



UNIVERSIDADE ESTADUAL PAULISTA  
"JÚLIO DE MESQUITA FILHO"  
Instituto de Ciência e Tecnologia  
Câmpus de Sorocaba

GABRIEL SILES PETROCCHI

# INTEGRAÇÃO ENTRE RFID E PADRÃO OPC UA APLICADA A UM SISTEMA DE MANUFATURA

Sorocaba

2024

GABRIEL SILES PETROCCHI

INTEGRAÇÃO ENTRE RFID E PADRÃO OPC UA APLICADA A UM SISTEMA DE  
MANUFATURA

Trabalho de Conclusão de Curso  
apresentado ao Instituto de Ciência e Tecnologia  
de Sorocaba, Universidade Estadual Paulista  
(UNESP), como parte dos requisitos para obtenção  
do grau de Bacharel em Engenharia de Controle e  
Automação.

Orientador: Prof. M.Sc. Vitor Mendes Caldana

Coorientador: Prof. Dr. Eduardo Paciencia Godoy

Sorocaba

2024

P497i      Petrocchi, Gabriel Siles  
Integração entre RFID e padrão OPC UA aplicada a um sistema de  
manufatura / Gabriel Siles Petrocchi. -- Sorocaba, 2024  
71 p. : il., tabs., fotos

Trabalho de conclusão de curso (Bacharelado - Engenharia de  
Controle e Automação) - Universidade Estadual Paulista (UNESP),  
Instituto de Ciência e Tecnologia, Sorocaba  
Orientador: Vitor Mendes Caldana  
Coorientador: Eduardo Paciencia Godoy

1. Internet das Coisas. 2. Sistemas de identificação por  
radiofrequência. 3. Automação. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Universidade  
Estadual Paulista (UNESP), Instituto de Ciência e Tecnologia, Sorocaba. Dados fornecidos pelo  
autor(a).

Essa ficha não pode ser modificada.



UNIVERSIDADE ESTADUAL PAULISTA  
“JÚLIO DE MESQUITA FILHO”  
Instituto de Ciência e Tecnologia  
Câmpus de Sorocaba

INTEGRAÇÃO ENTRE RFID E PADRÃO OPC UA APLICADA A UM SISTEMA DE  
MANUFATURA

GABRIEL SILES PETROCCHI

ESTE PROJETO FINAL DE CURSO FOI JULGADO ADEQUADO  
COMO PARTE DO REQUISITO PARA A OBTENÇÃO DO GRAU DE  
**BACHAREL EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO**

Prof. Dr. Everson Martins

Coordenador

**BANCA EXAMINADORA:**

Prof. Me. Vitor Mendes Caldana

Orientador/UNESP – Campus Sorocaba

Prof. Dr. Luis Armando de Oro Arenas

UNESP – Campus Sorocaba

Prof. Me. Diego Deotti

IFSP – Campus Campinas - Examinador Externo

Agosto de 2024

## **AGRADECIMENTOS**

Primeiramente gostaria de agradecer minha família, em especial meu pai Nelson Petrocchi e minha mãe Catarina Marcia que sempre me deram suporte e apoio durante a toda a minha caminhada.

Agradeço ao Prof. M.Sc. Vitor Mendes Caldana e Prof. Dr. Eduardo Paciencia Godoy pelas orientações e auxílios para o desenvolvimento do projeto.

Agradeço aos amigos e colegas de curso Giovanni Dotta, Guilherme Doubek, João Vitor, Luís Fernando e Mateus Costa pelo companheirismo e eventuais ajudas durante a minha graduação.

E por fim gostaria de agradecer aos professores e técnicos da UNESP campus Sorocaba, minha formação acadêmica foi possível graças aos esforços de todos vocês.

PETROCCHI, G. S. **Integração entre RFID e padrão OPC UA aplicada a um sistema de manufatura**. 2024. Trabalho de Conclusão de Curso (Engenharia de Controle e Automação) - Instituto de Ciência e Tecnologia, Universidade Estadual Paulista, Sorocaba, 2024.

## RESUMO

A tecnologia RFID (*Radio Frequency Identification*) é um sistema de alta confiabilidade para o gerenciamento de dados, pois com peças que possuem *tags*, a leitura e escrita de dados pode ser feita de uma maneira simples sem a necessidade de contatos físicos ou elétricos, além de que é possível fazer a leitura de múltiplas peças ao mesmo tempo. Outra tecnologia relacionada ao gerenciamento de dados em ascensão, graças a Indústria 4.0, é o padrão de comunicação OPC UA (*Open Platform Communication Unified Architecture*) que fornece uma troca segura de informações e dados na área da automação industrial, utilizando um protocolo padrão extensível que pode ser empregado em multiplataformas, além de que, é possível também fazer a alteração das informações pelo seu sistema de servidor e cliente graças as suas funcionalidades na questão de acesso aos dados. Com isto em vista, este trabalho implementou a integração entre a tecnologia RFID e o padrão OPC UA aplicada ao sistema de manufatura flexível (FMS) presente no Laboratório de Automação 2 na UNESP campus Sorocaba. O desenvolvimento do projeto utilizou como ferramenta de programação o ambiente do Node-RED, que realiza a configuração de inicialização do leitor RFID de forma automática, permitindo que as informações sejam recebidas e passadas via cabo de conexão Ethernet com o leitor. Também pelo Node-RED, o servidor é criado e configurado utilizando uma paleta de nós específica com as funcionalidades relacionadas ao OPC UA. A leitura e escrita das peças com *tags* é feita tanto pelo servidor quanto pela Dashboard do Node-RED, além do mais, dependendo dos valores de escrita, a peça faz diferentes paradas pelas estações do sistema de manufatura.

**Palavras-chave:** Indústria 4.0, Tecnologia RFID, OPC UA, Node-RED.

PETROCCHI, G. S. **Integration between RFID and OPC UA standard applied to a manufacturing system.** 2024. Final Paper (Bachelor's degree in Control and Automation Engineering) - Institute of Science and Technology, São Paulo State University, Sorocaba, 2024.

## **ABSTRACT**

RFID (Radio Frequency Identification) technology is a highly reliable system for data management, as with parts that have tags, reading and writing data can be done in a simple way without the need for physical or electrical contacts, in addition to it is possible to read multiple parts at the same time. Another technology related to data management on the rise, thanks to Industry 4.0, is the OPC UA (Open Platform Communication Unified Architecture) communication standard, which provides a secure exchange of information and data in the area of industrial automation, using an extensible standard protocol that can be used on multiplatforms, in addition, it is also possible to change information through your server and client system thanks to its functionalities in terms of data access. With this in mind, this work implemented the integration between RFID technology and the OPC UA standard applied to the flexible manufacturing system (FMS) present in the Automation Laboratory 2 at the UNESP Sorocaba campus. The development of the project used the Node-RED environment as a programming tool, which perform the initialization configuration of the RFID reader automatically, allowing information to be received and passed via Ethernet connection cable to the reader. Also using Node-RED, the server is created and configured using a specific palette of nodes with functionalities related to OPC UA. The reading and writing of parts with tags is done both by the server and by the Node-RED Dashboard. Furthermore, depending on the writing values, the part makes different stops at the stations of the manufacturing system.

**Keywords:** Industry 4.0, RFID Technology, OPC UA, Node-RED.

## LISTA DE FIGURAS

Figura 1 - Sistema básico da utilização da tecnologia RFID.....	16
Figura 2 - Aplicação do padrão OPC UA em um sistema de automação industrial. ....	18
Figura 3 - Espaço de endereço no servidor OPC UA. ....	19
Figura 4 - Nó Base e suas classes derivadas.....	19
Figura 5 - Ambiente de desenvolvimento do Node-RED.....	21
Figura 6 - Fluxograma apresentando o funcionamento geral do projeto.....	22
Figura 7 – FMS da FESTO presente no laboratório de automação da UNESP campus Sorocaba. .....	23
Figura 8 - Nós utilizados no Node-RED para o controle de dados do FMS. ....	24
Figura 9 - Indicadores luminosos presente na estrutura superior do leitor RFID INfinity 510. .....	25
Figura 10 – Endereços de porta e suas funcionalidades para a comunicação M2M do leitor INfinity 510. ....	26
Figura 11 - Foto das antenas posicionadas em frente as estações do FMS. ....	27
Figura 12 - Exemplo de tela de visualização de acesso de dados OPC UA pelo UA Expert. ..	28
Figura 13 - Programação feita para a configuração do leitor RFID via Node-RED parte I. ....	31
Figura 14 - Programação feita para a configuração do leitor RFID via Node-RED parte II....	32
Figura 15 - Programação feita dentro do nó Comando de inicialização para o envio de comandos de configuração ao leitor RFID. ....	33
Figura 16 - Programação do envio das confirmações ou erros da execução dos comandos utilizados na configuração do leitor RFID para o Dashboard. ....	33
Figura 17 - Programação do Node-RED destacando o envio de leitura de <i>tags</i> para o Dashboard. .....	34
Figura 18 - Configurações do nó Connection ID/Tag Report. ....	34
Figura 19 - Programação da leitura via Dashboard parte I.....	34
Figura 20 - Programação da leitura via Dashboard parte II. ....	35
Figura 21 - Programação feita dentro do nó <i>Comando checar ID Reader</i> .....	35
Figura 22 - Programação da escrita via Dashboard parte I.....	36
Figura 23 - Programação da escrita via Dashboard parte II. ....	36
Figura 24 - Configuração do nó Alterar <i>Tag ID</i> . ....	36
Figura 25 - Programação dentro do nó Comando alterar ID Reader. ....	37
Figura 26 - Nó responsável pela criação do servidor OPC UA em funcionamento. ....	37
Figura 27 - Configurações do nó responsável pela execução do servidor OPC UA.....	38
Figura 28 - Programação do Node-RED para a criação dos NameSpaces, pastas e variáveis. 39	
Figura 29 - Programação no Node-RED feita para enviar os dados de TagReport ao servidor OPC UA.....	41
Figura 30 - Configuração do nó para a organização de dados por antena.....	41
Figura 31 - Programação presente no nó function denominado Valor: Antena 1 TagReport. ..	42



