introdução da tecnologia OPC-UA como facilitadora da comunicação em plantas de manufatura flexíveis se tornou um elemento importante para evolução das linhas de produção, pois possibilitou que máquinas antigas e novas possam se comunicar de forma fácil e eficiente, independentemente de sua tecnologia subjacente, permitindo que empresas adicionem mais dispositivos e sistemas a medida que suas necessidades mudam, devido a sua escalabilidade. Pensando nisso, decidiu-se realizar a instrumentação de uma planta de manufatura flexível, utilizando padrão OPC-UA, para facilitar a comunicação entre os sistemas de controle envolvidos no processo.

2 Objetivos

2.1 Objetivo geral

Realizar a instrumentação de uma planta de manufatura flexível, utilizando padrão OPC-UA embarcado, com o intuito de estabelecer uma rede de comunicação entre servidores OPC-UA.

2.2 Objetivos específicos

Figuram os seguintes objetivos específicos:

- Desenvolver um sistema de estimativa da quantidade de peças em estoque, por meio de medição de distância;
- Construir um servidor OPC-UA integrado ao sistema de medição, o qual será implementado em um sistema embarcado;
- Estabelecer a comunicação entre servidor OPC-UA, presente no sistema de medição e o servidor OPC-UA, integrado à planta de manufatura flexível.

3 Referencial Teórico

3.1 Redes de Comunicação Industrial

Com o avanço da indústria 3.0, fez-se necessário a inserção de novas tecnologias que pudessem facilitar a troca de informações por meio de redes de comunicação industrial. Para isso, a indústria 4.0 trouxe novos padrões de comunicação a fim de superar as dificuldades ao lidar com o uso de cabos para conectar sensores e atuadores em certos cenários nos quais essa utilização trazia limitações [Wollschlaeger, Sauter e Jasperneite 2017]. Padrões de comunicação são definidos como descrições formais de formatos de mensagem e um conjunto de operações no meio digital [Techopedia 2020], se tornando uma parte essencial para o desenvolvimento de sistemas que lidam com hardware e software. Exemplos de protocolos universalmente utilizados são o TCP/IP, HTTP, MQTT, e o OPC. A larga utilização desses protocolos permitiu que novos horizontes tecnológicos pudessem ser alcançados, pois possibilitaram a conexão de dispositivos cabeados e não cabeados, facilitando a troca de informações em diversas aplicações como hospitalares e industriais [Al-Sarawi et al. 2017].

3.2 Protocolo TCP/IP

O TCP/IP (*Transmission Control Protocol/Internet Protocol*) é o conjunto de protocolos de comunicação utilizados para interconectar dispositivos de rede na Internet. O TCP é responsável por estabelecer uma conexão confiável entre dois ou mais dispositivos e garantir que os dados sejam transmitidos e recebidos na ordem correta. O IP é responsável por rotear pacotes de dados pela rede até seu destino. Juntos, permitem que a Internet funcione, fazendo com que os dispositivos se comuniquem entre si e ocorra a transferência de dados. O modelo TCP/IP possui quatro camadas, as quais são descritas a seguir:

- Camada de Enlace de Dados: Responsável por fornecer a interface entre o computador e a rede física. Essa camada lida com questões como endereçamento, configuração de hardware e controle de acesso à mídia.
- Camada da Rede: Responsável por rotear, endereçar e encaminhar pacotes de dados pela rede até seu destino. O *Internet Protocol* (IP) é o protocolo primário nesta camada.
- Camada de Transporte: Responsável por estabelecer uma conexão confiável entre dois dispositivos e garantir que os dados sejam transmitidos e recebidos na ordem

correta. O *Transmission Control Protocol* (TCP) e o *User Datagram Protocol* (UDP) são os protocolos primários nesta camada.

- Camada de Aplicação: Responsável pela prestação de serviços ao usuário. Protocolos como HTTP, FTP e DNS operam nessa camada.

Cada camada do modelo TCP/IP se comunica com a camada diretamente acima e abaixo dela e fornece serviços para a camada acima dela. Essa abordagem modular permite que diferentes protocolos sejam usados em cada camada e facilita o desenvolvimento de novos protocolos e a evolução da interconexão de redes [Forouzan 2002].

3.3 Padrão OPC Clássico

O OPC é um conjunto de padrões e especificações para software de automação industrial e troca de dados. O ponto de início para o desenvolvimento do padrão OPC clássico se dá pelo problema onde a comunidade científica enfrentava ao tentar acessar dados em dispositivos que já possuíam diversos outros protocolos de comunicação entre suas interfaces. Dessa forma, líderes da indústria de sistemas SCADA ou HMI, os quais enfrentavam os mesmos problemas, estavam interessados em estabelecer um padrão de comunicação para esses dispositivos. Utilizando soluções de automação da empresa Microsoft como COM e DCOM, esses líderes criaram uma abordagem de controle para processos industriais denominada OPC, do inglês, OLE for Process Control, onde OLE seria uma abreviação para *Microsoft Object Linking & Embedding*. Eis que a fundação OPC é criada, a partir desse momento, uma nova especificação OPC para dados de processamento foi criada, o OPC DA (*Data Access*) e se tornou o único padrão adotado universalmente para comunicação de dados entre sistemas de automação industrial. Outras especificações também foram criadas, como o OPC A&E (*Alarm & Events*), o qual descreve uma interface para informações baseadas em eventos, e o OPC HDA (*Historical Data Access*), o qual traz funções para acessar dados de arquivos [Mahnke, Leitner e Damm 2009].

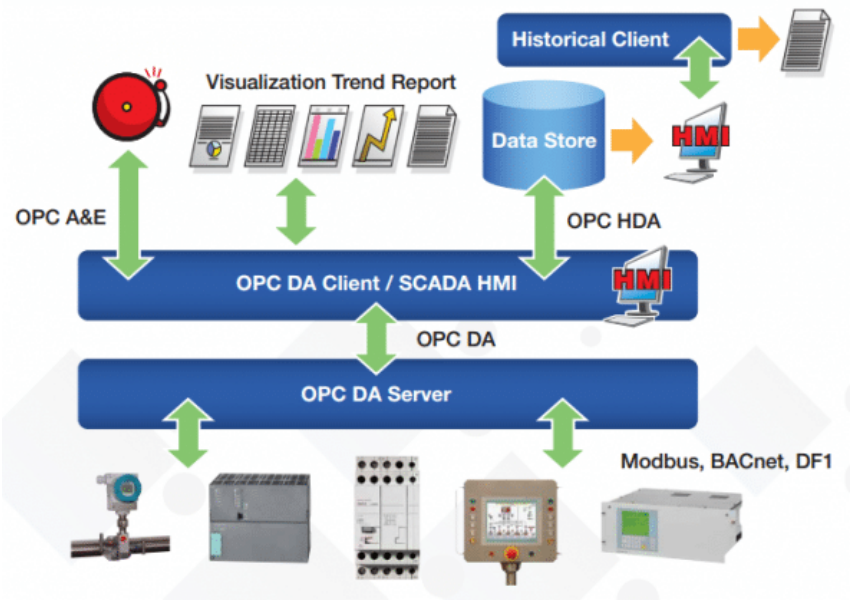

Figura 1 – Arquitetura do Servidor OPC Clássico

Fonte: [Melo 2018]

A comunicação OPC se baseia em uma arquitetura cliente-servidor, na qual o servidor possui o papel de encapsular a fonte de informações de processamento de um dispositivo e a torna disponível via interface. Já o cliente se conecta ao servidor OPC, tendo a possibilidade de acessar e consumir os dados recebidos.

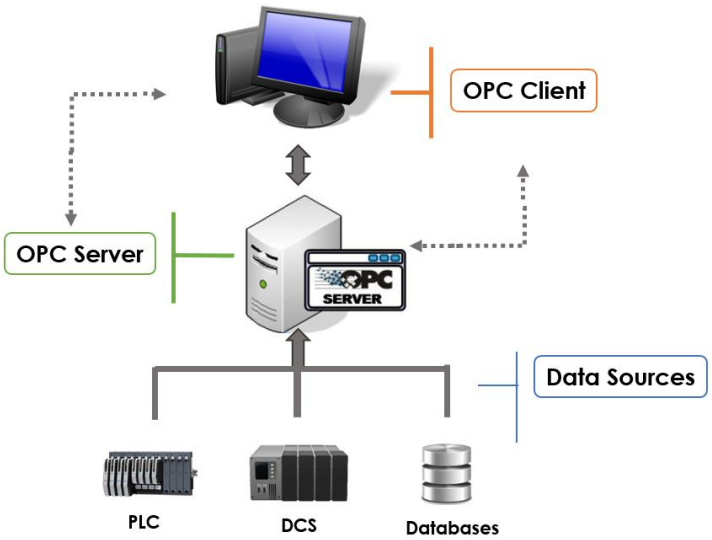

Figura 2 – Comunicação de Dados no OPC Clássico

Fonte: [OPCBlogAdmin 2018]

3.3.1 Padrão OPC UA

Ao passar do tempo, setores da indústria que utilizavam o padrão OPC Clássico estavam trabalhando com dados e sistemas cada vez mais complexos, e a dependência trazida pelo uso das tecnologias COM e DCOM, também traziam limitações em relação ao uso de plataformas proprietárias. Nesse sentido, o padrão OPC UA (*Unified Architecture*) foi criado para se tornar independente de plataforma, porém mantendo os padrões de desempenho. Outras funcionalidades também entraram como requisitos para a criação do padrão OPC UA, como a robustez e tolerância a falhas, a interoperabilidade entre sistemas de diferentes vendedores e a confiabilidade dos dados via Intranet [Mahnke, Leitner e Damm 2009].

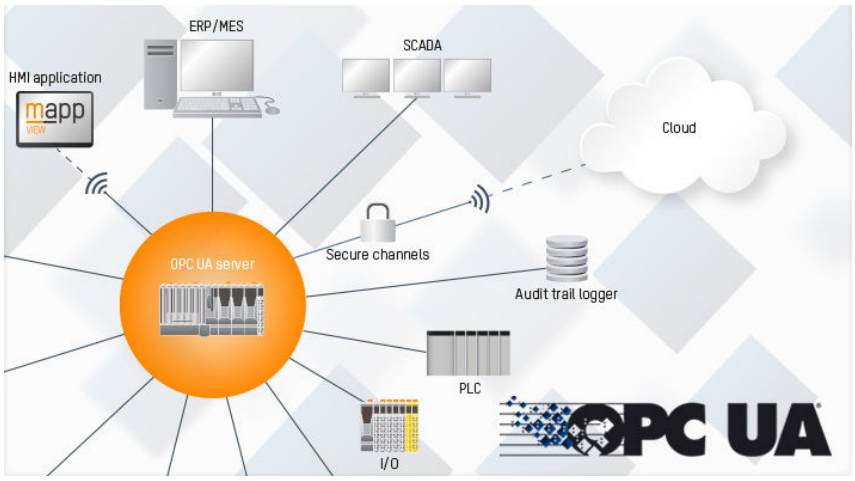

Figura 3 – OPC UA e suas aplicações

Fonte: [O que é OPC UA?]

O padrão OPC UA engloba dois protocolos de transporte diferentes: OPC TCP e SOAP/HTTP(s), que determinam o método de comunicação entre o cliente e o servidor. O OPC TCP é projetado para uso em automação industrial, controle de processos e ambientes de manufatura. Ele permite a transferência confiável e orientada à conexão de dados OPC entre clientes e servidores OPC, garantindo que os dados sejam transmitidos e recebidos na ordem correta e quaisquer pacotes ausentes ou duplicados sejam retransmitidos, fornecendo assim um canal de comunicação confiável e robusto.

3.3.2 Arquitetura OPC UA

Na arquitetura OPC UA, o servidor mantém os dados visíveis para o Cliente por meio do AddressSpace. O AddressSpace é formado por Nós (*Nodes*) que estão conectados por referência e podem ser acessados pelos clientes. Esses Nós representam Objetos, os quais representam elementos reais, como dados de sensores e atuadores, ou elementos de software. O servidor torna os Nós acessíveis para o Cliente por meio de uma Visão

(*View*) e a troca de mensagens entre Cliente e Servidor é gerenciada por meio da API presente no Servidor (OPC UA Server API), mantendo a aplicação isolada da pilha de comunicação OPC UA (OPC UA *Communication Stack*), a qual converte chamadas da API do cliente em Mensagens (*Messages*) e as envia por meio da entidade de comunicações subjacente. Além disso, o servidor também pode publicar Notificações (*Notifications*) para os Clientes por meio de uma Subscrição (*Subscription*), terminal de conexão entre a API e o Cliente [OPC Foundation 2017].

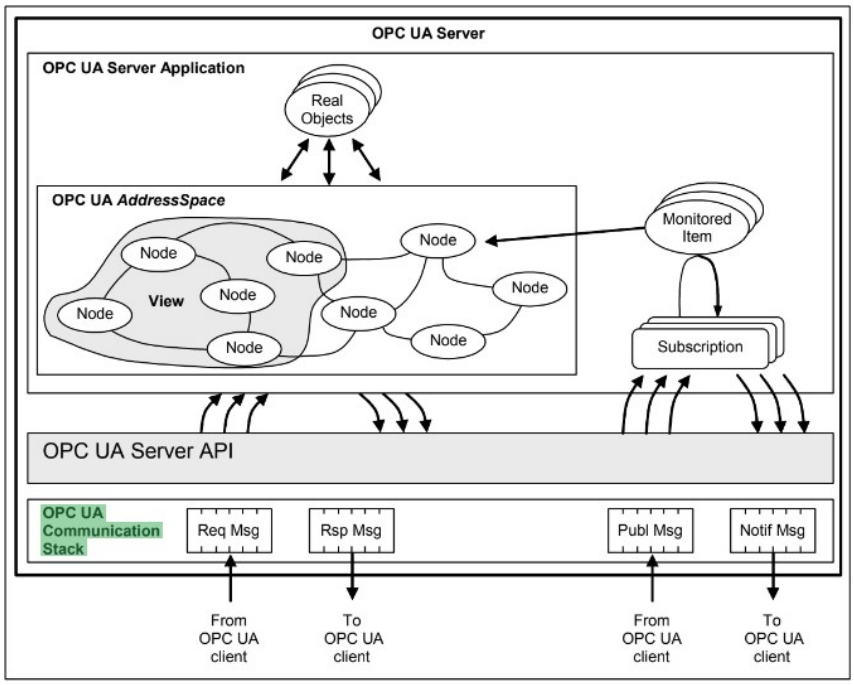

Figura 4 – Arquitetura do Servidor OPC UA.

Fonte: [OPC Foundation 2017]

3.4 Plantas de manufatura flexível

As plantas de manufatura flexível são instalações industriais projetadas para serem adaptáveis e capazes de produzir uma ampla gama de produtos usando vários processos de fabricação. Este tipo de planta é projetada a fim de alternar rápido e facilmente entre diferentes ciclos de produção e se adaptar às mudanças na demanda ou na mistura de produtos. Existem várias características-chave que definem uma planta flexível [Kaighobadi e Venkatesh 1994]:

- Modularidade: A planta é dividida em unidades ou módulos menores e independentes que podem ser facilmente reconfigurados ou substituídos para suportar diferentes processos de fabricação.