Figura 32 - Nó OPC UA Client funcionando corretamente. ....	42
Figura 33 - Configurações do nó OPC UA Client. ....	43
Figura 34 - Programação no Node-RED feita para enviar os dados de <i>Tag ID</i> e <i>User Data</i> ao servidor OPC UA. ....	43
Figura 35 - Configuração do nó <i>interruptor</i> que separa os dados de <i>Tag ID</i> e <i>User Data</i> . ....	44
Figura 36 - Programação do nó <i>Valor: Antena 1 Leitura TagID</i> . ....	44
Figura 37 - Programação do nó <i>Valor: Antena 1 Leitura UserData</i> . ....	44
Figura 38 - Programação feita para apagar os dados de <i>TagID</i> e <i>UserData</i> na saída da tag. ...	45
Figura 39 - Programação do nó <i>Apaga Valor Antena 1 Leitura TagID</i> . ....	45
Figura 40 - Programação do nó <i>Apaga Valor Antena 1 Leitura UserData</i> . ....	46
Figura 41 - Programação realizada no Node-RED para a escrita via servidor OPC UA parte I. ....	46
Figura 42 - Configurações do nó <i>Escreve do Server OPCUA</i> . ....	47
Figura 43 - Configuração do nó <i>Filtra dados corretos</i> . ....	47
Figura 44 - Configuração do nó <i>Separa os dados escritos por antena</i> . ....	47
Figura 45 - Programação realizada no Node-RED para a escrita via servidor OPC UA parte II. ....	48
Figura 46 - Configuração do nó <i>definir msg.payload.antena</i> relacionado a escrita pela antena 1. ....	48
Figura 47 - Configuração do nó <i>Escreve Antena 1</i> . ....	49
Figura 48 - Programação do nó <i>Escreve/Altera Tag ID</i> . ....	49
Figura 49 - Programação do nó <i>Escreve/Altera User Data</i> . ....	49
Figura 50 - Programação realizada no Node-RED para a escrita via servidor OPC UA parte III. ....	50
Figura 51 - Programação do nó <i>Escrita Servidor Sucesso</i> . ....	50
Figura 52 - Programação do nó <i>Escrita Servidor Erro</i> . ....	50
Figura 53 - Adicionando o servidor criado no Node-RED ao UA Expert. ....	51
Figura 54 - Informando as configurações do servidor criado no Node-RED ao UA Expert. ....	51
Figura 55 - UA Expert conectado ao servidor OPC UA criado no Node-RED. ....	52
Figura 56 - Fluxograma do funcionamento do percurso da peça conforme escrita. ....	52
Figura 57 - Programação da definição das <i>Tags IDs</i> e suas paradas. ....	53
Figura 58 - Configuração do nó <i>tag id=0xBB22</i> . ....	54
Figura 59 - Parte da programação que faz a confirmação da escrita de uma <i>Tag ID</i> que possui as paradas no FMS configuradas. ....	54
Figura 60 - Programação do nó <i>escrita realizada true</i> . ....	55
Figura 61 - Programação do nó <i>status cart 34</i> . ....	55
Figura 62 - Configuração do nó <i>junte 2</i> . ....	55
Figura 63 - Programação feita para a requisição da peça conforme a escrita com manipulação da peça. ....	56
Figura 64 - Programação do nó <i>Confirmação Cart parado</i> . ....	57
Figura 65 - Programação do nó <i>Parar no Processing true</i> . ....	57

Figura 66 - Programação do nó <i>Cart Processing</i> .....	58
Figura 67 - Programação do nó <i>Atividade Processing fim.</i> .....	58
Figura 68 - Programação do nó <i>Parar no processing false.</i> .....	58
Figura 69 – Programação da requisição da peça conforme escrita com parada por tempo na estação. ....	59
Figura 70 - Programação do nó <i>Parar no Vision true.</i> .....	59
Figura 71 - Programação do nó <i>Confirmação de propriedade</i> .....	60
Figura 72 - Programação do nó <i>Liberar Cart Vsision.</i> .....	60
Figura 73 - Programação do nó <i>Parar no vision false.</i> .....	61
Figura 74 - Reinicialização dos valores das variáveis <i>flow.</i> .....	61
Figura 75 - Programação do nó <i>Finalizar Sorting</i> .....	62
Figura 76 - Programação do nó <i>escrita realizada false.</i> .....	62
Figura 77 - Programação do nó <i>Resetar as variáveis flow.</i> .....	62
Figura 78 - Dashboard com os comandos de inicialização do leitor RFID e suas confirmações. ....	63
Figura 79 - Leitura das peças com <i>tag</i> apresentadas na Dashboard do Node-RED. ....	64
Figura 80 – Leitura manual e escrita da <i>Tag ID pela Dashboard</i> do Node-RED.....	65
Figura 81 - Varáveis de leitura e escrita do servidor OPC UA apresentadas na janela de Data Access View do software UA Expert .....	65
Figura 82 - Indicação na Dashboard de sucesso na escrita pelo servidor.....	66
Figura 83 - Funcionamento do sistema com o valor 0xAA11.....	66
Figura 84 - Funcionamento do sistema com o valor 0xBB11 .....	67

## **LISTA DE TABELAS**

Tabela 1 -Nome, tipo e funcionalidades dos nós utilizados no Node-RED relacionados aos dados pertinentes do FMS para a realização do projeto. ....	24
Tabela 2 - Comandos utilizados para a configuração do leitor RDID.....	29
Tabela 3 - Lista de variáveis criadas no servidor OPC UA. ....	39

## LISTA DE SIGLAS

CPS	- Sistemas Cíber Físicos
IoT	- Internet das Coisas
IoS	- Internet dos Serviços
CLP	- Controlador Lógico Programável
UTR	- Unidade Terminal Remota
RFID	- Radio Frequency Identification
IFF	- Identify Friend or Foe
OPC UA	- Open Platform Communication Unified Architecture
IBM	- International Business Machines Corporation
HTML	- HyperText Markup Language
CSS	- Cascading Style Sheet
JSON	- JavaScript Object Notation
FMS	- Flexible Manufacturing System
UNESP	- Universidade Estadual Paulista
UHF	- Ultra-high Frequency
CLI	- Command Line Interface
M2M	- Machine-to-Machine
IP	- Internet Protocol
ID	- Identification

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO.....</b>	<b>13</b>
1.1	OBJETIVOS .....	14
1.2	Estrutura do Trabalho .....	14
<b>2</b>	<b>REVISÃO DE LITERATURA .....</b>	<b>16</b>
2.1	TECNOLOGIA RFID .....	16
2.2	COMUNICAÇÃO OPC UA .....	17
2.3	NODE-RED.....	20
<b>3</b>	<b>INTEGRAÇÃO RFID E OPC UA APLICADO AO FMS.....</b>	<b>22</b>
3.1	Materiais e Métodos .....	22
3.1.1	FMS FESTO .....	22
3.1.2	Programação CLP Node-RED.....	23
3.1.3	Leitor RFID INfinity 510 .....	25
3.1.4	UA Expert.....	27
<b>4</b>	<b>DESENVOLVIMENTO .....</b>	<b>29</b>
4.1	Configuração do leitor RFID.....	29
4.2	Leitura e escrita pelo Dashboard do Node-RED .....	33
4.2.1	Leitura via Dashboard .....	33
4.2.2	Escrita via Dashboard.....	35
4.3	Leitura e escrita pelo servidor/cliente OPC UA .....	37
4.3.1	Criação do servidor e das variáveis .....	37
4.3.2	Leitura pelo Servidor OPC UA. ....	40
4.3.3	Escrita pelo Servidor OPC UA.....	46
4.3.4	Configuração UA Expert .....	51
4.4	Configuração do percurso da peça no FMS com base escrita. ....	52
4.4.1	Confirmação da realização da escrita. ....	52
4.4.2	Configuração da requisição com manipulação da peça.....	56
4.4.3	Configuração da requisição com parada por tempo .....	58
<b>5</b>	<b>RESULTADOS .....</b>	<b>63</b>
5.1	Inicialização do leitor RFID .....	63
5.2	Leitura e Escrita Dashboard .....	64

5.3	Leitura e Escrita Servidor OPC UA .....	65
5.4	Percurso da peça FMS. ....	66
5.4.1	Peça 0xAA11 .....	66
5.4.2	Peça 0xBB22 .....	67
5.5	Desafios e Discussões.....	67
<b>6</b>	<b>CONCLUSÃO .....</b>	<b>69</b>
	<b>REFERÊNCIAS .....</b>	<b>70</b>

## 1 INTRODUÇÃO

O crescimento e avanço da automação para melhorar a capacidade dos processos industriais é muito evidente, pois tais melhorias trouxeram redução de custos e desperdícios, aumento na qualidade, redução de mão de obra e uma ergonomia mais saudável às empresas. Com o desdobramento da automação industrial, o número de acidentes de trabalho reduziu, pois o trabalho braçal e repetitivo começou a ser realizados por máquinas, todavia essa substituição do homem pela máquina não acarretou no aumento do índice de desemprego, visto que a máquina precisa de pessoas capacitadas para seu monitoramento, controle e manutenção (XAVIER et al., 2023).

Devido a ascensão da automação industrial, ocorre a 4ª Revolução Industrial marcada pela presença da descentralização dos processos de manufatura, Sistemas Cíber Físicos (CPS), Internet das Coisas (IoT) e Internet de Serviços (IoS). Neste contexto, nasce o termo Indústria 4.0, nome este que vem de um projeto de indústria na Alemanha em 2011 chamado de Plattform Industrie 4.0 (Plataforma Indústria 4.0). A Indústria 4.0 se baseia na assimilação de tecnologias de informação e comunicação para gerar meios que melhoram a produtividade, gerenciamento e qualidade de produtos e processos, por exemplo, em uma planta industrial a linha produtiva pode ter controle e acionamento remoto, acompanhado de um modelo virtual para facilitar a prevenção de erros. Ao mesmo tempo que pedidos feitos por clientes são organizados automaticamente, sendo possível para o cliente acompanhar online todas as etapas de produção do seu pedido (SACOMANO; GONÇALVES; BONILLA, 2018).

O monitoramento e rastreamento das informações do processo produtivo, dentro da ideia da Industria 4.0, é realizado pelos sistemas supervisórios. Estes sistemas são compostos por sensores, Controladores Lógicos Programáveis (CLPs), Unidade Terminal Remota (UTR), os atuadores e a rede de comunicação. É graças a ajuda dos sistemas supervisórios que as operações industriais conseguem o aumento dos padrões de qualidade, reduções da inatividade e a percepção de falhas (ALTUS, 2023).

A tecnologia RFID (*Radio Frequency Identification*) teve sua primeira aparição na Segunda Guerra Mundial, utilizada em sistemas de radares para identificar a aproximação de aviões, porém não era possível diferenciar entre aviões inimigos e aliados. Por conseguinte, os ingleses criaram o primeiro identificador ativo chamado de IFF (*Identify Freind or Foe*), transmissores eram instalados nos aviões britânicos que enviavam sinais de resposta para os radares no solo, portanto, se houvesse um sinal de reposta o avião era identificado como aliado,

caso contrário era identificado como inimigo. Com o desenvolvimento da arquitetura dos sistemas RFID, a tecnologia passou a realizar a identificação de forma automática por meio de um leitor com antenas que envia sinais de radiofrequência a um microchip, o qual reflete o sinal para o leitor ou transmite seu próprio sinal. Esta nova arquitetura da tecnologia RFID tornou possível sua aplicação em sistemas de rastreamento e localização de produtos (PINHEIRO, 2017).

Para facilitar a configuração dos sistemas supervisórios em diferentes máquinas e equipamentos foi criado o padrão de comunicação *Open Platform Communication Unified Architecture* (OPC UA), sendo assim um protocolo de comunicação universal de máquina com máquina. O OPC UA surgiu a partir do bom desempenho do OPC clássico, possuindo e melhorando todas as funções do seu antecessor e apresentando também novas características, sendo compatível com os sistemas operacionais mais populares, servidores em nuvem e um confiável sistema segurança com sessões criptografadas, assinatura de mensagens e sistema de autenticação para os servidores (OPC FOUNDATION, 2017).

## 1.1 OBJETIVOS

É notável a importância que a Indústria 4.0 vem trazendo atualmente para a automação industrial, fornecendo um total monitoramento e rastreamento em tempo real de todos os processos, desde a produção até a entrega ao cliente, ao mesmo tempo que fornece uma segurança de acesso às informações, portanto o principal objetivo deste trabalho é fazer a integração de um sistema supervisório em um sistema de manufatura utilizando a tecnologia RFID e a comunicação OPC UA.

De forma específica, outros objetivos que o trabalho visa são a configuração de inicialização do leitor RFID e das suas antenas de forma automática, a leitura em tempo real da posição das peças com *tags* e assim como a possibilidade de ações de escrita na *tag*, em que as informações de leitura e ações de escrita poderão ser acessadas e executadas por uma Dashboard e pelo servidor OPC UA. Ademais com base no valor da escrita serão mapeadas diferentes rotas pelo sistema de manufatura.

## 1.2 Estrutura do Trabalho

Inicialmente é apresentado a revisão de literatura necessária para o entendimento das tecnologias utilizadas no projeto. Em seguida, é apresentado os materiais e métodos utilizados



para a realização do projeto. Logo após, é descrito todo o desenvolvimento do projeto destacando as programações e funções utilizadas dentro do ambiente do Node-RED. Depois disso, são apresentados e discutidos os resultados obtidos do projeto, destacando maiores desafios encontrados durante o desenvolvimento e, por fim, é apresentado uma conclusão do trabalho destacando os objetivos alcançados e uma possível melhoria a ser implementada ao projeto.

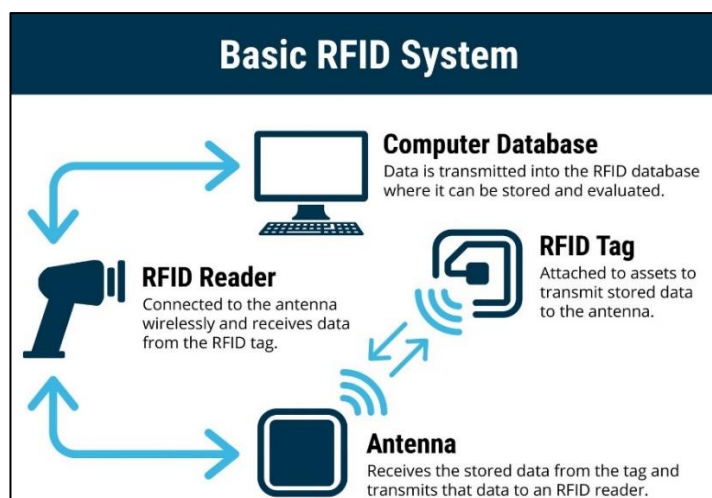
## 2 REVISÃO DE LITERATURA

Nesta seção serão abordados os conceitos e tecnologias utilizados para a execução deste projeto final de curso e alcançar os objetivos propostos, divididos nos seguintes tópicos: Tecnologia RFID, Comunicação OPC-UA (*Open Platform Communication Unified Architecture*), Node-RED

### 2.1 TECNOLOGIA RFID

A ideia principal dos sistemas RFIDs é que um objeto qualquer é identificado com uma *tag*. Essas *tags* possuem os *Transponders* (ou etiquetas RFID) que emitem mensagens em rádio frequência que são captadas pelas antenas do leitor RFID fornecendo informações como número de identificação, data e hora do sinal captado, dados de usuário e entre outras informações. O leitor possui seu próprio banco de dados, em que é organizado com base na identificação da etiqueta. As etiquetas também possuem memória gravável, possibilitando a inserção e modificação de dados, o que possibilita um melhor rastreamento e controle de objetos que possuem uma etiqueta RFID (WEINSTEIN, 2005). A Figura 1 demonstra um sistema básico do RFID e suas interações.

Figura 1 - Sistema básico da utilização da tecnologia RFID.



Fonte: TT Electronics (2021)

As *tags* de RFID possuem diferentes formas e tamanhos dependendo de seu uso e aplicação. Podendo ter a dimensão de um cartão de crédito, como as que são usadas para uso de controle de acesso de pessoas, ou terem menos 3 mm, como as utilizadas na identificação de animais ou peças microeletrônicas. Os formatos das *tags* podem ser inúmeros, entre elas tem-se: formato de varetas, pastilhas, cartão, prego, moeda etc. (MEDEIROS, 2011).

As *tags* são classificadas em duas categorias, ativas e passivas. As *tags* ativas possuem sua própria fonte de alimentação com baterias, uma transmissão de sinal mais forte possibilitando o leitor acessar as informações emitidas pela etiqueta de uma distância consideravelmente maior, trabalham com sinais em altas frequências e a presença da bateria torna as *tags* maiores em tamanho e com um custo maior. As *tags* passivas são mais baratas e menores, conseguem trabalhar em altas e baixas frequências, sua capacidade de memória é menor em relação à ativa e seu sinal emitido não consegue percorrer distâncias muito grandes (WEINSTEIN, 2005).