- Automação: A planta utiliza alto grau de automação, incluindo robôs, para realizar tarefas e manusear materiais. Isso permite que a planta alterne rápida e facilmente entre diferentes ciclos de produção.
- Integração: A planta é integrada com sistemas de computador que permitem monitoramento e controle em tempo real do processo de produção, bem como programação eficiente e alocação de recursos.
- Padronização: A planta usa equipamentos e processos padronizados para permitir flexibilidade e facilidade de reconfiguração.

4 Materiais e Métodos

4.1 Materiais

4.1.1 Proposta do Trabalho

É proposto o desenvolvimento de um modelo de medição para realizar a instrumentação de uma planta de manufatura flexível fabricada pela empresa alemã FESTO. O seguinte sistema de medição será composto, principalmente, por uma placa de desenvolvimento ESP32 e um sensor de distância ultrassônico HC-SR04, o qual possibilitará a determinação da quantidade de peças no estoque da planta. Os dados fornecidos pelo sistema de medição serão transmitidos para a planta FESTO por meio de OPC UA. Para realizar a comunicação entre o CLP (Controlador Lógico Programável) presente na planta FESTO e o sistema de medição, um servidor OPC-UA será implementado na placa de desenvolvimento ESP32 utilizando a biblioteca de código aberto open62541 e a ferramenta *Node-RED*, a fim de realizar operações de leitura e escrita nos servidores OPC-UA envolvidos.

4.1.2 Biblioteca Open62541

A biblioteca open62541 é uma implementação de código aberto do protocolo OPC UA a qual fornece um conjunto de APIs e ferramentas para implementar servidores e clientes OPC UA. Ela é escrita na linguagem de programação C e projetada para ser tátil, eficiente e fácil de usar. Alguns dos recursos da biblioteca open62541 incluem:

- Suporte para todos os tipos de dados OPC UA e regras de codificação
- Manipulação do lado do cliente e do servidor do protocolo binário OPC UA
- Comunicação segura com mensagens criptografadas
- Ferramentas para gerar e analisar mensagens OPC UA codificadas em XML e ASN
- Camada de rede intercambiável (plug-in) para usar APIs de rede personalizadas

A biblioteca está disponível gratuitamente no GitHub sob a licença de código aberto Mozilla Public License v2, além de possuir uma comunidade de desenvolvedores voluntários responsáveis por relatar erros e melhorar a documentação do código [open62541].

4.1.3 Código Base — GitHub

4.1.3.1 opcua-esp32

O repositório *opcua-esp32* consiste na implementação de um servidor OPC-UA em uma placa de desenvolvimento ESP32 utilizando a biblioteca de código aberto open62541.

4.1.3.2 HC-SR04-Ultrasonic-Sensor-with-ESP32-ESP-IDF

O projeto *HC-SR04-Ultrasonic-Sensor-with-ESP32-ESP-IDF* permite realizar a interface do sensor ultrassônico HC-SR04 com a placa ESP32 utilizando ESP-IDF.

4.1.4 UAExpert

O UAExpert é um software multiplataforma que atua como um cliente de teste OPC UA, permitindo que o mesmo se conecte a um servidor OPC UA e solicite dados a ele. Além disso, também é possível enviar comandos e dados para o servidor, fazendo com que ele interaja e controle dispositivos.

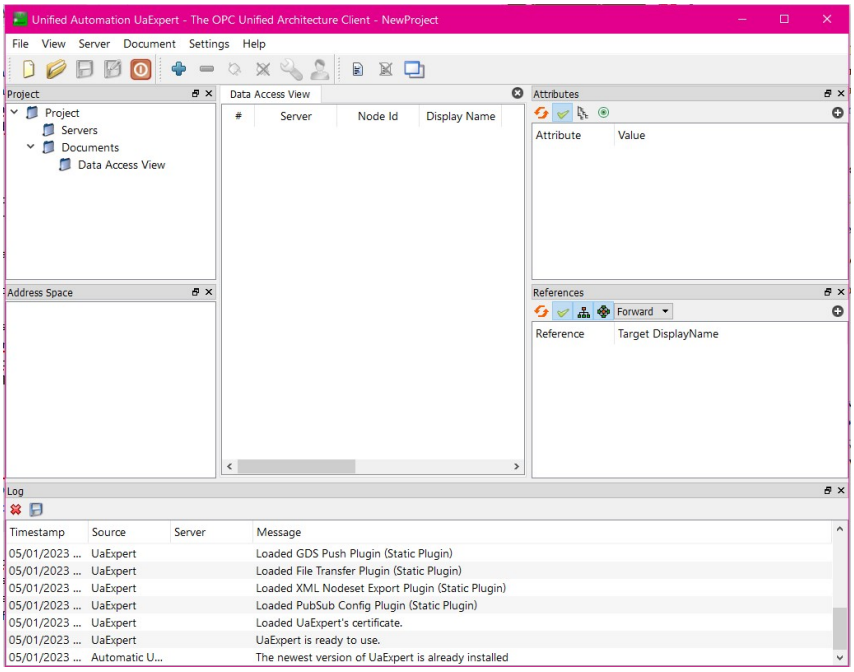

Figura 5 – Interface do UA Expert

Fonte: Autoria própria

O software é frequentemente utilizado em aplicações industriais e comerciais, como controle de processos, automação predial e análise de dados e oferece uma variedade de recursos e ferramentas para trabalhar com servidores e dados OPC UA. Alguns recursos disponíveis incluem [UaExpert—A Full-Featured OPC UA Client]:

- Conectividade com vários servidores OPC UA

- Ferramentas de visualização e análise de dados
- Interface personalizável e opções de exibição
- Suporte para diferentes tipos de dados e modelos de dados OPC UA
- Recursos de segurança, como criptografia e autenticação
- Recursos para execução de *scripts* e automatização de tarefas

4.1.5 CODESYS *provided by Festo*

O CODESYS é um ambiente de desenvolvimento de software para programação de controladores lógicos programáveis (CLPs) amplamente utilizado em sistemas de automação industrial. O CODESYS *provided by Festo* é baseado no padrão IEC 61131-3, padrão internacional para programação de sistemas de controle industrial. É compatível com uma ampla gama de PLCs de diferentes fornecedores e suporta vários protocolos de comunicação, incluindo OPC, Modbus e Profinet, o que o torna muito flexível.

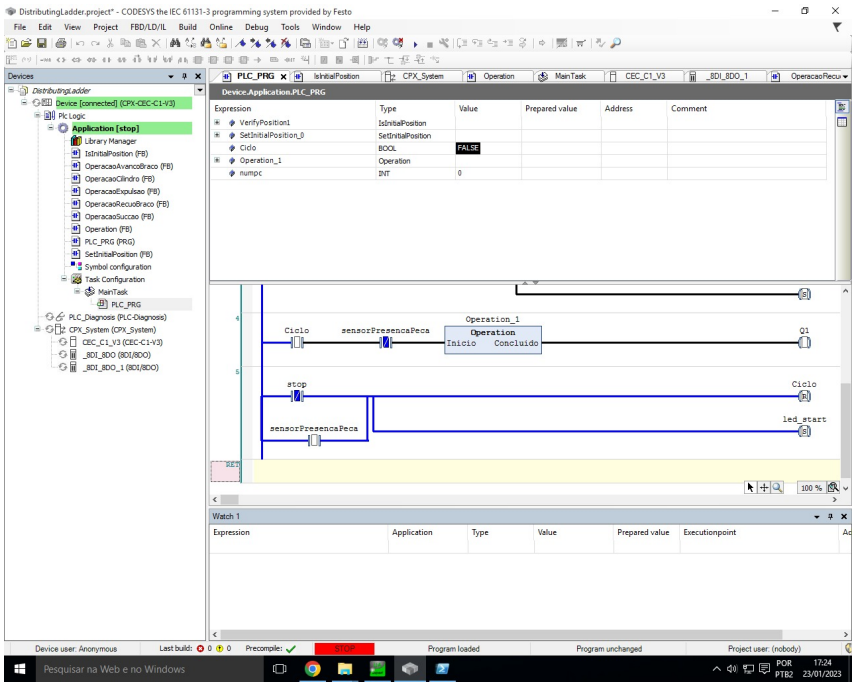

Figura 6 – Interface do CODESYS *provided by Festo*

Fonte: Autoria própria

Além disso, fornece uma biblioteca de blocos de função e bibliotecas para diferentes tipos de dispositivos e protocolos, o que torna mais fácil para os desenvolvedores integrarem diferentes sistemas e dispositivos. A FESTO também fornece bibliotecas e blocos de função adicionais para seus próprios produtos, facilitando a integração das soluções de automação da FESTO nos sistemas desenvolvidos com CODESYS [CODESYS® provided by Festo].

4.1.6 Node-RED

O Node-RED é uma ferramenta de programação visual que permite aos desenvolvedores conectar dispositivos, APIs e serviços online em uma interface intuitiva de arrastar e soltar. Construído em Node.js, facilita a criação de aplicativos orientados a eventos para tarefas como processamento de dados, visualização e integração com outros sistemas. A interface baseada na web permite a criação e gerenciamento de fluxos, que consistem em nós interconectados que representam funções ou operações específicas, como leitura de dados de sensores, processamento de dados ou envio de dados para serviços em nuvem. O fluxo de dados através do sistema pode ser facilmente modificado ou reconectado conforme necessário.

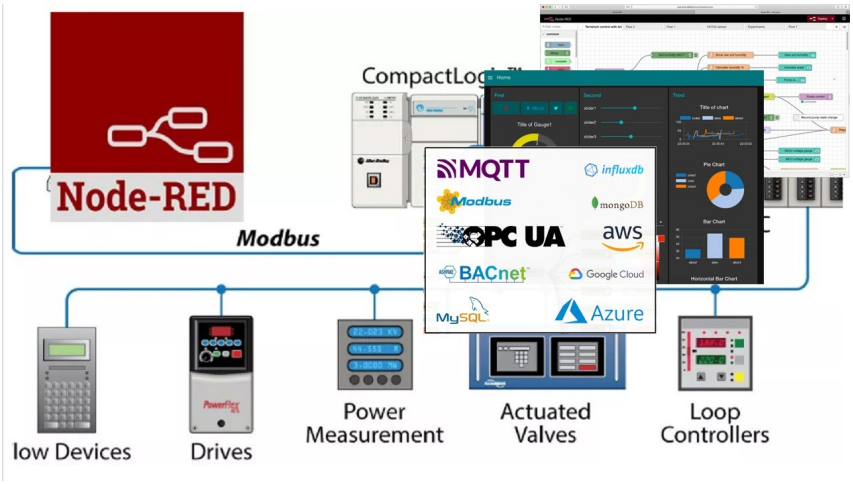

Figura 7 – Interface do Node-RED

Fonte: [Jabalquinto 2018]

Além disso, o Node-RED oferece uma vasta biblioteca de nós pré-construídos para vários dispositivos e serviços, incluindo plataformas IoT populares como MQTT, HTTP e TCP, simplificando a integração de diferentes sistemas e dispositivos e acelerando o processo de criação e prototipagem de aplicativos IoT [Contributors].

4.1.7 FESTO MPS

FESTO MPS (*Modular Production System*) é um sistema de automação modular e flexível projetado para uso nas indústrias de manufatura e montagem composto por componentes e módulos que podem ser configurados para atender às necessidades específicas de uma determinada aplicação. O sistema inclui atuadores pneumáticos e elétricos, sensores, controladores e soluções de software que podem ser combinados para criar soluções de automação personalizadas.

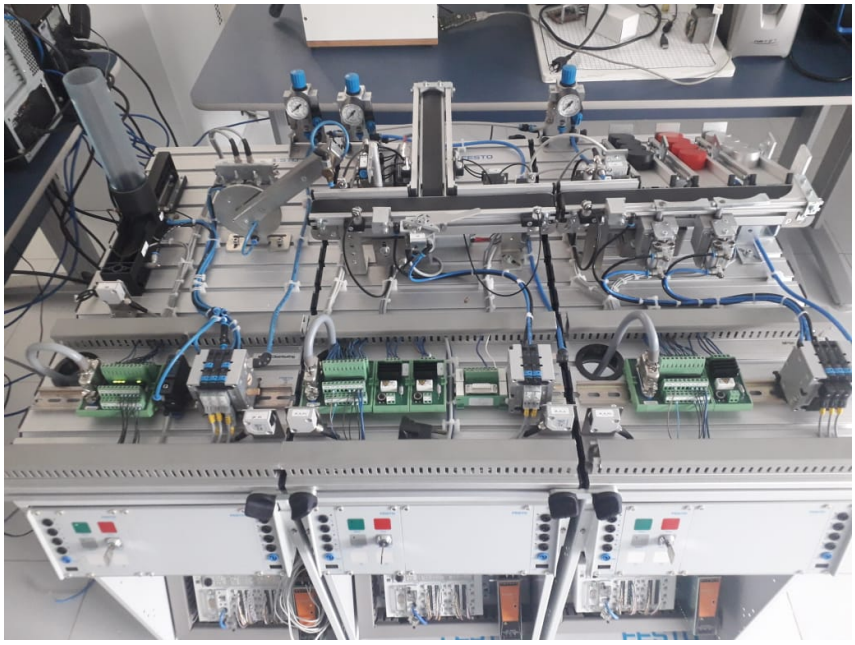

Figura 8 – Planta Industrial FESTO MPS

Fonte: Autoria própria

O MPS é baseado no princípio de modularidade e escalabilidade, facilitando a adaptação às mudanças nos requisitos de produção. Dessa forma, o sistema pode ser facilmente expandido ou modificado, e novos componentes podem ser adicionados ou substituídos conforme necessário, além de fornecer um alto grau de flexibilidade e adaptabilidade, o que permite que seja usado em uma ampla gama de aplicações e indústrias, como montagem, embalagem e manuseio de materiais. O FESTO MPS oferece uma variedade de vantagens, como fácil comissionamento, teste e depuração de aplicações de controle, fácil integração das soluções de automação da FESTO nos sistemas desenvolvidos com CODESYS e uma interface amigável para criar e editar lógica ladder, diagramas de blocos de função, e texto estruturado [MPS® The Modular Production System]. Além disso, é composto por três módulos, sendo estes o módulo de distribuição, responsável por fornecer peças ao módulo subsequente, o módulo de separação, que permite identificar a posição de uma peça e determinar seus caminhos distintos ao longo das esteiras e o módulo de classificação, que permite identificar os tipos de peças e separá-los conforme as necessidades.

4.1.8 ESP-32

A ESP32 é uma placa de desenvolvimento que integra o ESP32 SoC (*System on Chip*) desenvolvida pela empresa Espressif Systems, a qual possui WiFi e Bluetooth integrado, utilizada em diversas aplicações de Internet das Coisas (IoT) por se tratar de uma opção de baixo custo, mas que traz versatilidade e robustez para diversas aplicações.

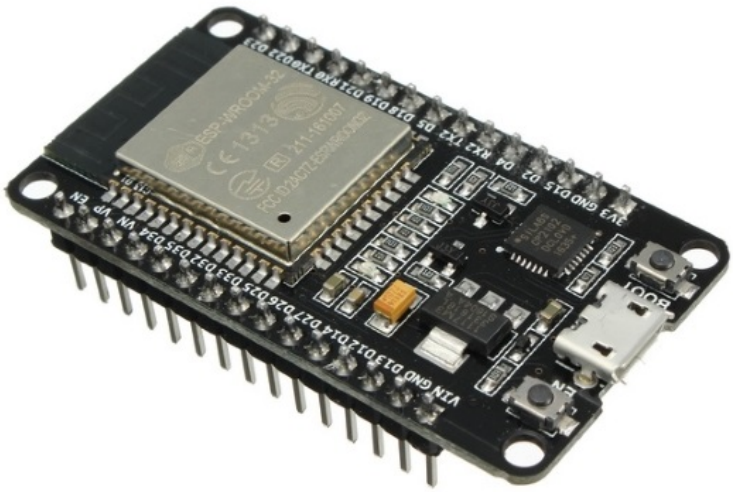

Figura 9 – Representação da Placa de Desenvolvimento ESP32.

Fonte: [Módulo WiFi ESP32 Bluetooth]

Além disso, a placa disponibiliza diversas portas programáveis (GPIO), memória RAM estática de 520k bytes, RTC (Real Time Clock) com 16Kb de SRAM, aceleradores de hardware (criptografia) como AES, SHA, RSA e ECC, entre outros periféricos [Espressif 2012]. O diagrama de blocos do chip ESP32 é ilustrado na Figura 10.

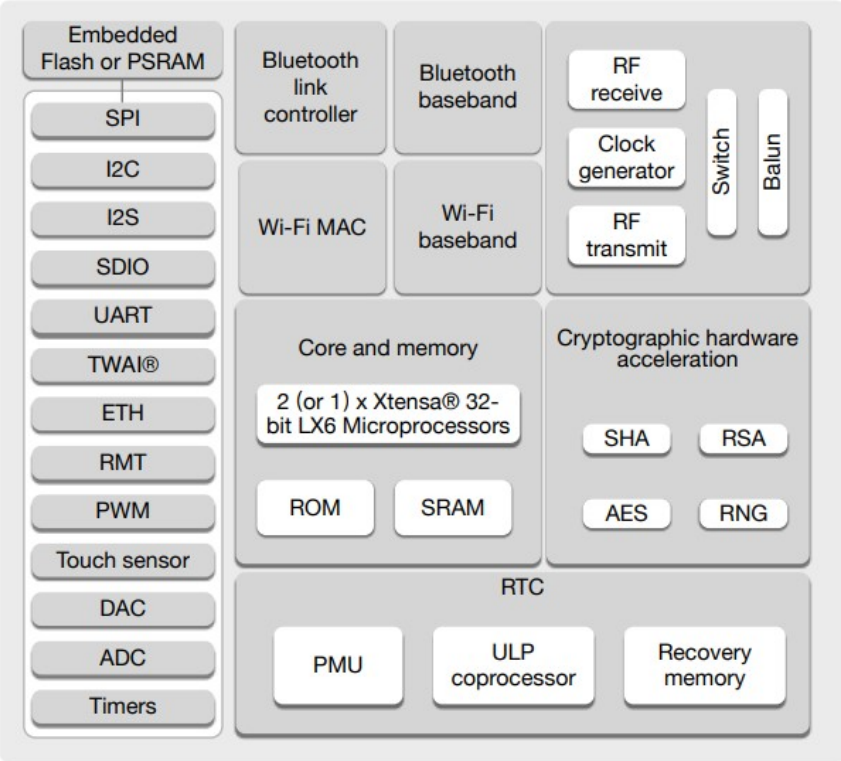

Figura 10 – Diagrama de Blocos do ESP32.

Fonte: [Espressif 2012]

4.1.9 Componentes Eletrônicos

4.1.9.1 Sensor de Distância Ultrassônico HC-SR04

O HC-SR04 é um tipo de sensor ultrassônico que pode ser usado para medição de distância. Ele funciona emitindo um pulso curto de som ultrassônico e medindo o tempo que leva para o som retornar após atingir um objeto. A distância até o objeto pode então ser calculada com base no tempo que o som levou para retornar e na velocidade do som no ambiente. É comumente utilizado em uma ampla gama de aplicações, incluindo detecção de objetos e prevenção de colisões.

Figura 11 – Sensor Ultrassônico HC-SR04

Fonte: [Módulo Sensor de Distância Ultrassônico HC-SR04]

Seu funcionamento é baseado em dois componentes principais: um transmissor e um receptor. O transmissor envia um pulso de som ultrassônico e o receptor escuta o eco de retorno. Ao medir o tempo que leva para o som viajar até o objeto e voltar, o sensor pode calcular a distância até o objeto [Thomsen 2011].

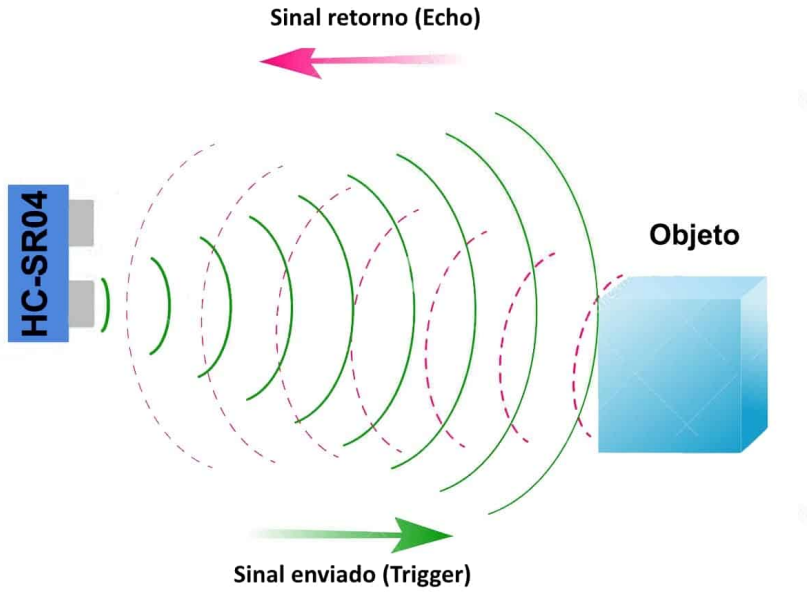

Figura 12 – Funcionamento do Sensor Ultrassônico HC-SR04

Fonte: [Thomsen 2011]

A equação utilizada para determinar a distância medida é destacada a seguir, sendo T_{high} o intervalo de tempo no nível *HIGH* e V_{som} a velocidade do som (340[m/s]) [Ultrasonic Ranging Module HC-SR04].

$$Distancia = \frac{T_{high} * V_{som}}{2} \tag{4.1}$$

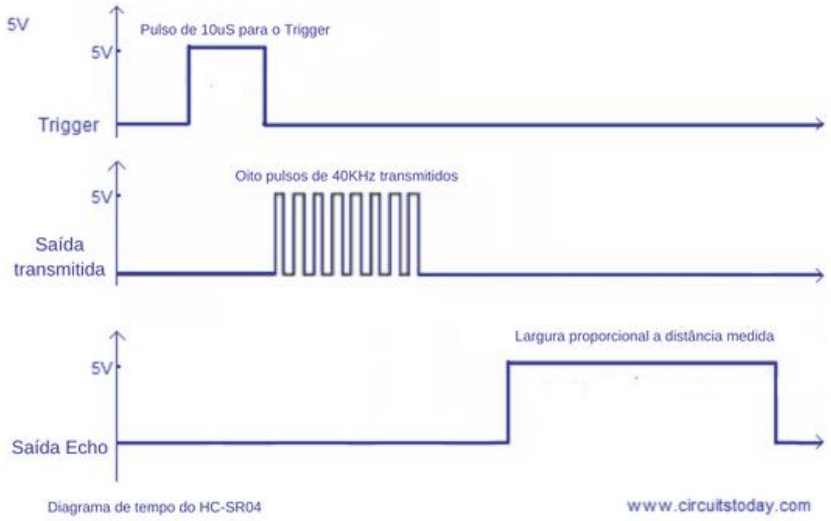

Figura 13 – Diagrama de tempo do HC-SR04

Fonte: Adaptado de [HC-SR04 Timing diagram]

4.2 Métodos

4.2.1 Desenvolvimento do servidor OPC-UA na ESP32

Inicialmente, para implementação na placa de desenvolvimento ESP32, foi realizada a instalação da ESP-IDF, framework oficial da *Espressif* para o desenvolvimento de aplicações nas placas ESP32, no editor de código-texto *Visual Studio Code*.

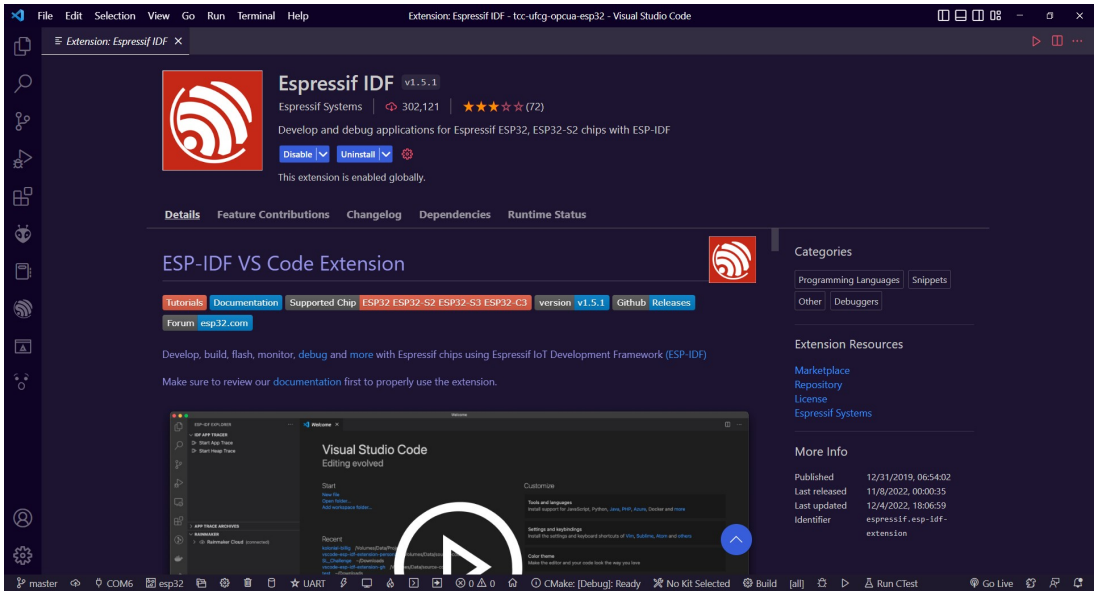

Figura 14 – Extensão ESP-IDF para Visual Studio Code

Fonte: Autoria própria

Após realizadas as configurações necessárias para desenvolvimento na ESP32, o projeto referente ao código base *opcua-esp32* foi carregado na placa e adicionada uma nova funcionalidade OPC-UA referente a variável de distância, a qual será preenchida com os valores mensurados pelo sensor de distância HC-SR04. Para isso, foram modificados os arquivos *opcua_esp32.c* localizados no diretório *main*, bem como os arquivos *model.c* e *model.h* localizado no diretório *components*.

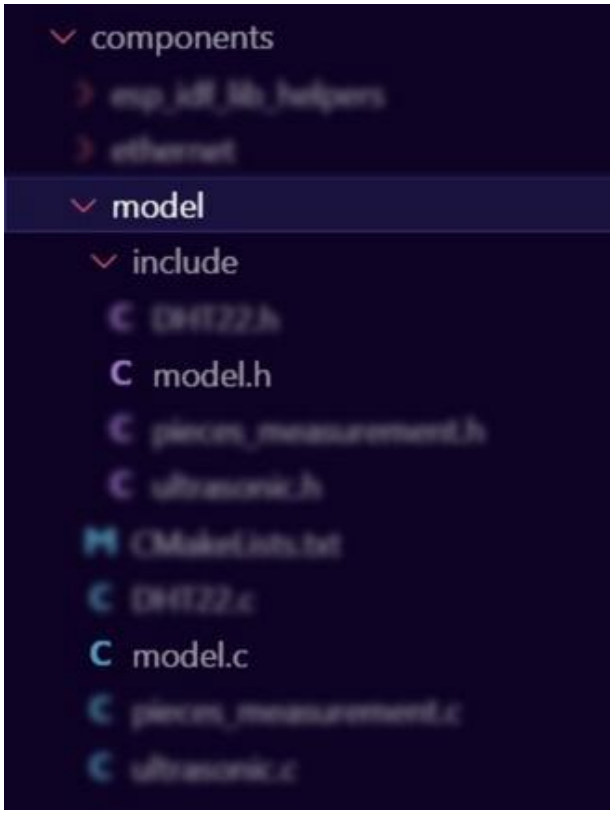

Figura 15 – Arquivos do diretório *components*

Fonte: Autoria própria

Figura 16 – Arquivo do diretório *main*

Fonte: Autoria própria

A alteração desses arquivos foi necessária para a implementação das rotinas de construção de um Nó OPC-UA. Após a construção do Nó, foram utilizadas funções do código base *HC-SR04-Ultrasonic-Sensor-with-ESP32-ESP-IDF* para adquirir os valores de distância recebidos no sensor HC-SR04. Essas funções foram adicionadas ao Nó OPC-UA referente a distância, de modo que ele pudesse realizar a sua exibição. As Figuras 17, 18 e 19 ilustram a funcionalidade OPC-UA implementada.

```
73  /* Distance*/
74
75  UA_StatusCode
76  readCurrentDistance(UA_Server *server,
77                      const UA_NodeId *sessionId, void *sessionContext,
78                      const UA_NodeId *nodeId, void *nodeContext,
79                      UA_Boolean sourceTimeStamp, const UA_NumericRange *range,
80                      UA_DataValue *dataValue);
81
82
83  void
84  addCurrentDistanceDataSourceVariable(UA_Server *server);
85
```

Figura 17 – Funcionalidade da distância implementada no arquivo *model.h*

Fonte: Autoria própria

```c
/* Distance */

ultrasonic_sensor_t ultrassonic_sensor = {
            .trigger_pin = TRIGGER_GPIO,
            .echo_pin = ECHO_GPIO
        };

UA_StatusCode
readCurrentDistance(UA_Server *server,
                    const UA_NodeId *sessionId, void *sessionContext,
                    const UA_NodeId *nodeId, void *nodeContext,
                    UA_Boolean sourceTimeStamp, const UA_NumericRange *range,
                    UA_DataValue *dataValue) {

    UA_Float ua_distance;
    float distance;
    esp_err_t res;

    ultrasonic_init(&ultrassonic_sensor);

    res = ultrasonic_measure(&ultrassonic_sensor, MAX_DISTANCE_CM, &distance);

    if (res != ESP_OK){
        printf("Error %d: ", res);
        switch (res){
            case ESP_ERR_ULTRASONIC_PING:
                printf("Cannot ping (device is in invalid state)\n");
                ua_distance = 0;
                break;
            case ESP_ERR_ULTRASONIC_PING_TIMEOUT:
                printf("Ping timeout (no device found)\n");
                ua_distance = 0;
                break;
            case ESP_ERR_ULTRASONIC_ECHO_TIMEOUT:
                printf("Echo timeout (i.e. distance too big)\n");
                ua_distance = 0;
                break;
            default:
                printf("%s\n", esp_err_to_name(res));
        }
    } else {
        ua_distance = distance*100; //distance in cm
    }

    UA_Variant_setScalarCopy(&dataValue->value, &ua_distance,
                             &UA_TYPES[UA_TYPES_FLOAT]);
    dataValue->hasValue = true;
    return UA_STATUSCODE_GOOD;
}
```

Figura 18 – Funcionalidade da distância implementada no arquivo *model.c*

Fonte: Autoria própria

```
117     /* Add Information Model Objects Here */
118     // addLEDMethod(server);
119     addCurrentTemperatureDataSourceVariable(server);
120     addRelay0ControlNode(server);
121     addRelay1ControlNode(server);
122     addCurrentDistanceDataSourceVariable(server);
123     addCurrentPiecesQuantityDataSourceVariable(server);
```

Figura 19 – Funcionalidade da distância implementada no arquivo *opcua_esp32.c*

Fonte: Autoria própria

As funcionalidades implementadas estão encapsuladas em uma *thread* OPC UA. A função *opcua_task* inicia o servidor OPC UA para receber os Nós, bem como realiza inicialização da pilha TCP/IP. Além disso, verifica o valor restante na pilha após a execução da task OPC UA. Finalizada a implementação, o código base modificado foi carregado na placa. Ao acessar o monitor serial no *Visual Studio Code*, foi possível obter o dado do IP necessário para acesso do cliente ao servidor OPC-UA, como ilustra a Figura 20.

```
I (22982) SNTP: Current time: 1970-01-01 00:00:22
I (22982) MEMORY: Heap size after OPC UA Task : 206724
I (22982) esp_netif_handlers: sta ip: 192.168.2.100, mask: 255.255.255.0, gw: 192.169.2.1
I (22992) online_connection: Got IP event!
```

Figura 20 – IP do servidor exibido no monitor serial

Fonte: Autoria própria

Para realização da consulta dos valores enviados pelo Servidor OPC-UA, foi utilizado o software *UAExpert* para conexão com o servidor OPC-UA. A Figura 22 ilustra a interface do *UAExpert*, bem como o Nó OPC-UA referente a distância.