Uma das maiores vantagens da tecnologia RFID é a sua confiabilidade, visto que com a presença dos *Transponders* a sua leitura é simples, não necessitando de uma posição específica e não dependendo de nenhum contato físico ou elétrico, eliminando problemas como sujeira, desgaste ou oxidação. Outros pontos positivos são a leitura simultânea de vários itens diferentes, reutilização e alta durabilidade das etiquetas e maior segurança em tarefas repetitivas. (PINHEIRO, 2017).

## 2.2 COMUNICAÇÃO OPC UA

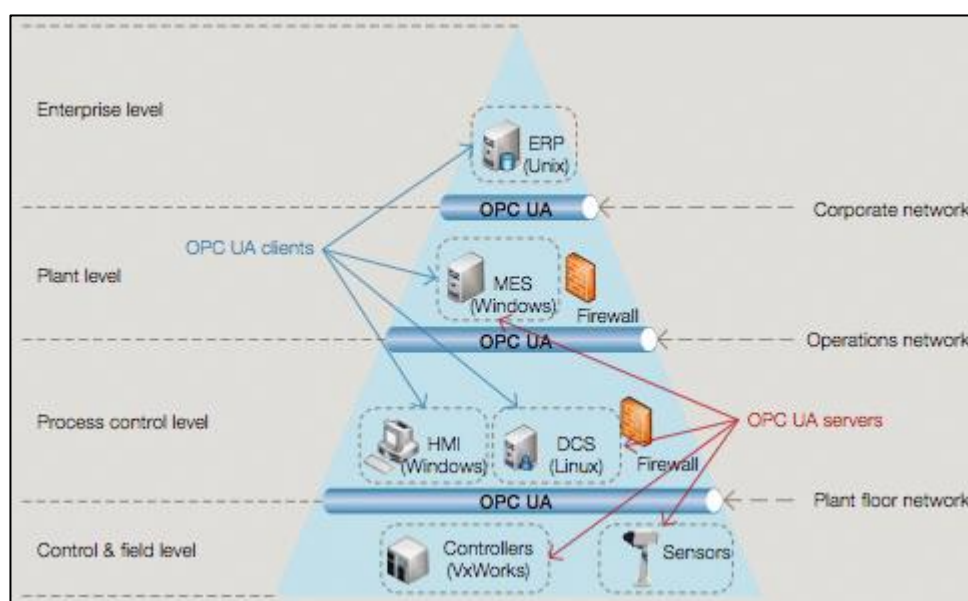
O OPC conhecido como clássico foi criado por volta de 1990, pela *OPC Foundation* sendo definido como um padrão de interoperabilidade para a troca segura de informações e dados na área da automação industrial. Na época o serviço era compatível apenas com as tecnologias da *Microsoft*, proporcionando uma série de métodos entre servidores e clientes com funcionalidades como leitura e escrita de variáveis, monitoramento e histórico de dados (MELO, 2020). O OPC UA foi lançado em 2008 herdando as funcionalidades do OPC clássico, porém sem exclusividade de funcionamento com a *Microsoft*, tornando-se funcional em multiplataformas em uma estrutura extensível. As principais características do OPC UA são: (OPC FOUNDATION, 2017).

- Funções equivalentes: todas as funcionalidades do OPC Clássico estão presentes no OPC UA.
- Independência de plataforma: mudança de microcontroladores para uma infraestrutura baseada em nuvem.
- Mais segurança: criptografia, autenticação e audição de dados.
- Extensível: possibilidade de adição de novas ferramentas e funcionalidades sem afetar aplicações existentes.

- Ampla Modelagem de Informações: possibilitando a manipulação de dados complexos.

Como o protocolo OPC UA possui uma arquitetura baseada em padrões de tecnologia Web, com ele é possível fazer a comunicação entre clientes e servidores utilizando os mais diversos tipos de redes, o que faz o protocolo ser um poderoso criador de modelos de informações. No servidor, estes modelos de informações definem um padrão organizando os dados dentro de um espaço de endereço (*AddressSpace*), fazendo com que esta estrutura seja perfeita para a integração vertical dos sistemas de automação. Com isso, a comunicação OPC UA pode ser adotada em vários componentes do setor industrial, com o objetivo de haver trocas de informações entre sensores, atuadores e os sistemas de fabricação, planejamento e controle (MELO, 2020). A Figura 2 ilustra esse sistema de troca de informações proporcionado pela comunicação OPC UA.

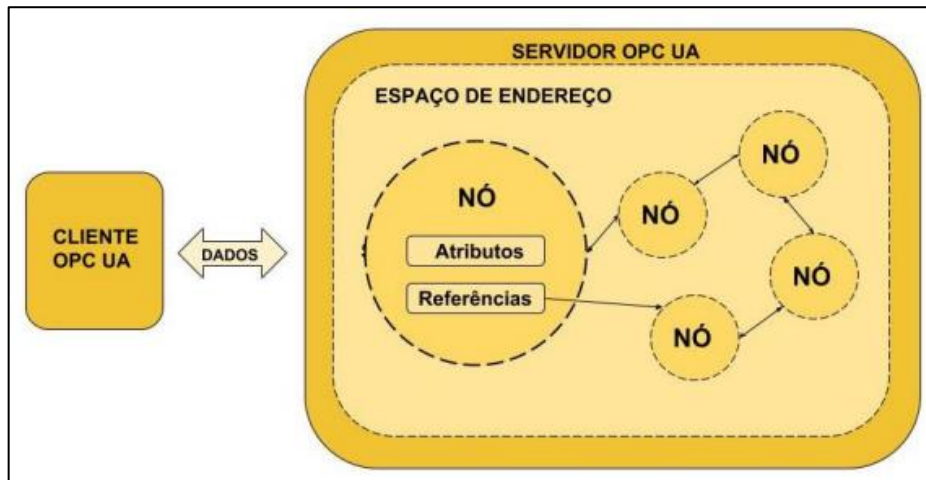
Figura 2 - Aplicação do padrão OPC UA em um sistema de automação industrial.



Fonte: Melo (2017).

Um dos mais importantes conceitos do OPC UA é o espaço de endereço. Nos espaços de endereço todos os dados são organizados em uma hierarquia permitindo estruturas complexas e simples a serem utilizadas pelos clientes, tal como oferecer diferentes acessos de permissão para os dados de leitura e escrita (OPC FOUNDATION, 2017). O espaço de endereço pode ser representado por um conjunto de Nós (*Nodes*), suas características representadas por Atributos e cada Nó pode estar conectado entre si por meio de Referências. As Referências têm a função exclusiva de fazer a ligação entre Nós para haver a corrente de informações (MELO, 2020). A Figura 3 ilustra o endereço de espaço de um servidor OPC UA de forma geral.

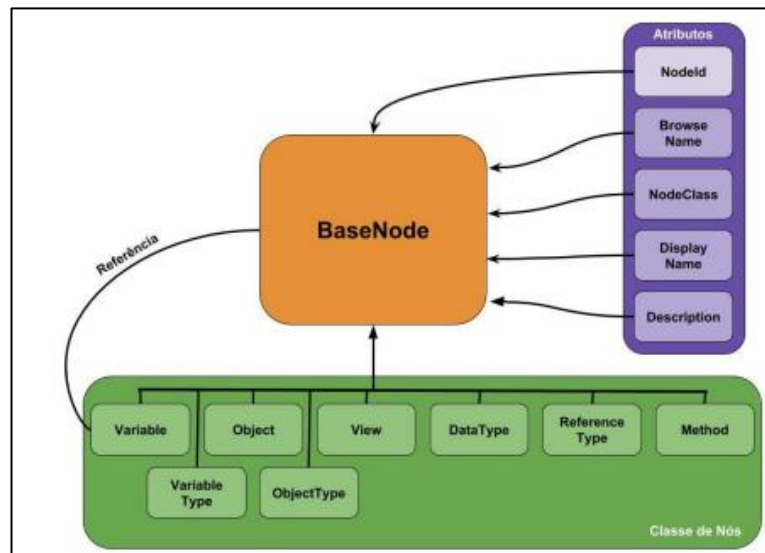
Figura 3 - Espaço de endereço no servidor OPC UA.



Fonte: Melo (2020).

Existem diversas classes de Nós, sendo eles: *Variable*, *Object*, *View*, *DataType*, *ReferenceType*, *Method*, *VariableType* e *ObjectType*. Todas essas classes de Nós são derivadas do chamado Nó Base (BaseNode). É pelo Nó Base que os Atributos são passados para as demais classes de Nós, como é ilustrado na Figura 4 (MELO, 2020).

Figura 4 - Nó Base e suas classes derivadas.



Fonte: Melo (2020).

Os Atributos são os componentes que caracterizam os Nós, é a partir deles que é possível fazer a leitura e escrita de dados, ressaltando que dependendo da derivação do Nó os tipos de Atributos mudam, porém existe Atributos que estão presentes em todos os Nós, sendo eles: (MELO, 2020).

- *NodeId*: identificação exclusiva de um nó.
- *BrowseName*: identificador de um nó quando buscado no espaço de endereço.