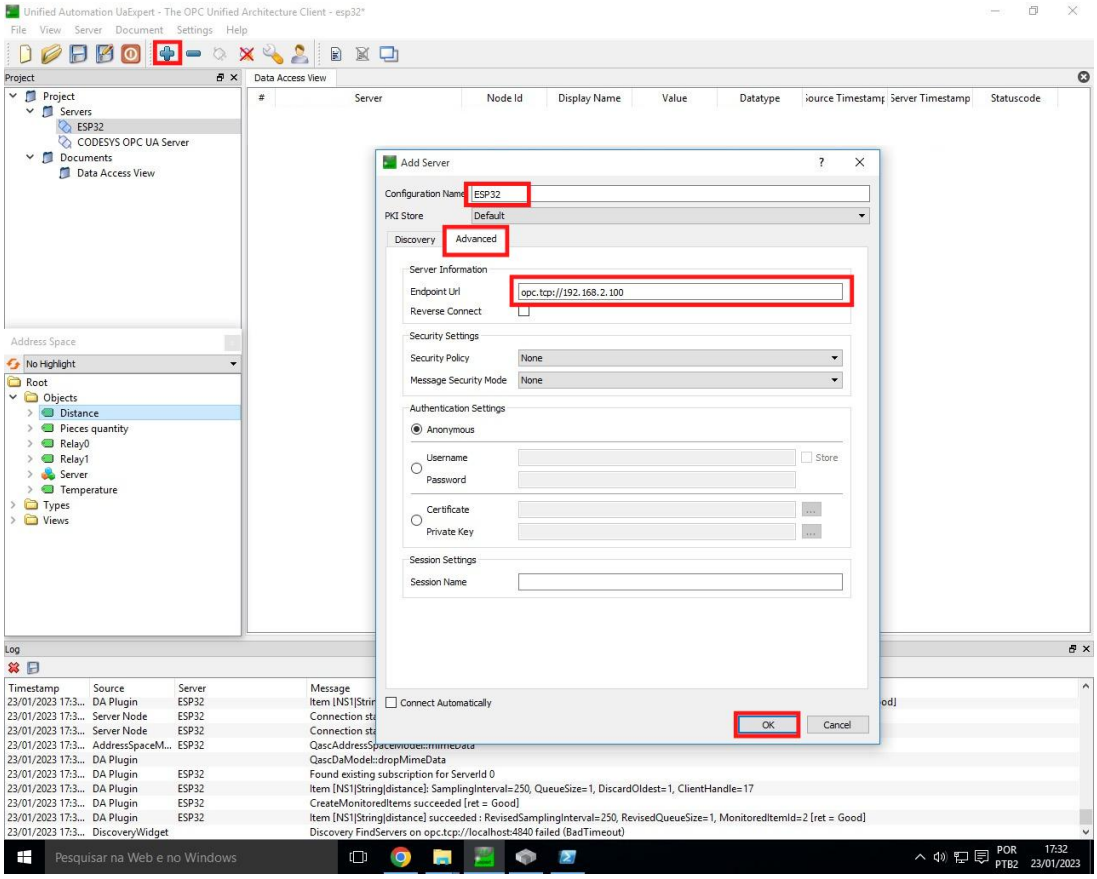

Figura 21 – Configuração para adicionar o servidor OPC-UA da placa ESP-32 na interface do *UAExpert*

Fonte: Autoria própria

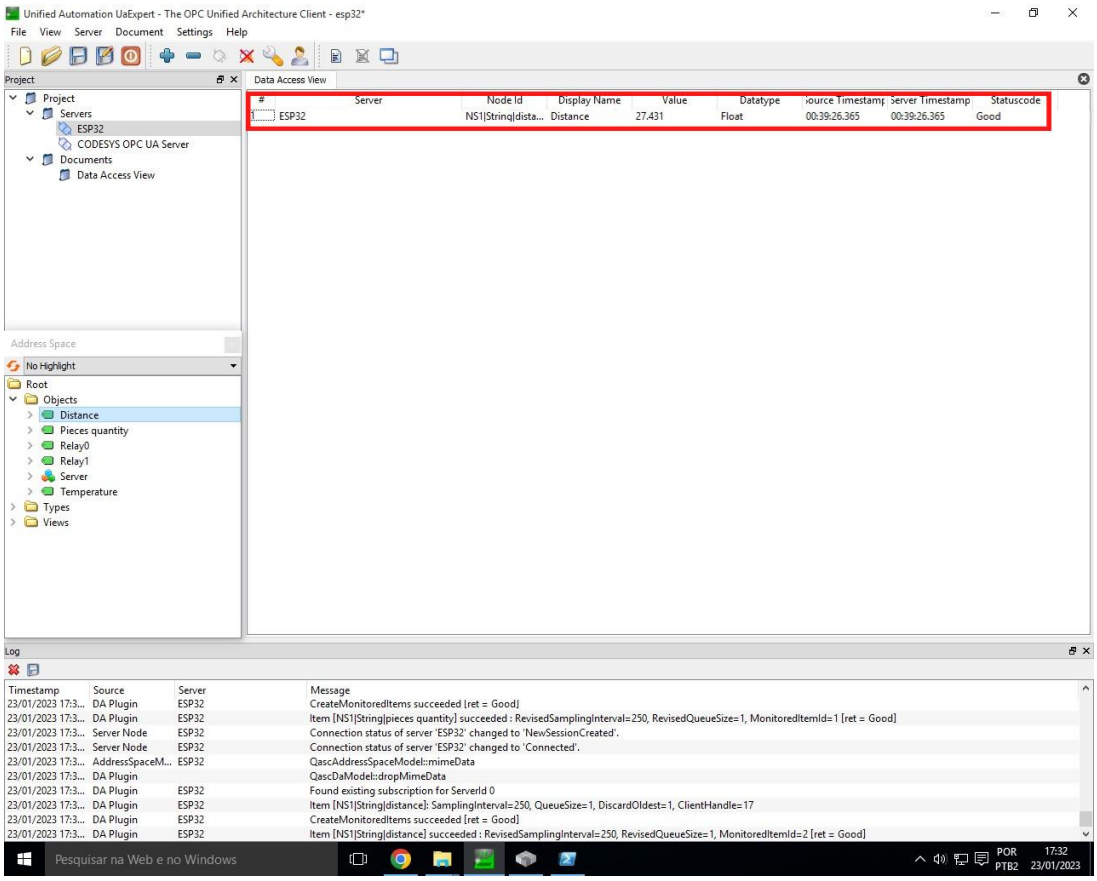

Figura 22 – Nó OPC-UA referente a variável de distância

Fonte: Autoria própria

4.2.2 Construção da plataforma base para o sensor de distância

Para determinar a quantidade de peças no estoque da planta industrial *FESTO MPS* foi construída uma estrutura base para inserção do sensor de distância HC-SR04. A estrutura foi construída com placas de isopor, as quais se encontram suspensas ao recipiente de inserção das peças.

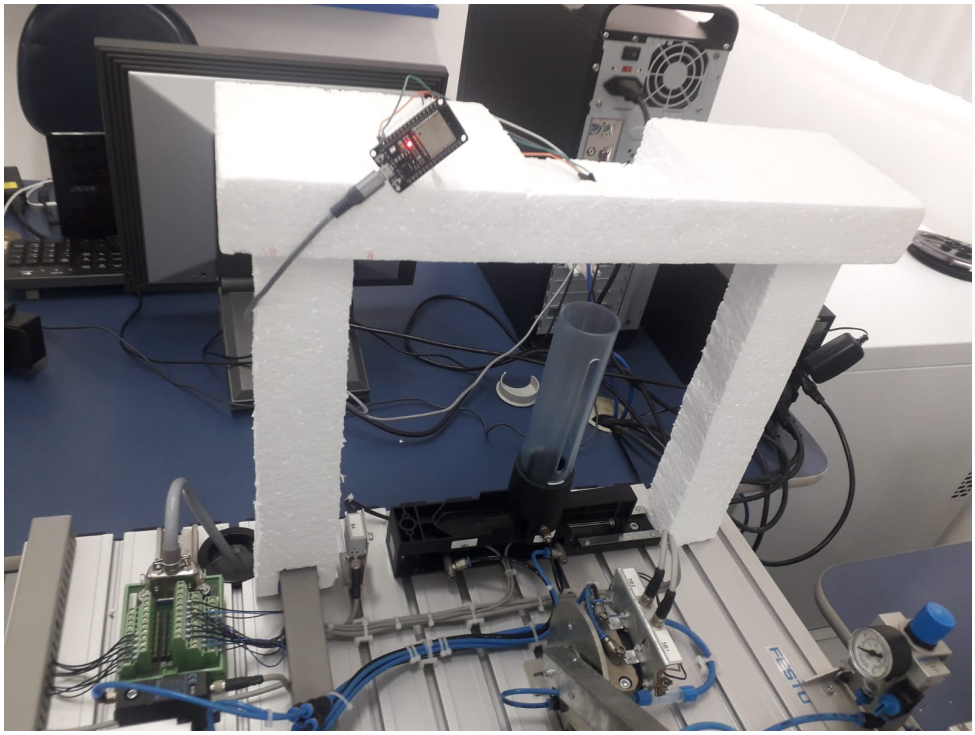

Figura 23 – Estrutura base para o sensor de distância

Fonte: Autoria própria

Foram adicionados dois furos à placa de isopor superior, de forma que o sensor HC-SR04 pudesse ser encaixado, como ilustra a Figura 24.

Figura 24 – Furos realizados na estrutura base

Fonte: Autoria própria

A montagem do circuito do sensor de distância HC-SR04 conectado a placa ESP32 é ilustrada na Figura 25. Os pinos *VCC*, *Trig*, *Echo* e *GND* do sensor foram conectados as entradas *Vcc*, *D18*, *D19* e *GND* da placa ESP32, respectivamente.

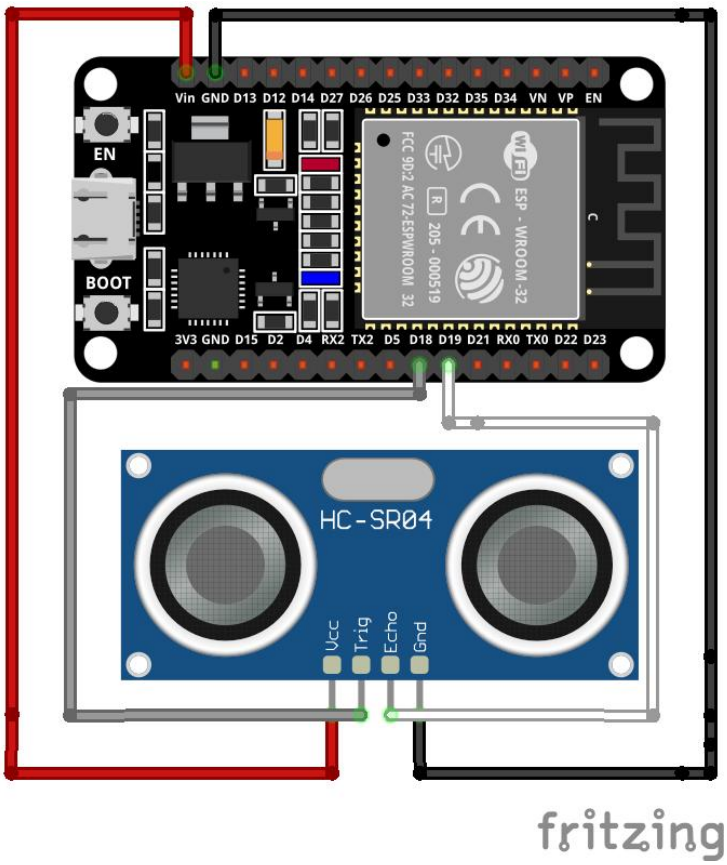

Figura 25 – Montagem do circuito do sensor de distância HC-SR04

Fonte: Autoria própria

Após a construção da plataforma base, foi realizada a implementação do código para determinação da quantidade de peças. Para isso, foram adicionados os arquivos *pieces_measurement.c* e *pieces_measurement.h* no diretório *components* presente no código base.

```h
#ifndef __PIECES_MEASUREMENT_H__
#define __PIECES_MEASUREMENT_H__

#define EMPTY_STOCK 31.9
#define FULL_STOCK 7.14
#define PIECE_SIZE 2.3

int pieces_quantity(float *distance);

#endif
```

Figura 26 – Funcionalidade implementada no arquivo *pieces_measurement.h*

Fonte: Autoria própria

```c
#include <stdio.h>
#include <math.h>
#include "pieces_measurement.h"

int pieces_quantity(float *distance){

    int quantity = 0;
    float distance_cm = (*distance)*100;

    if(distance_cm <= EMPTY_STOCK && distance_cm >= FULL_STOCK){
        quantity = round((EMPTY_STOCK - distance_cm)/PIECE_SIZE);
    }else{
        printf("OUT OF RANGE\n");
    }
    return quantity;
}
```

Figura 27 – Funcionalidade implementada no arquivo *pieces_measurement.c*

Fonte: Autoria própria

Com base na distância total medida quando o recipiente está vazio, quando está com o estoque completo, bem como a altura total da peça inserida, foi construído a rotina de definição da quantidade de peças. Caso ela esteja entre um limiar da distância total do estoque vazio, e da redução da distância de uma peça, é exibida a quantidade de peças presentes no estoque. A Figura 28 ilustra o recipiente vazio, e a Figura 29 ilustra o recipiente completo.