- *NodeClass*: numeração que identifica a classe do Nó.
- *Description*: descrição em forma de texto do Nó.
- *DisplayName*: nome do Nó que é apresentado em uma interface de usuário.

## 2.3 NODE-RED

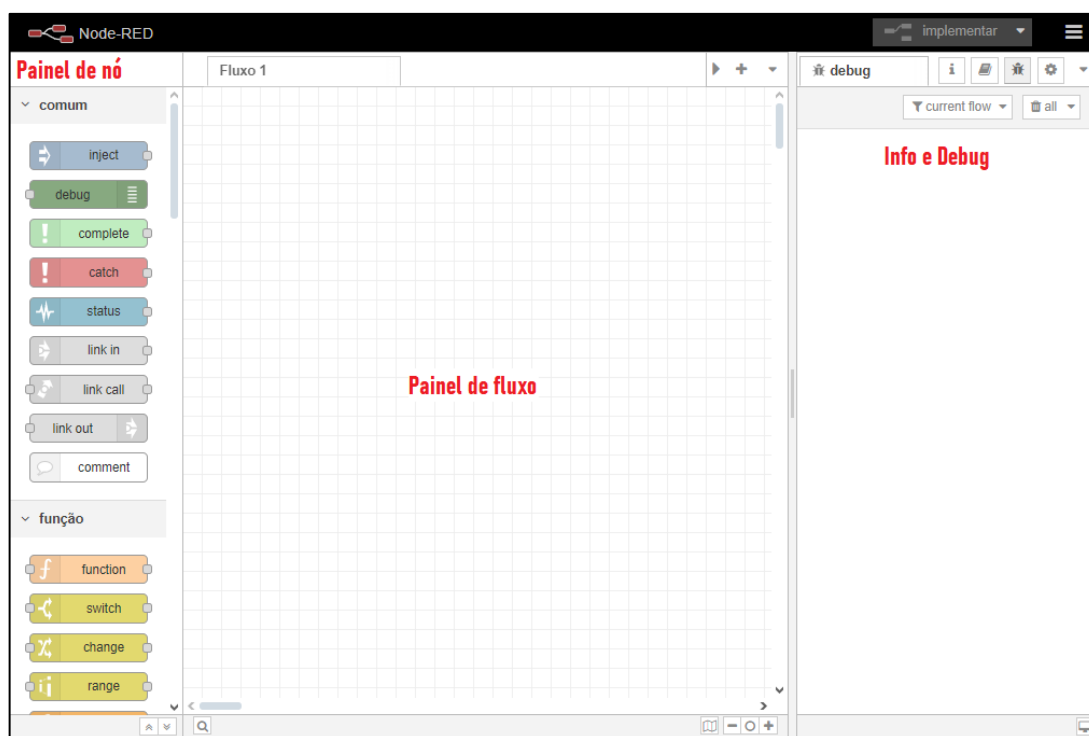
Desenvolvido por engenheiros da IBM em 2013, o Node-RED é um ambiente de desenvolvimento aberto ao público baseado em *JavaScript* e *Node.js*, sendo ideal para o desenvolvimento de sistemas IoT (FERENCZ; DOMOKOS, 2019). O seu ambiente de programação virtual é baseado na criação de fluxo de dados interligando blocos de funções definidas, blocos esses denominados nós, sendo a própria rede responsável pelo fluxo de dados entre os nós (OPENJS FOUNDATION & CONTRIBUTORS, 2019). Com sua lógica de funcionamento, fica simples realizar o manuseio de dados e a transferência para sistemas de alto nível (servidores, central de coleta de dados, serviços baseados em nuvem) em questão de minutos ou até instantaneamente. O Node-RED também conta com uma interface *Dashboard* editável que possibilita criar uma apresentação dos dados de forma textual ou gráfica sem a necessidade de conhecimentos de HTML e CSS (FERENCZ; DOMOKOS, 2019).

Inicialmente quando criado, o Node-RED foi anunciado como uma ferramenta visual projetada para as necessidades da Internet das Coisas, mas hoje em dia é uma ferramenta para os mais diversos usos. Pode ser usada para interface de dados de sensores, comunicação com e sem fio, conexão com a nuvem, alertas de mídias sociais, análise de dados, sistemas de monitoramento e qualquer implementação que possua os serviços de funcionalidade *Node.js*. O Node-RED possibilita os desenvolvedores, inclusive aqueles com pouca experiência, a criar códigos reutilizáveis e fazer protótipos de forma rápida com uma interface de fácil uso disponível em navegador web. Por isso, devido a sua simplicidade, permite que desenvolvedores de aplicações CPS foquem em identificar e desenvolver soluções ao invés de construir os componentes da aplicação do zero (FIAIDHI; MOHAMMED, 2021).

O Node-RED possui 3 elementos básicos, sendo eles o painel de nó, o painel de fluxo e o painel de Info e Debug, como é ilustrado na Figura 5. Cada nó fornece uma função diferente que cria e/ou armazena os dados em forma de objetos ou cadeia de caracteres *JSON*, em que *JSON* é uma forma de representação dos objetos dentro do *JavaScript* que facilita a aplicação em diversos protocolos (FERENCZ; DOMOKOS, 2019). Duas propriedades importantes dos nós são o *msg.payload* e o *msg.topic*. O *msg.payload* é a propriedade padrão que a maioria dos

nós trabalha, é nela que são armazenados os dados enviados e movimentos pelos fluxos em forma de mensagens, podendo ser uma simples *String* ou até mesmo um Objeto com diversas classes. O *msg.topic* é utilizado como um identificador da fonte das mensagens ou pode ser utilizado para identificar diferentes tipos de mensagens em um mesmo fluxo, por exemplo, dados meteorológicos de temperatura, umidade e velocidade do vento são enviados de uma mesma fonte para uma aplicação no Node-RED, definindo um *msg.topic* diferente para cada dado, não haverá problemas em analisar os dados separadamente mesmo eles vindo de uma mesma fonte (mesmo fluxo de mensagens) (OPENJS FOUNDATION & CONTRIBUTORS, 2019).

Figura 5 - Ambiente de desenvolvimento do Node-RED.



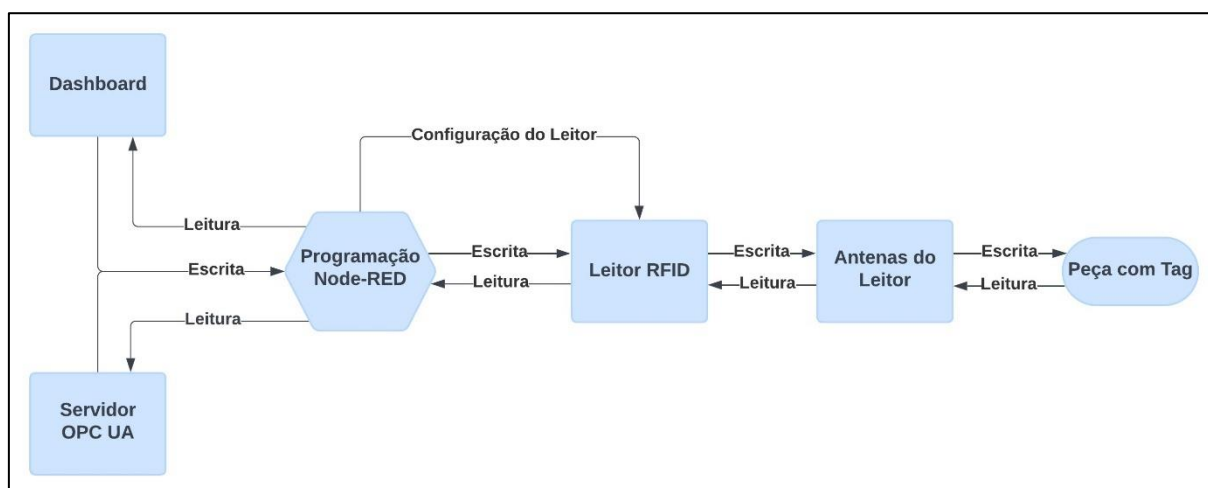
Fonte: Autoria Própria.

### 3 INTEGRAÇÃO RFID E OPC UA APLICADO AO FMS

O FMS utilizado são os módulos da FESTO<sup>1</sup> instalados no Laboratório de Automação II da UNESP Campus de Sorocaba. O leitor RFID utilizado foi o INfinity 510 desenvolvido pela empresa Sirit<sup>2</sup>. Toda a programação do trabalho foi desenvolvida a partir do ambiente do Node-RED. Para validar e testar o cliente OPC UA com a leitura e escrita das variáveis foi utilizado o software UA Expert desenvolvido pela Unified Automation<sup>3</sup>.

A Figura 6 apresenta a idealização de um fluxo do funcionamento do projeto, destacando a trajetória dos dados de escrita e leitura das *tags* e os dados relacionados à configuração automática do leitor.

Figura 6 - Fluxograma apresentando o funcionamento geral do projeto.



Fonte: Autoria Própria.