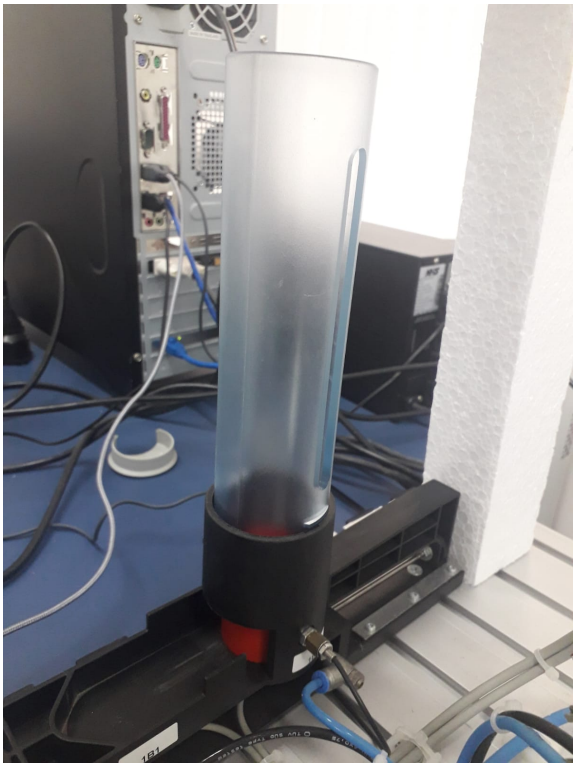

Figura 28 – Recipiente de peças vazio

Fonte: Autoria própria

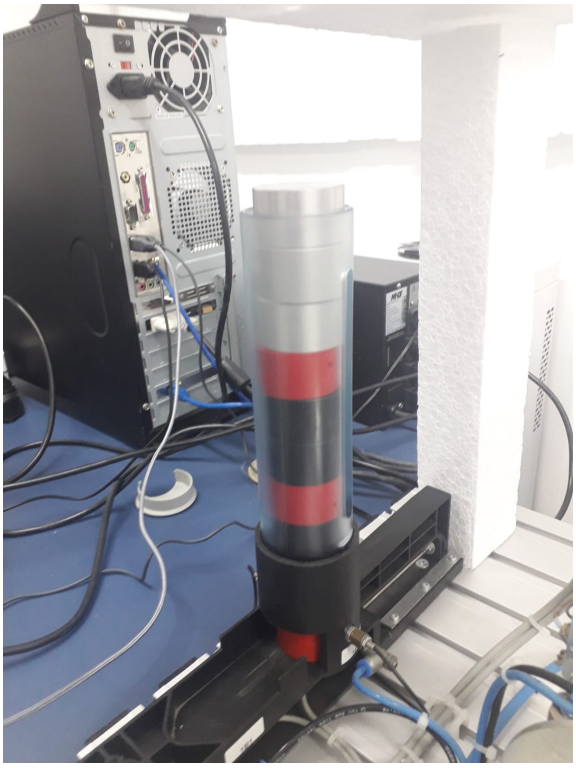

Figura 29 – Recipiente de peças completo

Fonte: Autoria própria

Para realizar a transmissão do dado da quantidade de peças para o cliente OPC-UA, foi implementado um novo Nó ao código base, alterando os arquivos *opcua_esp32.c*, *model.c* e *model.h*, como ilustram as Figura 26, 27 e 32.

```
86  /* Pieces Quantity */
87
88  UA_StatusCode
89  readCurrentPiecesQuantity(UA_Server *server,
90                  const UA_NodeId *sessionId, void *sessionContext,
91                  const UA_NodeId *nodeId, void *nodeContext,
92                  UA_Boolean sourceTimeStamp, const UA_NumericRange *range,
93                  UA_DataValue *dataValue);
94
95  void
96  addCurrentPiecesQuantityDataSourceVariable(UA_Server *server);
97
```

Figura 30 – Funcionalidade da quantidade de peças implementada no arquivo *model.h*

Fonte: Autoria própria

```
308 /* Pieces Quantity */
309
310 UA_StatusCode
311 readCurrentPiecesQuantity(UA_Server *server,
312                 const UA_NodeId *sessionId, void *sessionContext,
313                 const UA_NodeId *nodeId, void *nodeContext,
314                 UA_Boolean sourceTimeStamp, const UA_NumericRange *range,
315                 UA_DataValue *dataValue) {
316
317     UA_Int16 ua_pieces_quantity;
318     float distance;
319     esp_err_t res;
320
321     res = ultrasonic_measure(&ultrassonic_sensor, MAX_DISTANCE_CM, &distance);
322
323     if (res != ESP_OK){
324         printf("Error %d: ", res);
325         switch (res){
326             case ESP_ERR_ULTRASONIC_PING:
327                 printf("Cannot ping (device is in invalid state)\n");
328                 break;
329             case ESP_ERR_ULTRASONIC_PING_TIMEOUT:
330                 printf("Ping timeout (no device found)\n");
331                 break;
332             case ESP_ERR_ULTRASONIC_ECHO_TIMEOUT:
333                 printf("Echo timeout (i.e. distance too big)\n");
334                 break;
335             default:
336                 printf("%s\n", esp_err_to_name(res));
337         }
338     } else {
339         ua_pieces_quantity = pieces_quantity(&distance);
340     }
341
342     UA_Variant_setScalarCopy(&dataValue->value, &ua_pieces_quantity,
343                              &UA_TYPES[UA_TYPES_INT16]);
344     dataValue->hasValue = true;
345     return UA_STATUSCODE_GOOD;
346 }
```

Figura 31 – Funcionalidade da quantidade de peças implementada no arquivo *model.c*

Fonte: Autoria própria

```
117     /* Add Information Model Objects Here */
118     // addLEDMethod(server);
119     addCurrentTemperatureDataSourceVariable(server);
120     addRelay0ControlNode(server);
121     addRelay1ControlNode(server);
122     addCurrentDistanceDataSourceVariable(server);
123     addCurrentPiecesQuantityDataSourceVariable(server);
124
```

Figura 32 – Funcionalidade da quantidade de peças implementada no arquivo *opcua_esp32.c*

Fonte: Autoria própria

Dessa forma, a partir do software *UAExpert*, foi possível visualizar o dado da quantidade de peças, como ilustra a Figura 33.

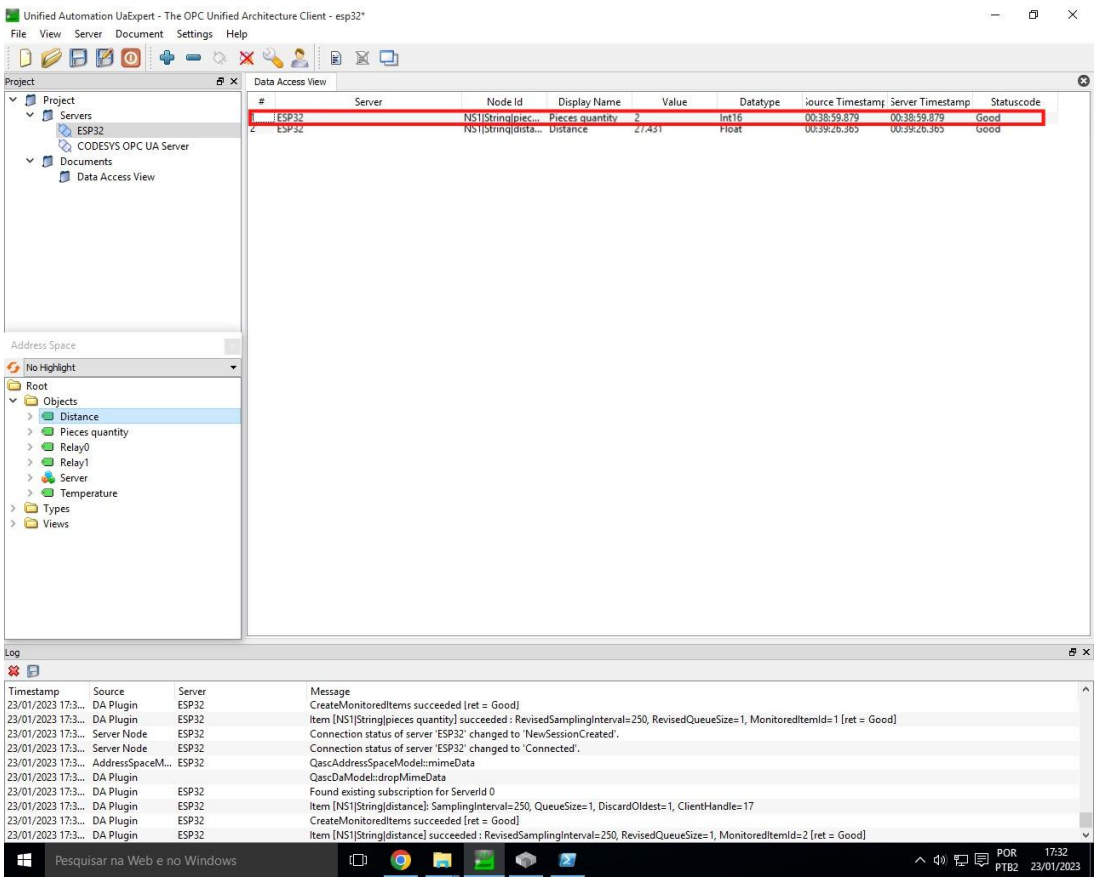

Figura 33 – Nó OPC-UA referente a variável de quantidade de peças no servidor da ESP32

Fonte: Autoria própria

4.2.3 Conexão entre os servidores OPC-UA

Para realizar a conexão entre o servidor OPC-UA presente na placa de desenvolvimento ESP32 e o servidor OPC-UA presente no CLP da planta industrial *FESTO MPS*,

foi adicionada uma variável *numpc* do tipo inteiro ao arquivo *PLC_PRG* presente na aba *Application* de um programa anteriormente carregado no CLP no software *CODESYS provided by Festo*.

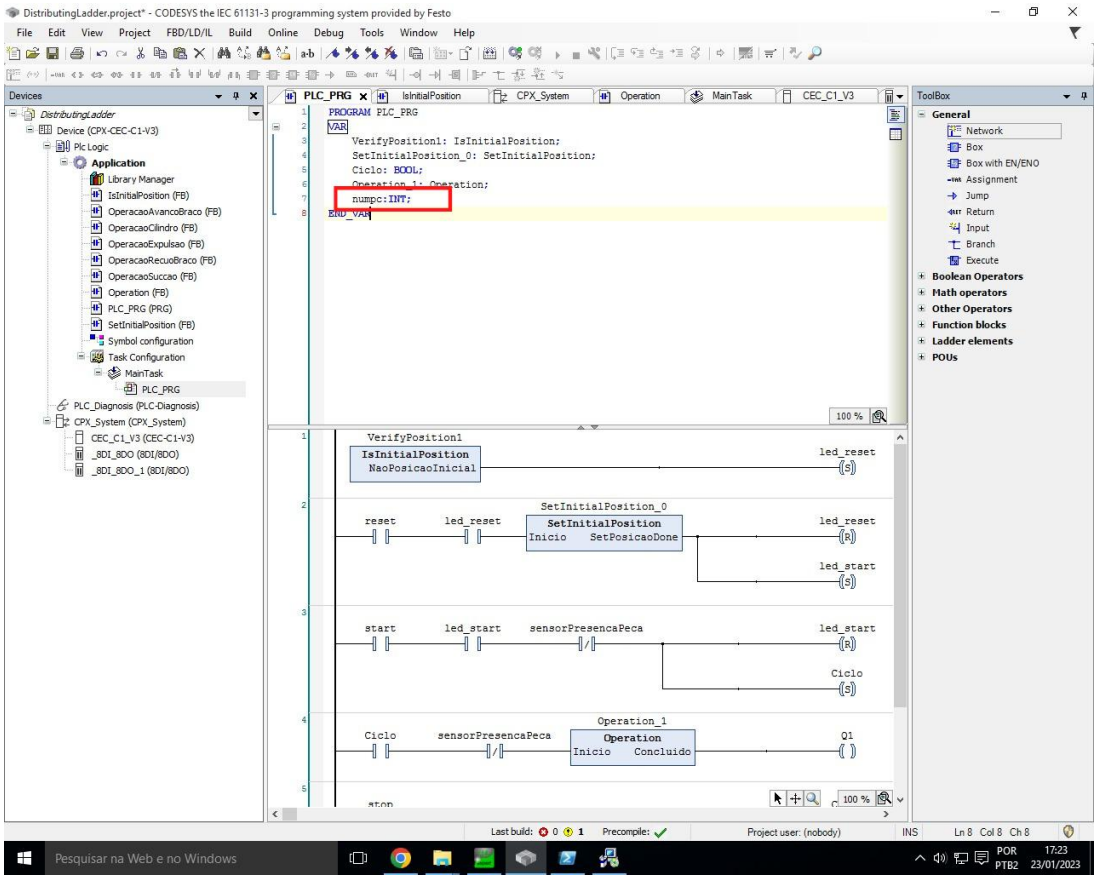

Figura 34 – Variável referente a quantidade de peças no software *CODESYS provided by Festo*

Fonte: Autoria própria

Em seguida, para ser possível sua visualização e edição via OPC-UA, foi adicionada a configuração *Symbol configuration* e selecionado o campo *Support OPC-UA Features*.

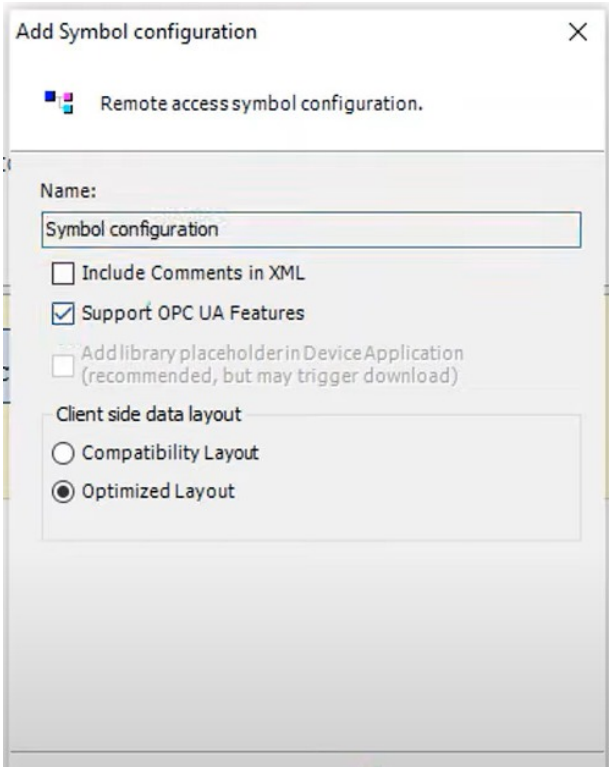

Figura 35 – Configuração do *Symbol configuration* e seleção o campo *Support OPC-UA Features*

Fonte: Autoria própria

A variável *numpc* foi atualizada para permissão de leitura e escrita. Em seguida, o código modificado foi novamente carregado no CLP.