#### 3.1 Materiais e Métodos

##### 3.1.1 FMS FESTO

O sistema FESTO de manufatura flexível presente na UNESP Sorocaba, além de ser utilizado para a realização de trabalho e pesquisas, é usado também para fins didáticos em disciplinas ministradas no campus, o que torna bem convencional a sua característica modular. No atual momento da realização deste projeto, o FMS possui 6 estações, com os seguintes módulos de processos em cada estação:

<sup>1</sup> Site FESTO: <https://www.festo.com/us/en/>

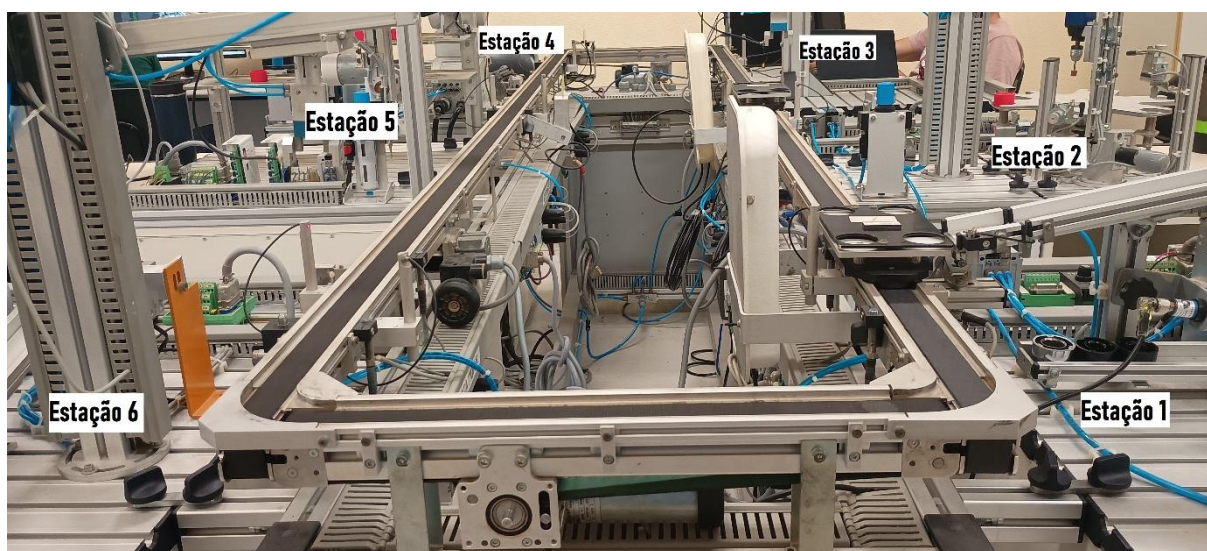
<sup>2</sup> LinkedIn Sirit (Site oficial fora do ar): <https://www.linkedin.com/company/sirit/>. Informações Sirit: <https://pitchbook.com/profiles/company/53727-58#overview>

<sup>3</sup> Site Unified Automation: <https://www.unified-automation.com>



- Estação 1: Módulos do *Distribution* e *Testing*. No *Distribution* a peça é separada de uma pilha e posicionada no *Testing*. No *Testing* ocorre o reconhecimento da peça;
- Estação 2: Módulo do *Handling II* e *Processing*. O *Handling II* manipula a peça do carrinho para o *Processing* e vice-versa. No *Processing* a peça é posicionada em uma mesa giratória que simula processos industriais;
- Estação 3: Módulo do *Vision*. O *Vision* bate uma foto da(s) peça(s) no próprio carrinho;
- Estação 4: Módulo do *Robot Arm*. Um braço robótico que manipula e pode organizar as peças por cor;
- Estação 5: Módulo do *Storage*. O *Storage* simula uma estação de estoque de peças;
- Estação 6: Módulos do *Handling I* e *Sorting*. O *Handling I* manipula a peça do carrinho até o *Sorting*. No *Sorting* as peças são separadas por cores por uma esteira elétrica e sensores.

Figura 7 – FMS da FESTO presente no laboratório de automação da UNESP campus Sorocaba.



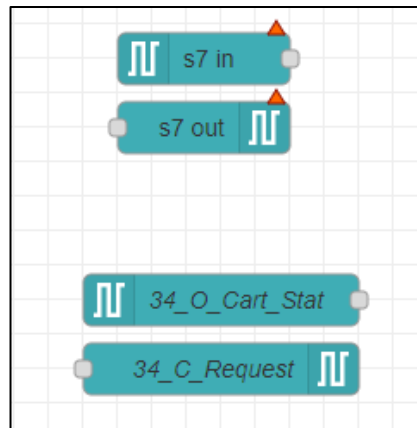
Fonte: Autoria Própria.

### 3.1.2 Programação CLP Node-RED

Este projeto será integrado com outros existentes e futuros, a vista disso, é utilizado para seu desenvolvimento informações e o estado de sensores do FMS via Node-RED que são fornecidos por projetos anteriormente elaborados em que este será integrado. A paleta de nós utilizada para o trânsito dessas informações é denominada *node-red-contrib-s7* que possui nós

para controle de dados de CLPs, a Figura 8 apresenta nós utilizados dessa paleta. O *s7 in* tem a função de ler os dados de um CLP e o *s7 out* escreve os dados em um CLP, *34\_O\_Cart\_Stat* e *34\_C\_Request* são exemplos de como foram denominados os nós utilizados no controle dos dados do FMS. A Tabela 1 apresenta nós que foram utilizados para este projeto acompanhado de sua funcionalidade (SOUZA; GODOY; CALDANA, 2024).

Figura 8 - Nós utilizados no Node-RED para o controle de dados do FMS.



Fonte: Autoria própria.

Tabela 1 -Nome, tipo e funcionalidades dos nós utilizados no Node-RED relacionados aos dados pertinentes do FMS para a realização do projeto.

Nome	Tipo	Funcionalidade
34_O_Cart_Stat	s7 in	Verifica se o carrinho está parado na estação 1.
34_C_Request	s7 out	Requisita uma parada do carrinho na estação 1.
35_O_Cart_Stat	s7 in	Verifica se o carrinho está parado na estação 2.
35_C_Request	s7 out	Requisita uma parada do carrinho na estação 2.
36_O_Cart_Stat	s7 in	Verifica se o carrinho está parado na estação 3.
36_C_Request	s7 out	Requisita uma parada do carrinho na estação 3.
37_O_Cart_Stat	s7 in	Verifica se o carrinho está parado na estação 4.
37_C_Request	s7 out	Requisita uma parada do carrinho na estação 4.
38_O_Cart_Stat	s7 in	Verifica se o carrinho está parado na estação 5.
38_C_Request	s7 out	Requisita uma parada do carrinho na estação 5.
39_O_Cart_Stat	s7 in	Verifica se o carrinho está parado na estação 6.
39_C_Request	s7 out	Requisita uma parada do carrinho na estação 6.

Fonte: Autoria Própria.

### 3.1.3 Leitor RFID INfinity 510

O *INfinity* 510 é um leitor de alta performance de Rádio Frequência multiprotocolo e multirregional que opera na faixa entre 860 e 960 MHz UHF (Ultra-high Frequency), suporta até quatro antenas de leitura e escrita, possui interface de entrada serial e interface Ethernet. Também possui entradas e saídas digitais. (SIRIT INC, 2010). O leitor está instalado junto ao FMS e suas informações são passadas através da comunicação Ethernet.

O leitor RFID possui quatro indicadores luminosos no topo da sua estrutura, como ilustra a Figura 9:

- *Sense*: indica que o leitor detectou uma *tag* no alcance de uma de suas antenas;
- *Transmit*: indica que o transmissor de dados do leitor está operando;
- *Fault*: indica que uma falha ocorreu;
- *Power*: indica que o leitor está ligado;

Figura 9 - Indicadores luminosos presente na estrutura superior do leitor RFID INfinity 510.



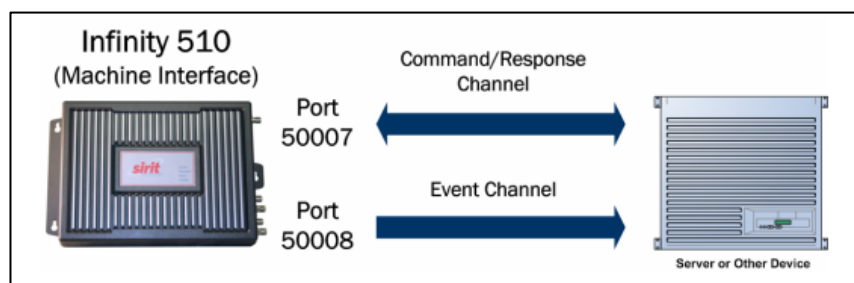
Fonte: Sirit Inc (2010)

Seu funcionamento é bem versátil, operando na maioria das aplicações de sistemas RFID. Todas as funções e controle de parâmetros são facilmente configurados através do envio de Interface de Linha de Comandos (CLI) para o canal de comando do leitor. Todos os comandos do leitor são organizados através do uso de *namespaces*, proporcionado um fácil entendimento da hierarquia e finalidade dos comandos, por exemplo, comandos relacionados a comunicação do leitor estão presentes no *namespace com.* (*com.serial*, *com.network*, *com.serial.baudrate*)(. (SIRIT INC, 2008).