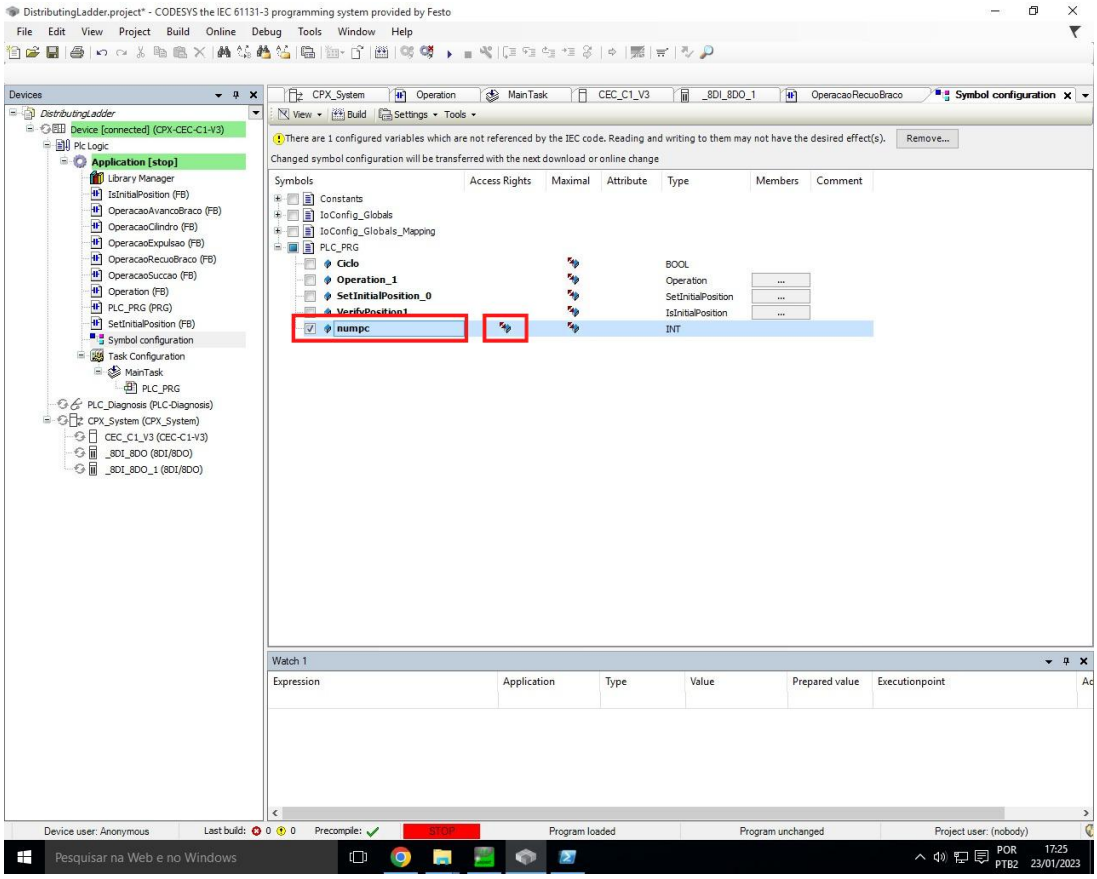

Figura 36 – Variável *numpc* e sua permissão de acesso (leitura e escrita)

Fonte: Autoria própria

Além disso, na aba *Devices*, é possível visualizar o IP de acesso ao servidor OPC-UA presente no CLP.

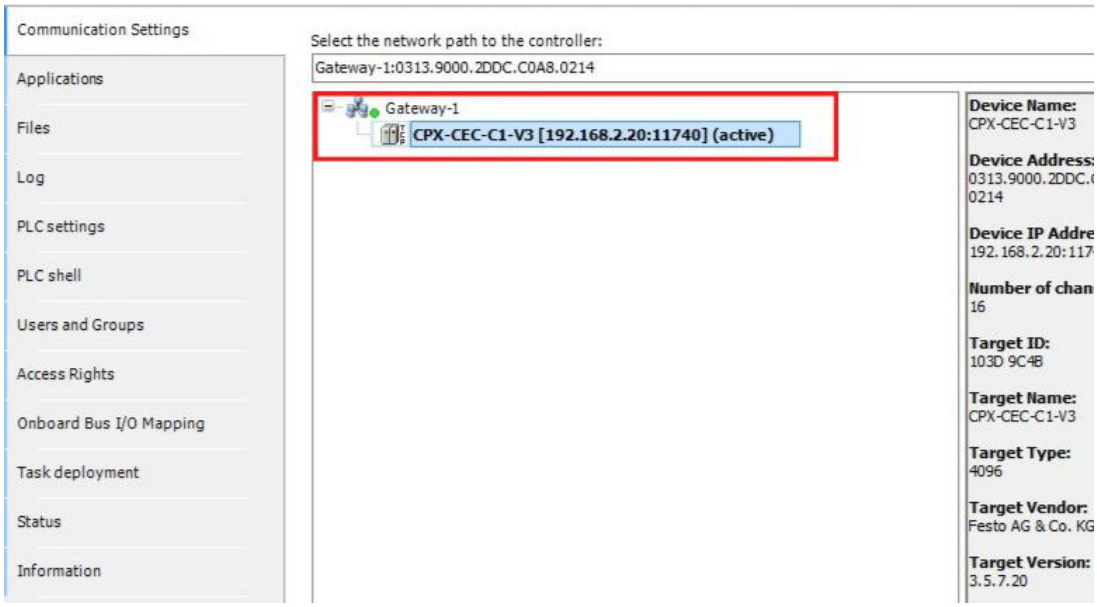

Figura 37 – IP do servidor exibido na aba *devices*

Fonte: Autoria própria

Para realizar a escrita do dado da quantidade de peças do Nó OPC-UA presente na placa ESP32 na variável *numpc* presente no CLP, foi utilizada a ferramenta *Node-RED*. Realizada a instalação da ferramenta, foi instalado o *node* que possibilita a realização da comunicação entre dispositivos de automação utilizando o protocolo OPC-UA, *node-red-contrib-opcua*.

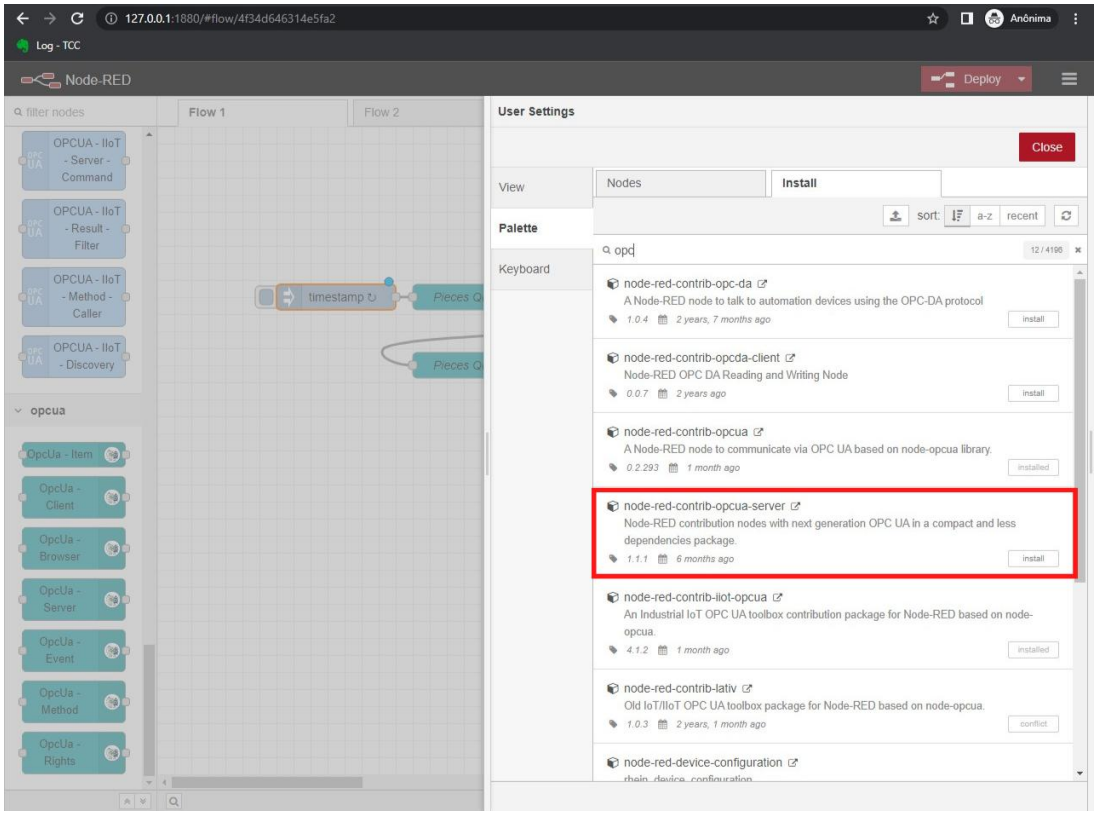

Figura 38 – Extensão para *Node-RED node-red-contrib-opcua*

Fonte: Autoria própria

Posteriormente, foi adicionado o campo *OpcUa - Client* que estabelece a comunicação do cliente com o servidor OPC-UA configurado, e o campo *OpcUa - Item* que seleciona a variável do servidor a qual deverá ser avaliada.

Figura 39 – Campos *OpcUa - Client* e *OpcUa - Item* na interface do Node-RED

Fonte: Autoria própria

Inicialmente, foi feita a configuração para execução da leitura da variável do número de peças proveniente do servidor OPC-UA da placa ESP32. Para isso, no item *OpcUa - Client* foi configurado o IP do servidor, bem como a ação desejada, no caso *READ*, como ilustra a Figura 40.

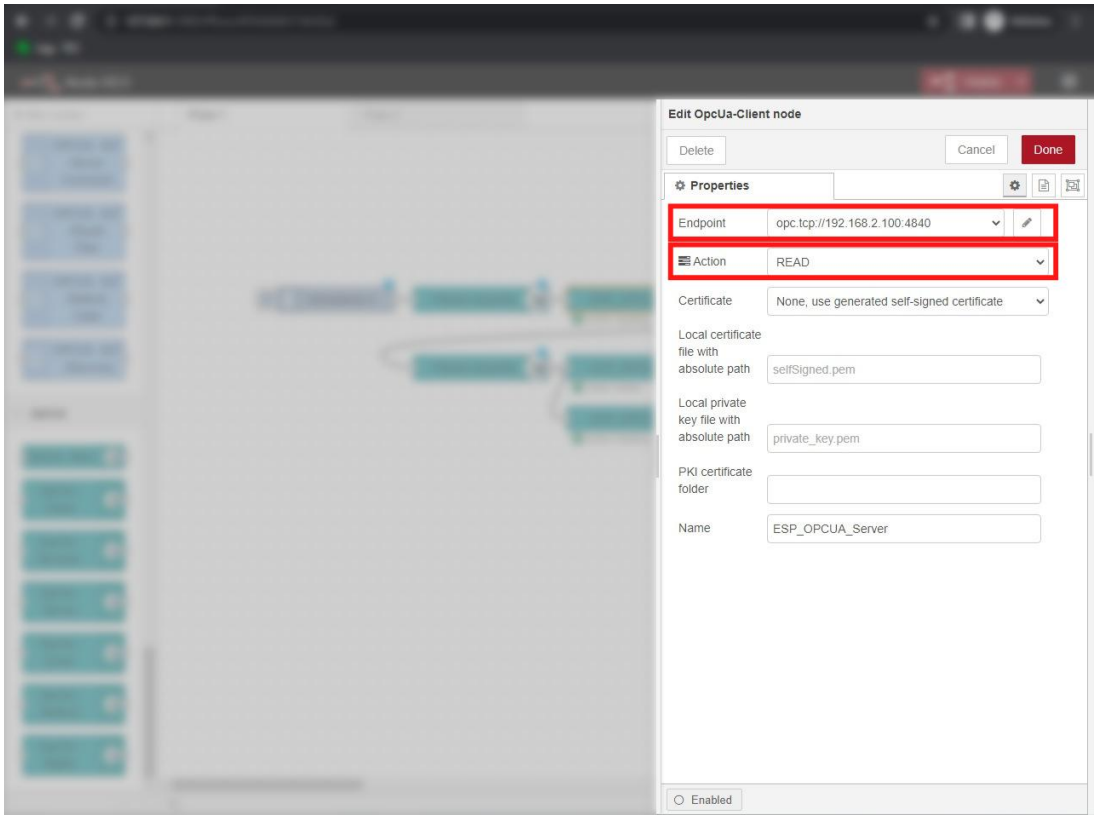

Figura 40 – Configuração do item *OpcUa - Client* na interface do *Node-RED* referente a aplicação na ESP32

Fonte: Autoria própria

A variável a ser lida foi configurada no item *OpcUa - Item*, onde foi selecionado o respectivo *NodeId* da variável de quantidade de peças, como ilustra a Figura 41.

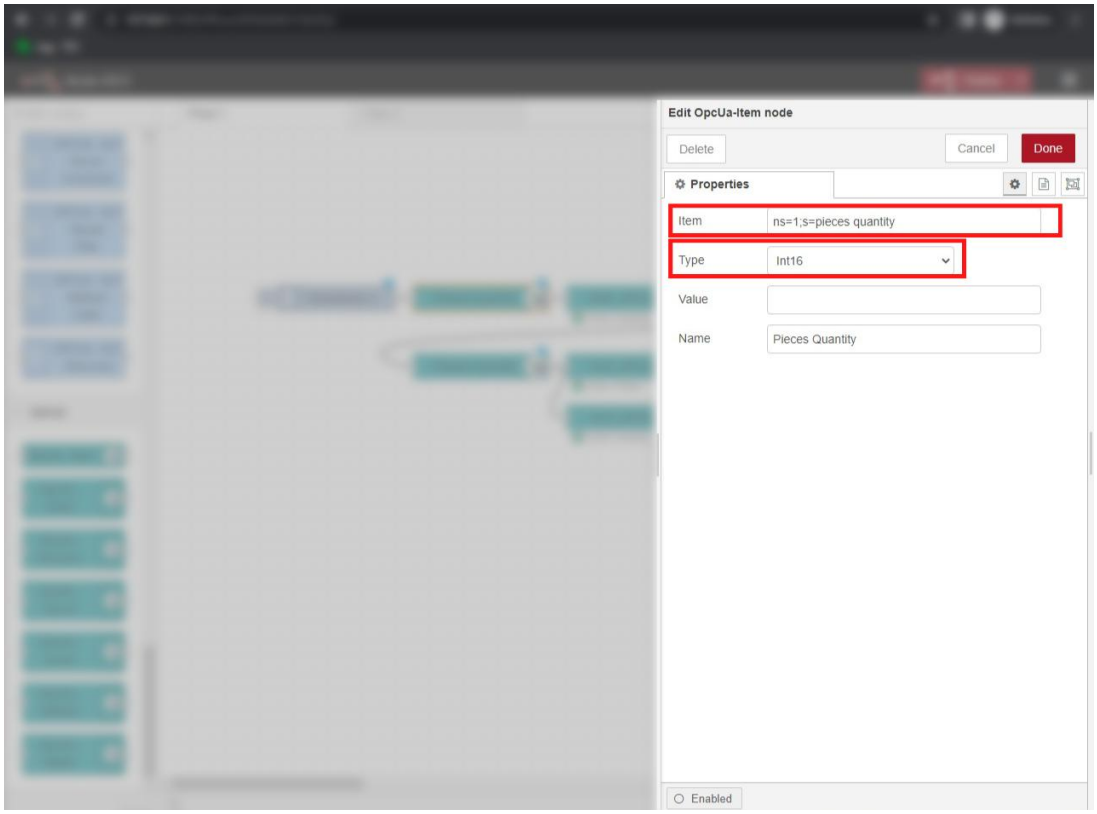

Figura 41 – Configuração do item *OpcUa - Item* na interface do Node-RED referente a aplicação na ESP32

Fonte: Autoria própria

Além disso, também foram adicionados um item de *timestamp* para obter os dados em uma determinada frequência de tempo configurada, e um item de *debug*, para visualizar a mensagem exibida na execução do comando. O item de *timestamp* foi configurado para realizar requisições de leitura a cada 0.5 segundo, como ilustra a Figura 42.