Para realizar a comunicação Machine-to-Machine (M2M) pela conectividade Ethernet, este leitor RFID possui dois endereços de porta, as portas 50007 e 50008. O recebimento de comandos para o leitor é realizado pela porta 50007, todo comando que é enviado para o leitor gera uma resposta pela mesma porta alertando se o comando foi realizado com sucesso ou não.

A porta 50008 é uma porta que apenas envia (unidirecional) dados de eventos assíncronos coletados pelo leitor, tal como erros, chegada de *tags* no alcance das antenas, saída de *tags* do alcance das antenas e acionamentos das suas saídas digitais (SIRIT INC, 2008).

Figura 10 – Endereços de porta e suas funcionalidades para a comunicação M2M do leitor INfinity 510.



Fonte: Sirit Inc (2008).

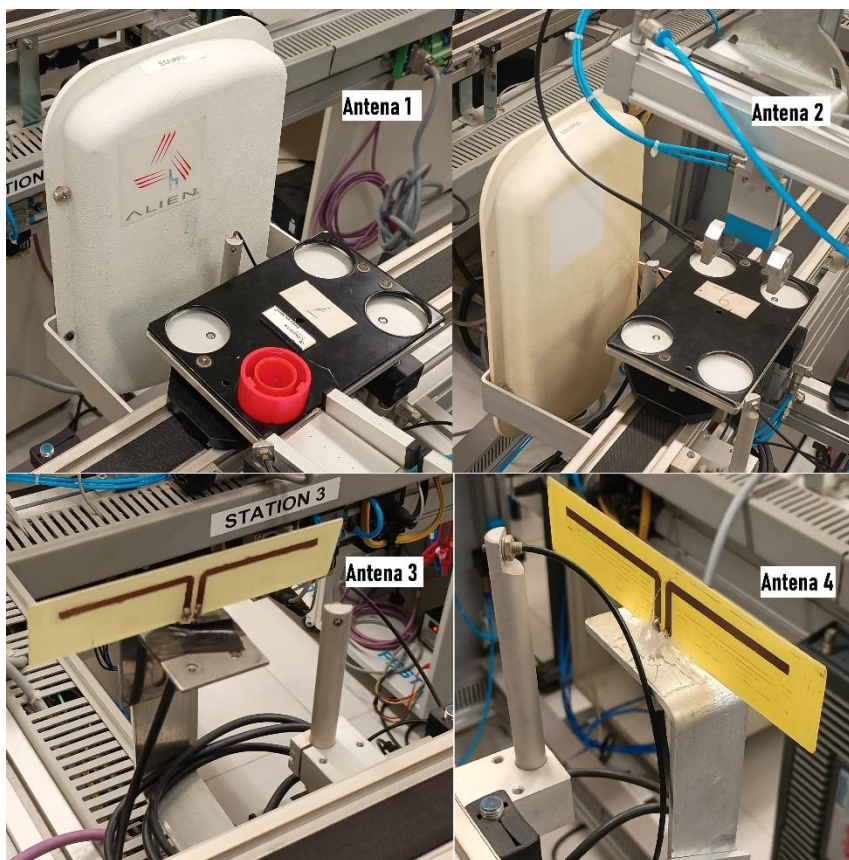
O leitor utilizará 4 antenas posicionadas em diferentes estações do FMS. Antena 1 e antena 2 são do modelo ALR-9610-AL fabricado pela empresa Alien Technology<sup>4</sup>, trabalham em uma frequência de 890 a 930 MHz com uma polarização do tipo linear, possuindo 28,4 cm de altura, 19,5 de comprimento e 4,3 de largura (ALIEN TECHNOLOGY, 2004). As antenas 3 e 4 serão antenas do tipo dipolo, foram originalmente fabricadas para uma tese de mestrado utilizando as máquinas de prototipagem disponíveis no Laboratório de Automação II da UNESP Campus Sorocaba (SANCHES, 2018), sendo agora reaproveitadas para este projeto.

O posicionamento das antenas no FMS está da seguinte forma:

- Antena 1: Estação 1 (*Testing*).
- Antena 2: Estação 2 (*Processing*).
- Antena 3: Estação 4 (*Robot Arm*).
- Antena 4: Estação 5 (*Storage*).

<sup>4</sup> Site Alien Technology: <https://www.alientechnology.com>

Figura 11 - Foto das antenas posicionadas em frente as estações do FMS.



Fonte: Autoria Própria.

### 3.1.4 UA Expert

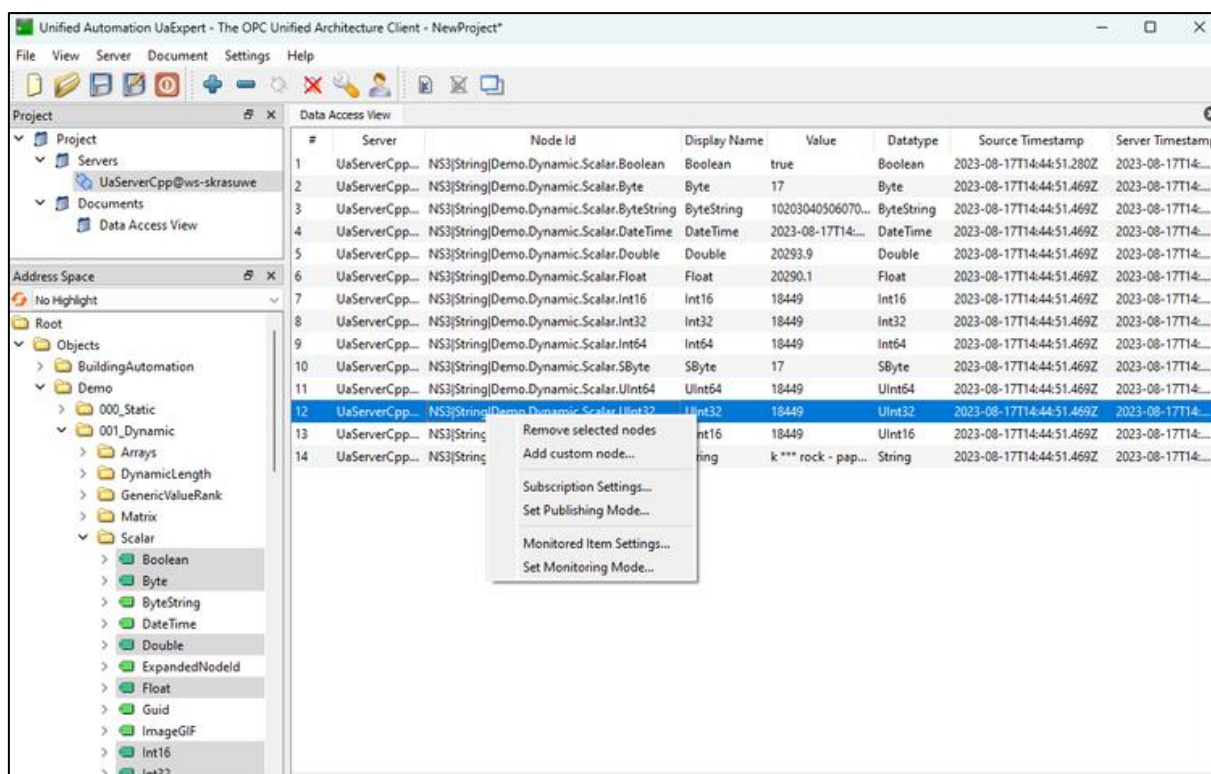
O UA Expert é um software projetado para o uso de testes de um cliente OPC UA multiplataforma, programado em C++ com uma estrutura básica que pode ser estendida com implementação de Plugins. Porém sua versão gratuita vem com plugins instalados fornecendo os principais recursos do OPC UA, sendo eles (UNIFIED AUTOMATION GMBH, 2023):

- Visualização de Acesso de Dados OPC UA.
- Visualização de Alarmes e Condições OPC UA.
- Visualização de Histórico OPC UA.
- Visualização de Integração de Dispositivo OPC UA.
- Visualização de Imagem.
- Visualização de Diagnóstico de Servidor
- Plugin Simples de Registrador de Dados
- Plugin de Performance OPC UA



Neste projeto, o recurso do software mais utilizado será a visualização de acesso de dados, para testes e validações dos dados e variáveis do servidor. A Figura 12 é um exemplo de tela da sua utilização, é possível arrastar, na janela de *Address Space*, as variáveis da pasta *Scalar* para a janela de *Data Access View* e visualizar as informações dos dados. No exemplo da figura é possível ver o *Node Id*, o nome da variável (*Display Name*), seu valor (*Value*), o tipo de variável (*Datatype*), o tempo de alteração da variável pela fonte do dado (*Source timestamp*) e o tempo que o servidor recebeu o valor da variável (*Server timestamp*). A partir dessa tela também é possível alterar os dados escrevendo no campo de *Value* da variável. Pela janela *Project*, é possível fazer as configurações de conexão do servidor na seção *Servers* e a configuração dos plugins é feita na seção abaixo, *Documents*.