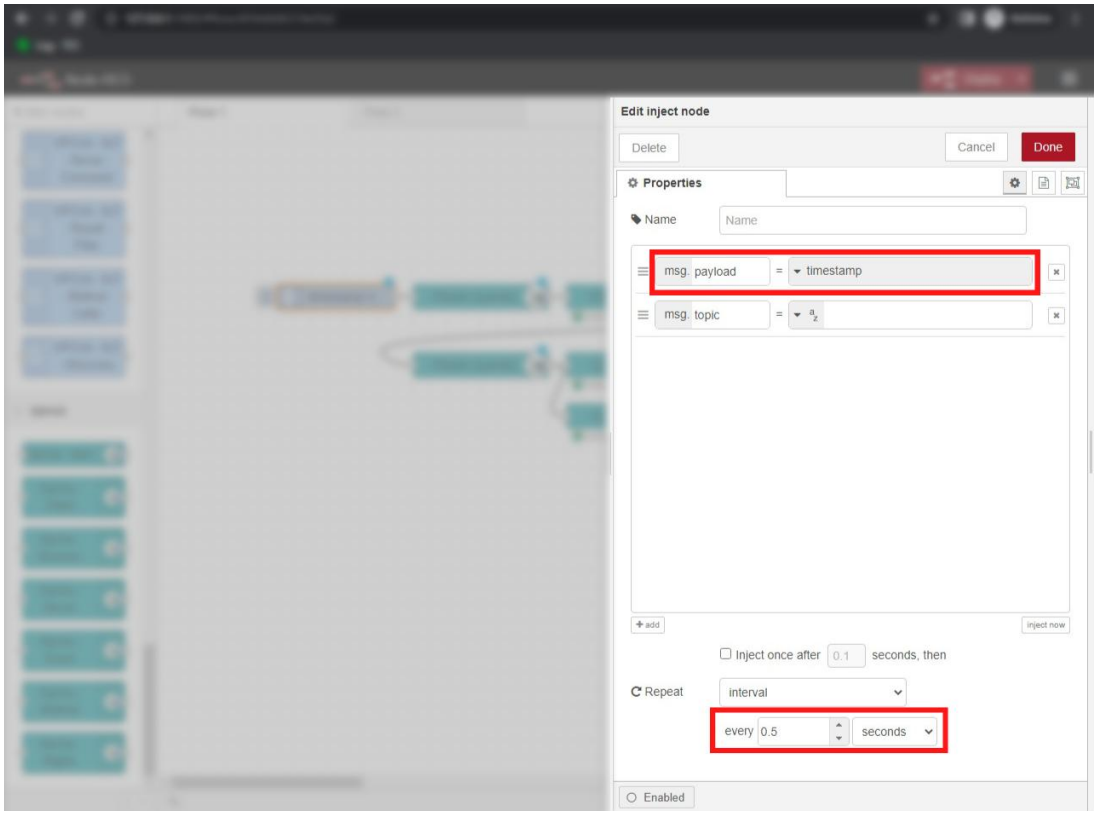

Figura 42 – Configuração do item *timestamp* na interface do *Node-RED*

Fonte: Autoria própria

Finalizada a configuração do comando de leitura da variável da quantidade de peças do servidor OPC-UA da ESP32, foi realizada a adição dos itens referentes a escrita da variável *numpc* no servidor OPC-UA do CLP. Para isso, os itens *OpcUa - Client* e *OpcUa - Item* também foram adicionados. A configuração do item *OpcUa - Client* é exibido na Figura 43, nele é configurado o IP do servidor, bem como a ação de escrita, no caso *WRITE*.

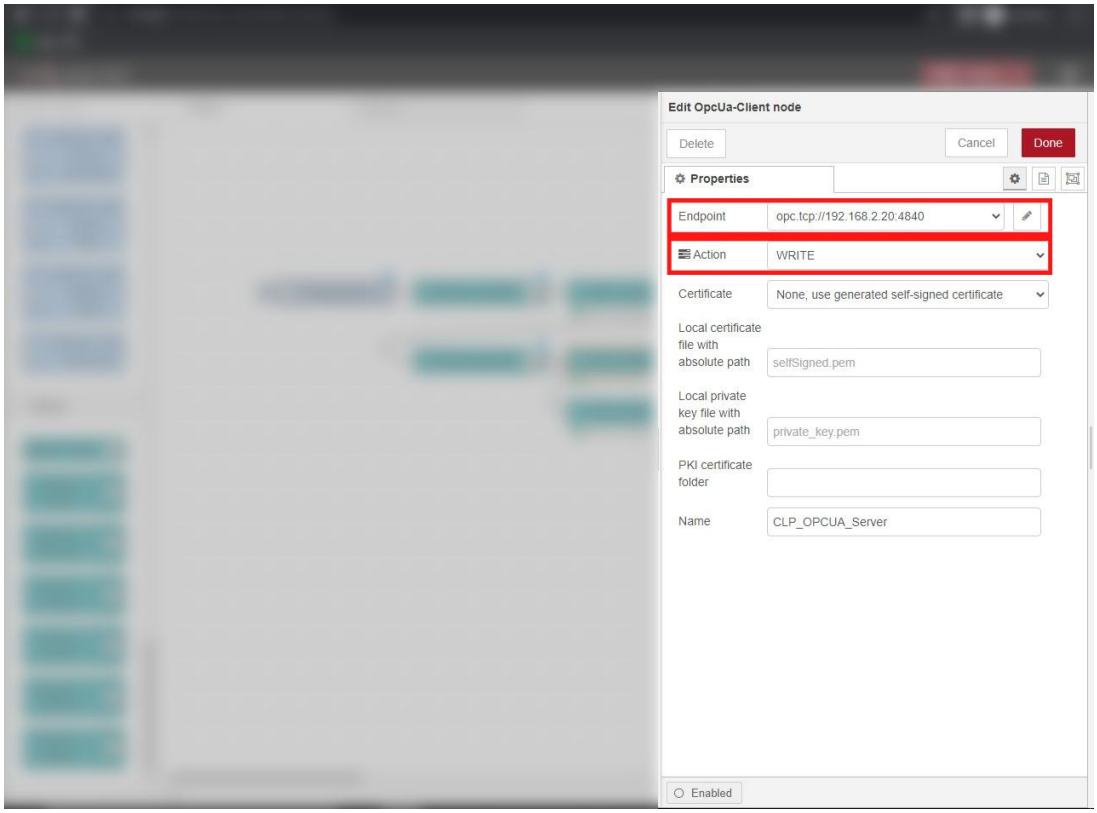

Figura 43 – Configuração do item *OpcUa - Client* na interface do Node-RED referente a aplicação no CLP

Fonte: Autoria própria

No item *OpcUa - Item*, foi selecionado o respectivo *NodeId* da variável desejada *numpc*, como ilustra a Figura 44.

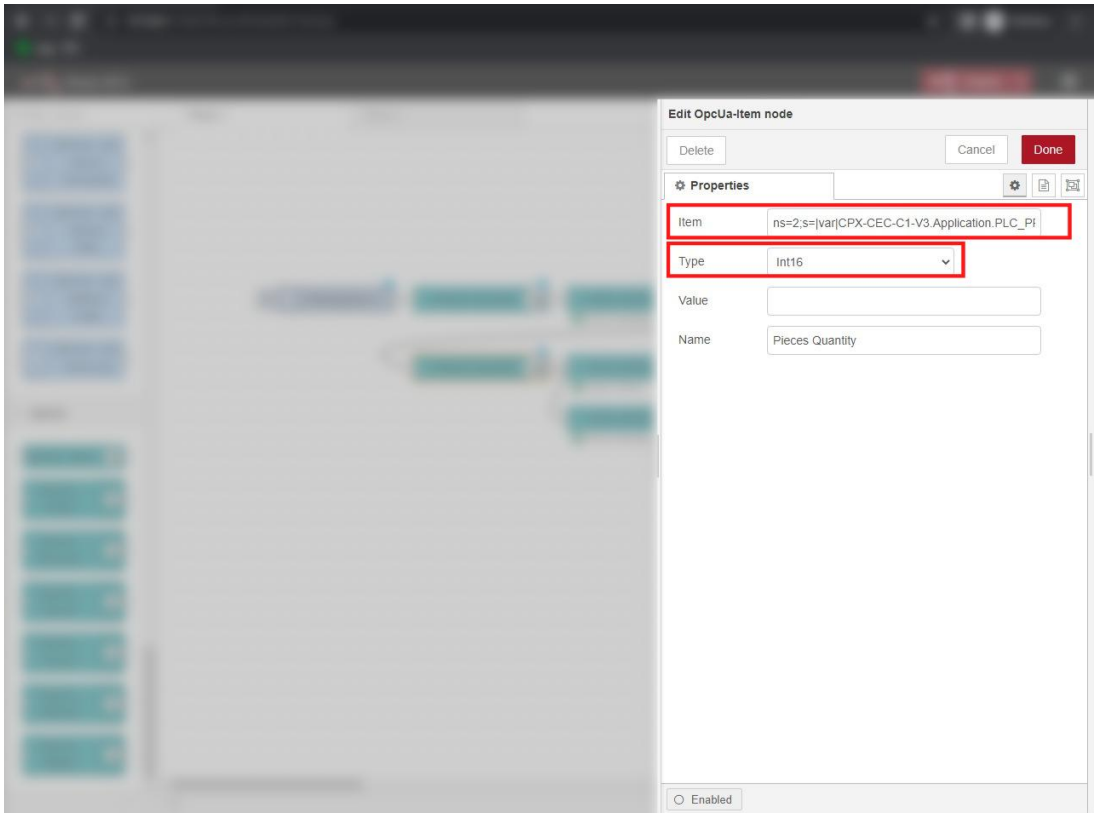

Figura 44 – Configuração do item *OpcUa - Item* na interface do *Node-RED* referente a aplicação no CLP

Fonte: Autoria própria

Em seguida, foi conectada a saída da leitura da variável da quantidade de peças do servidor OPC-UA da ESP32, a entrada da variável *numpc* e adicionada a conexão com o servidor OPC-UA.

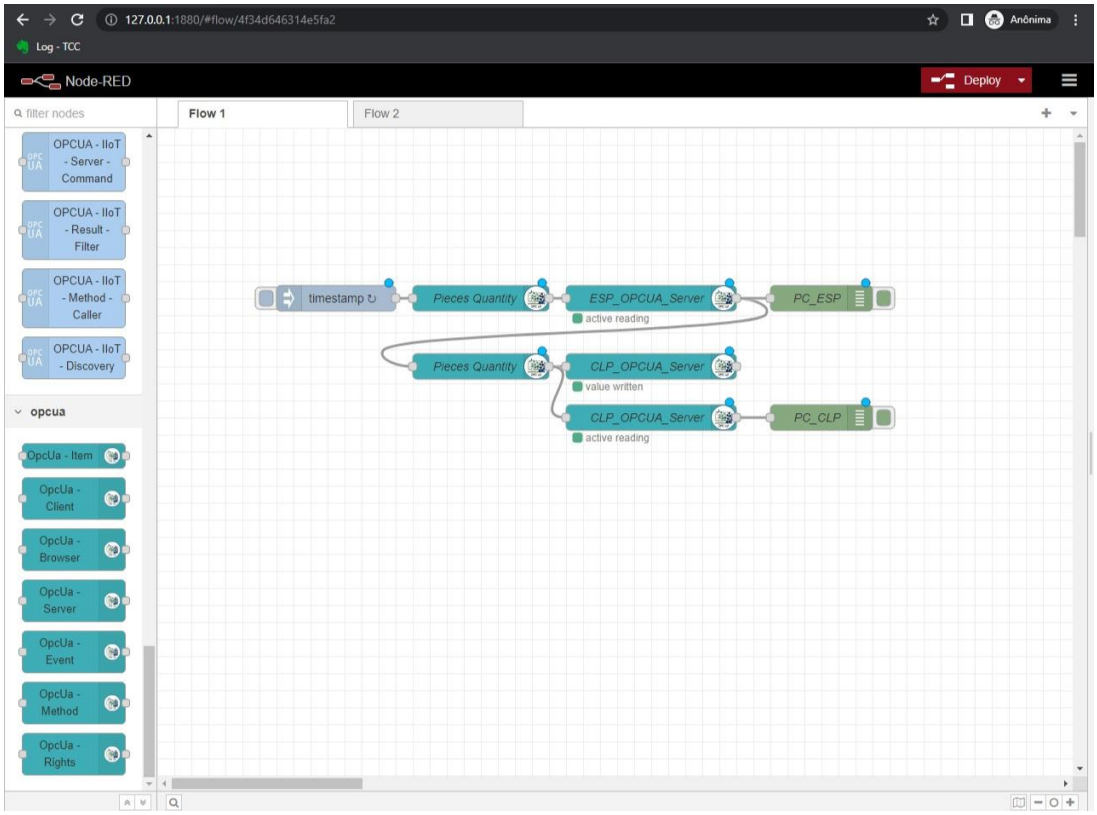

Figura 45 – Conexão dos itens na interface do *Node-RED*

Fonte: Autoria própria

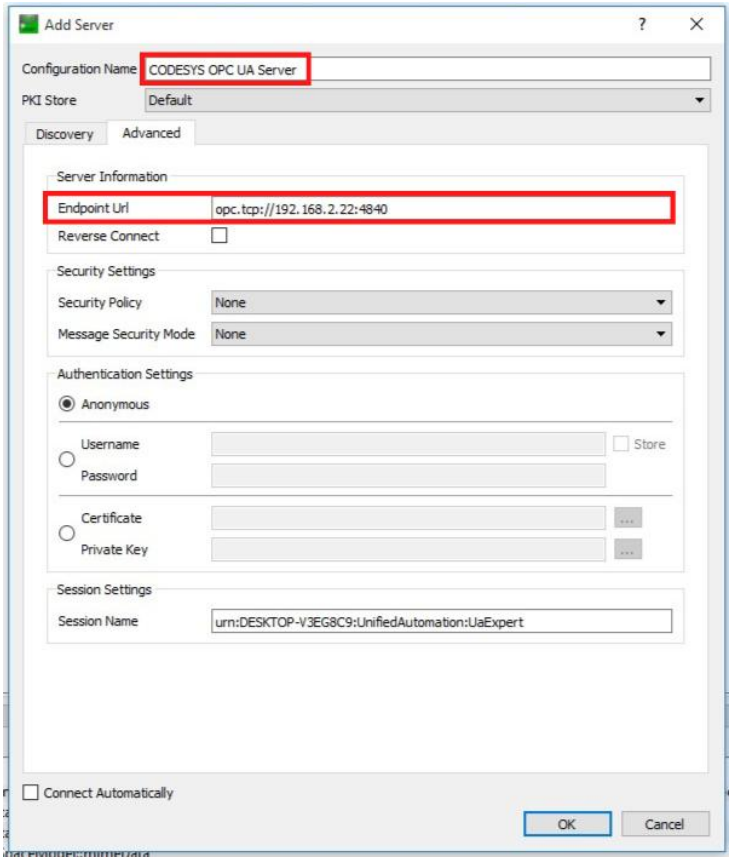

Figura 46 – Configuração para adicionar o servidor OPC-UA do CLP na interface do *UAExpert*

Fonte: Autoria própria

5 Resultados e discussões

Utilizando os métodos e materiais presentes neste trabalho, foi possível realizar a implementação de uma aplicação que determina a quantidade de peças totais no recipiente da planta FESTO MPS, utilizando padrão OPC-UA embarcado na placa de desenvolvimento ESP32.

O estágio inicial de configuração do ambiente de desenvolvimento na placa ESP32, bem como a comunicação entre os dispositivos utilizados (PC, ESP32 e CLP) ocorreu de forma satisfatória. No entanto, foi necessária a utilização de um roteador para que todos os elementos envolvidos pudessem compartilhar de uma mesma faixa de rede, de forma que a comunicação OPC TCP pudesse ocorrer. Com isso, foi construída a seguinte topologia de rede de modo que todos os dispositivos incluídos pudessem enxergar reciprocamente.

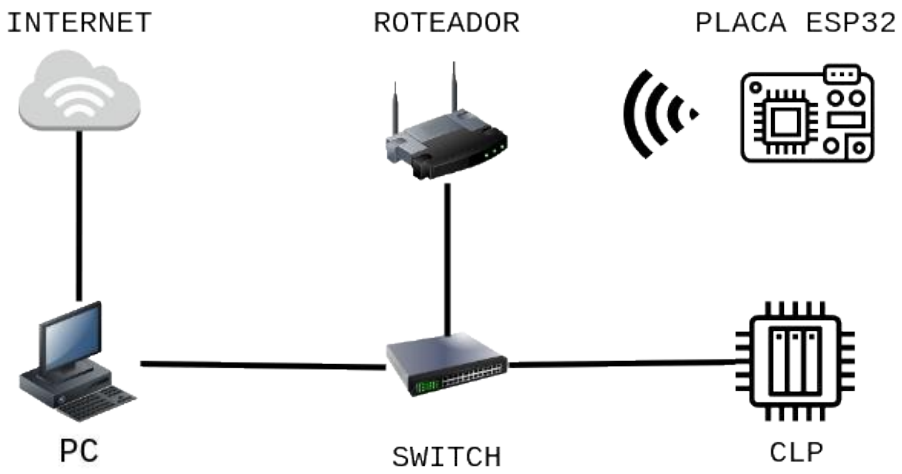

Figura 47 – Topologia de rede presente no projeto

Fonte: Autoria própria

A estrutura base montada para o sensor de distância HC-SR04 também trouxe resultados aceitáveis. Foram realizadas 4 coletas com diversas quantidades de peças, desde o estoque vazio (0 peças) até o estoque completo (11 peças), no qual foi obtido uma média de 1,55% de erro experimental, comparando a real quantidade de peças presentes no recipiente com o valor aproximado pelo sistema. O gráfico representado na Figura 48 ilustra a comparação das coletas versus o valor real inserido no estoque.

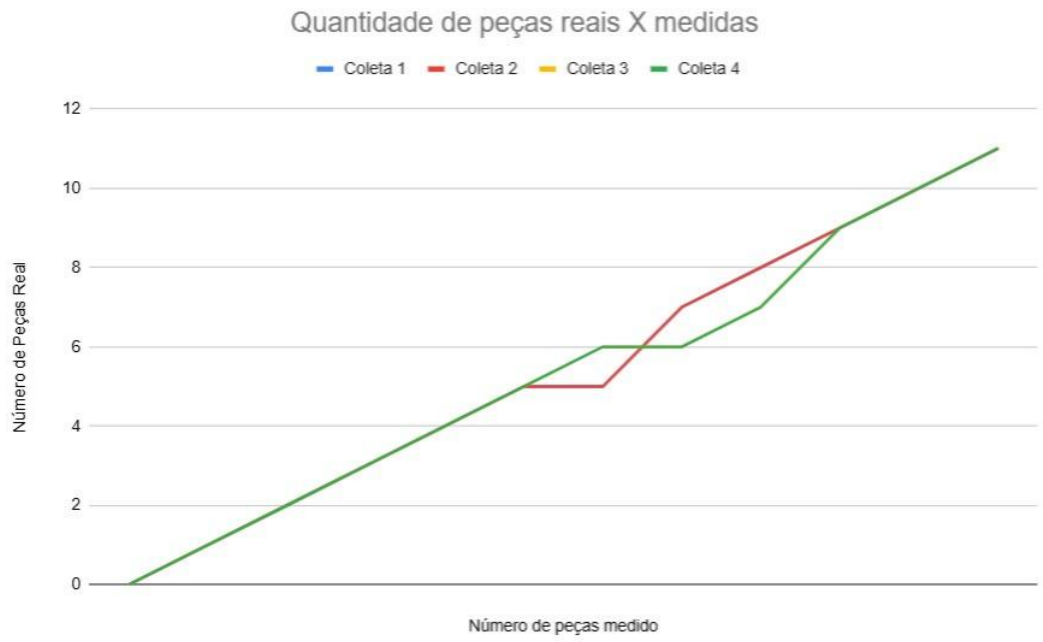

Figura 48 – Gráfico representativo da quantidade de peças reais versus medidas

Fonte: Autoria própria

Todavia, a presença de outros objetos no alcance do sensor pode detectar a presença de uma peça de forma errônea, pois a aplicação trata a modificação na distância total recebida pelo sensor como a adição ou remoção de peças, de modo que até mesmo a detecção do pistão que movimenta as peças no recipiente pode ser detectado como a presença de uma peça. Além disso, a estrutura foi colocada em uma localização estratégica, de modo que o sensor pudesse ficar mais próximo da entrada do recipiente, reduzindo a chances de imprecisão do sensor, conforme fosse aumentada a distância e seu alcance pudesse detectar outros objetos não desejados.

Dessa forma, uma sugestão de melhoria seria a aplicação de um sensor com apenas um feixe de transmissão, de forma que a reduzir o espalhamento. Ademais, a estrutura foi montada com placas de isopor, de maneira a reduzir os custos. Contudo, devido ao uso do isopor, a estrutura base tem facilidade de movimentação, o que pode alterar os valores obtidos pelo sistema, pois o sensor se encontra apontado para o centro do recipiente alvo. Com isso, uma forma para reduzir as chances de perturbação é a construção da base com o auxílio de uma impressora 3D, trazendo mais firmeza para a estrutura.

Por fim, foi possível realizar a transmissão do dado da quantidade de peças do servidor presente na placa ESP32 para o servidor presente no CLP da planta FESTO MPS. Assim, é possível visualizar a quantidade de peças totais no estoque da planta apenas ao acessar o servidor do CLP. A Figura 49 exibe a quantidade de peças exibidas pela ESP32 e pelo CLP, ambas com os mesmos resultados.

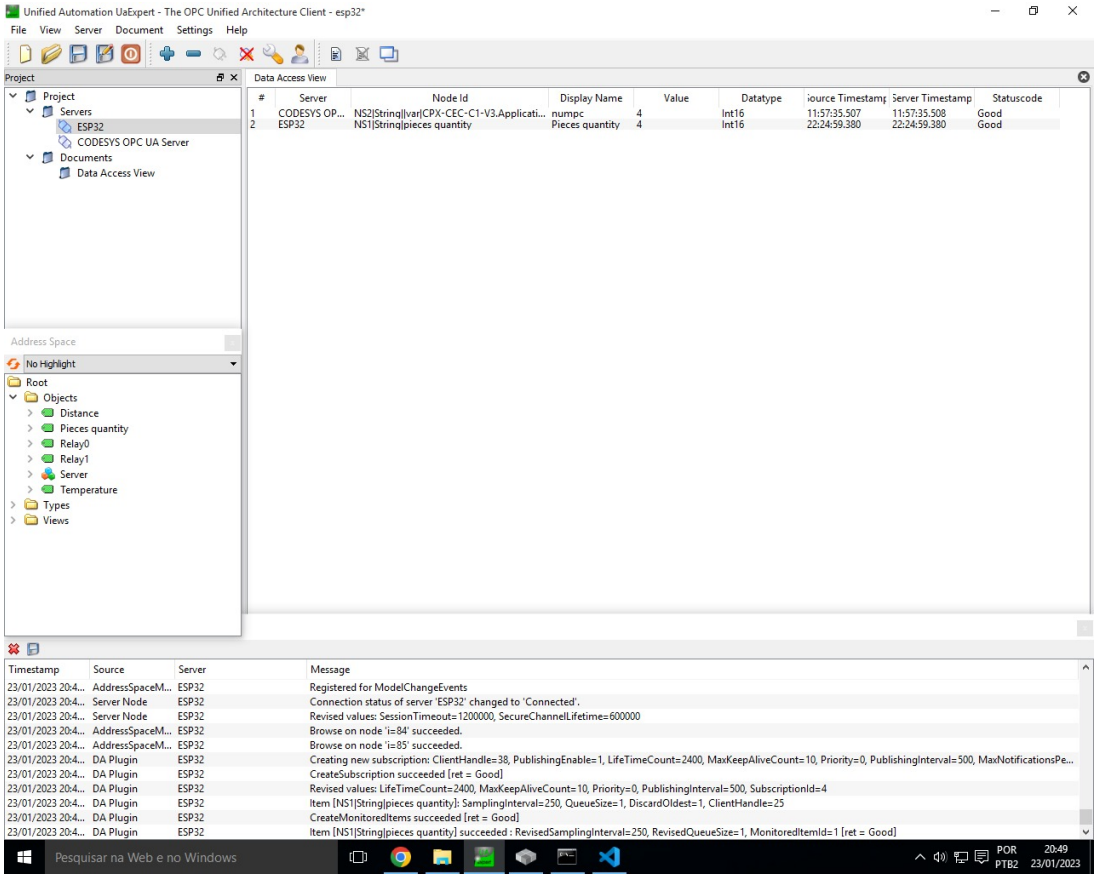

Figura 49 – Interface do software *UAExpert* exibindo os valores da quantidade de peças em ambos os servidores

Fonte: Autoria própria

6 Considerações finais

Diante do que foi proposto neste trabalho, foi possível realizar a instrumentação de uma planta de manufatura flexível, utilizando padrão OPC-UA embarcado, visando estabelecer uma rede de comunicação entre servidores OPC-UA, conforme foi definido como objetivo.

Considera-se um elemento importante para o desenvolvimento deste trabalho a vasta comunidade de desenvolvedores, engenheiros e entusiastas da tecnologia, os quais disponibilizam implementações de código aberto do padrão OPC-UA para o desenvolvimento em placas como a ESP32, agilizando o processo de desenvolvimento devido às funções e recursos oferecidos.

Em suma, espera-se que o desenvolvimento deste trabalho possa servir como exemplo de aplicação OPC-UA em sistemas industriais para aqueles que desejem entender mais sobre o padrão, bem como uma base para aqueles que desejem aprimorar o sistema de medição implementado na planta industrial.

Referências

AL-SARAWI, S. et al. Internet of things (iot) communication protocols. In: IEEE. **2017 8th International conference on information technology (ICIT)**. [S.l.], 2017. p. 685–690. Citado na página 14.

CODESYS® provided by Festo. Disponível em: <https://www.festo-didactic.com/int-en/learning-systems/software-e-learning/programming-software/codesys-provided-by-festo.htm?fbid=aW50LmVuLjU1Ny4xNy4xOC41NjYuNzYwNA>. Citado na página 22.

CONTRIBUTORS, O. F. . **Node-RED**. Disponível em: <https://nodered.org/about/>. Citado na página 23.

ESPRESSIF. **ESP32 Series**. Espressif Systems, 2012. ESP32 Series Datasheet. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>. Citado 2 vezes nas páginas 25 e 26.

FOROUZAN, B. A. **TCP/IP protocol suite**. [S.l.]: McGraw-Hill Higher Education, 2002. Citado na página 15.

HANNELIUS, T.; SALMENPERA, M.; KUIKKA, S. Roadmap to adopting opc ua. In: IEEE. **2008 6th IEEE International Conference on Industrial Informatics**. [S.l.], 2008. p. 756–761. Citado na página 12.

HC-SR04 Timing diagram. Disponível em: <http://www.circuitstoday.com/ultrasonic-range-finder-using-8051>. Citado na página 28.

JABALQUINTO, A. **Create your own framework: Node-Red. First steps**. 2018. Disponível em: <https://www.industrialshields.com/blog/arduino-industrial-1/post/create-your-own-framework-node-red-first-steps-6>. Citado na página 23.

KAIGHOBADI, M.; VENKATESH, K. Flexible manufacturing systems: an overview. **International Journal of Operations & Production Management**, MCB UP Ltd, 1994. Citado na página 18.

MAHNKE, W.; LEITNER, S.-H.; DAMM, M. **OPC unified architecture**. [S.l.]: Springer Science & Business Media, 2009. Citado 3 vezes nas páginas 12, 15 e 17.

MELO, P. **Introdução ao OPC UA (Open Platform Communications Unified Architecture)**. 2018. Disponível em: <https://embarcados.com.br/introducao-ao-opc-ua/">. Citado na página 16.

MPS® The Modular Production System. Disponível em: <https://www.festo-didactic.com/int-en/learning-systems/mps-the-modular-production-system/>. Citado na página 24.

MóDULO Sensor de Distância Ultrassônico HC-SR04. Disponível em: <https://www.eletrogate.com/modulo-sensor-de-distancia-ultrassonico-hc-sr04>. Citado na página 27.

MóDULO WiFi ESP32 Bluetooth. Disponível em: <https://www.filipeflop.com/produto/modulo-wifi-esp32-bluetooth/>. Citado na página 25.

O que é OPC UA? Disponível em: <https://www.br-automation.com/pt-br/tecnologias/opc-ua/>. Citado na página 17.

OPC FOUNDATION. **OPC unified architecture—part 1: Overview and concepts**. [S.l.], 2017. Release 1.04. Citado na página 18.

OPCBLOGADMIN. **What is OPC? Learn about the most used technology in the automation world**. 2018. Disponível em: <http://opcconnect.integrationobjects.com/what-is-opc-most-used-technology-automation-world/>. Citado na página 16.

OPEN62541. Disponível em: <https://github.com/open62541/open62541>. Citado na página 20.

SOUZA, D. A. C. d. Implementação de comunicação no padrão opc ua em microcontroladores de baixo custo. Universidade Estadual Paulista (Unesp), 2022. Citado na página 12.

TECHOPEDIA. **What is a communication protocol?** 2020. Disponível em: <https://www.techopedia.com/definition/25705/communication-protocol>. Citado na página 14.

THOMSEN, A. **Como conectar o Sensor Ultrassônico HC-SR04 ao Arduino**. 2011. Disponível em: <https://www.filipeflop.com/blog/sensor-ultrassonico-hc-sr04-ao-arduino/>. Citado na página 27.

UAEXPERT—A Full-Featured OPC UA Client. Disponível em: <https://www.unified-automation.com/products/development-tools/uaexpert.html>. Citado na página 21.

ULTRASONIC Ranging Module HC-SR04. Elecfreaks. Disponível em: <https://www.electroschematics.com/wp-content/uploads/2013/07/HCSR04-datasheet-version-1.pdf>. Citado na página 27.

WOLLSCHLAEGER, M.; SAUTER, T.; JASPERNEITE, J. The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0. **IEEE industrial electronics magazine**, IEEE, v. 11, n. 1, p. 17–27, 2017. Citado na página 14.

XU, L. D.; XU, E. L.; LI, L. Industry 4.0: state of the art and future trends. **International journal of production research**, Taylor & Francis, v. 56, n. 8, p. 2941–2962, 2018. Citado na página 12.