Figura 12 - Exemplo de tela de visualização de acesso de dados OPC UA pelo UA Expert.



Fonte: Unified Automation GmbH (2023).

## 4 DESENVOLVIMENTO

Nesta seção são detalhados os procedimentos realizados para o desenvolvimento do projeto, separando por etapas: configuração do leitor RFID, leitura e escrita pelo Dashboard do Node-RED, leitura e escrita pelo servidor/cliente OPC UA e configuração do percurso da peça no FMS com base escrita.

### 4.1 Configuração do leitor RFID

As configurações do leitor RFID foram feitas através da sua Interface de Linha de Comandos (CLI), o Node-RED envia toda a configuração necessária em uma única mensagem assim que a comunicação com a porta 50008 é estabelecida com o Node-RED. Toda vez que está comunicação é realizada, o leitor envia através da porta 50008 o chamado *Connection ID* que é uma identificação de 2 dígitos utilizada para configurar os dados de eventos (chegada e saída de *tags*) enviados pela porta 50008.

A lista de aspectos necessários a serem configurados pelo Node-RED ao leitor para o seu funcionamento ideal para o projeto são:

- O modo de operação do leitor;
- Configuração dos parâmetros físicos;
- Configuração do consumo de cada antena, quanto maior o consumo maior o alcance;
- Configuração das informações recebidas quando uma *tag* entra no alcance das antenas;
- Configuração das informações recebidas quando uma *tag* sai do alcance das antenas;
- Registrar o canal de eventos com o *Connection ID* (porta 50008) para que as informações possam ser transmitidas;

A Tabela 2 apresenta a lista de comandos CLI utilizada e a suas funcionalidades

Tabela 2 - Comandos utilizados para a configuração do leitor RDID

Comando	Função
<i>setup.operation_mode=autonomous</i>	Configura o modo de operação do leitor no modo <i>autonomous</i> , fazendo funcionar toda vez que uma <i>tag</i>

	entre no alcance das antenas sinalizando de forma assíncrona pelo canal de eventos.
<i>modem.protocol.isoc.physical.set</i> <i>(tari=tari_12_50,</i> <i>return_link_freq=LF160,</i> <i>data_1_lenght=d1_len_20,</i> <i>rt_modulation=rt_mod_pr,</i> <i>tr_enconding=tr_enc_fm0,</i> <i>interrogator_mode=dense)</i>	<p>Configuração dos parâmetros físicos.</p> <ul style="list-style-type: none"> <li>• <i>tari=tari_12_50</i> configura o comprimento de referência para o sinal baixo (0) para 12,5 microssegundos.</li> <li>• <i>return_link_freq=LF160</i> configura a frequência que as <i>tags</i> respondem para o leitor em 160 Kbps.</li> <li>• <i>data_1_lenght=d1_len_20</i> configura o comprimento de referência para o sinal alto (1) em 2 vezes maior que o sinal baixo.</li> <li>• <i>rt_modulation=rt_mod_pr</i> configura o tipo de modulação usado na transmissão entre o leitor e a <i>tag</i> para modo de amplitude reversa de fase.</li> <li>• <i>tr_enconding=tr_enc_fm0</i> configura a codificação usada entre a <i>tag</i> e o leitor para o modo padrão.</li> <li>• <i>interrogator_mode=dense</i> configura filtros regulatórios no leitor no modo denso, no modo denso todos os filtros são ativados.</li> </ul>
<i>antennas.X.conducted_power = Y</i>	Configuração a alimentação das antenas, onde X é o número da antena e Y é o consumo da antena em dBm (decibéis miliwatt), para as antenas 1 e 4 foi configurado 260 dBm, para antena 2 foi configurado 180 dBm e para a antena 3 200 dBm
<i>tag.reporting.arrive_fields=tag_id</i> <i>antenna time user_data</i> e <i>tag.reporting.depart_fields=tag_id</i> <i>antenna time user_data.</i>	Comandos que configuram o leitor RFID a toda vez que uma <i>tag</i> entrar ( <i>arrive</i> ) ou sair ( <i>depart</i> ) do alcance de uma antena mostrar através do canal de eventos as seguintes informações da <i>tag</i> : seu número



	de identificação <i>tag ID</i> , a antena que a captou, a data/hora da captação e o os dados de usuário escritos na tag.
<code>reader.events.register(XX, event.tag.arrive)</code> e <code>reader.events.register(XX, event.tag.depart)</code>	Registrar o canal de eventos com o <i>Connection ID</i> tanto para as informações de chegada quanto de saída de <i>tags</i> do alcance das antenas, em que XX é o <i>Connection ID</i> fornecido pela porta 50008 do leitor RFID

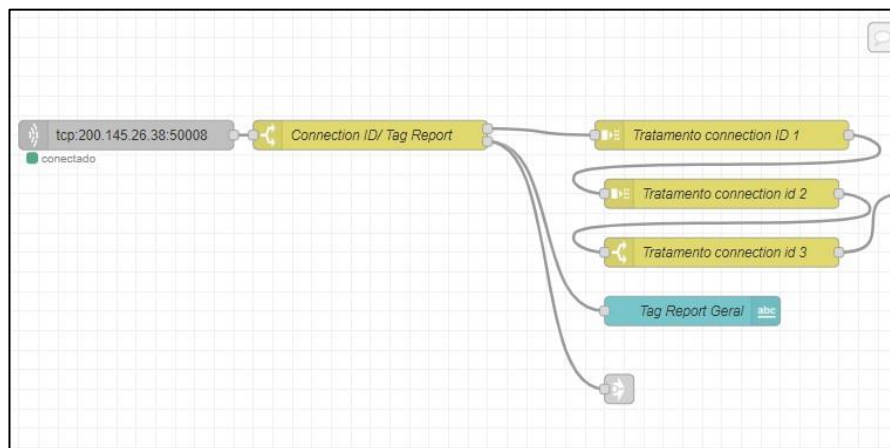
Fonte: Autoria própria.

Os valores adotados para a alimentação das antenas serão justificados no capítulo de resultados.

Cada comando CLI deve ser finalizado por `\r\n` caso contrário o leitor não identifica a mensagem como uma linha de comando.

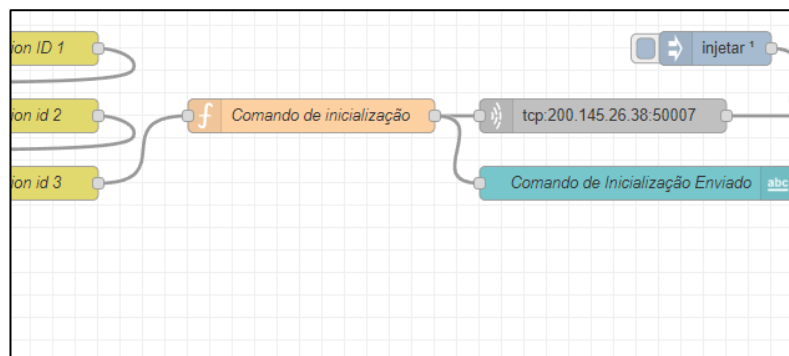
A Figura 13 e a Figura 14 apresentam a programação que foi realizada no Node-RED para a configuração automática do leitor.

Figura 13 - Programação feita para a configuração do leitor RFID via Node-RED parte I.



Fonte: Autoria própria.

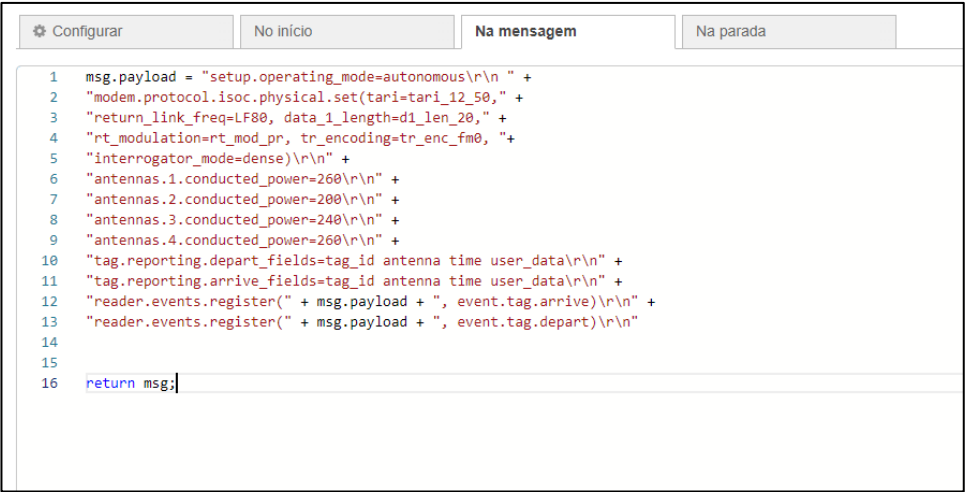
Figura 14 - Programação feita para a configuração do leitor RFID via Node-RED parte II.



Fonte: Autoria própria.

Sendo assim, o nó *tcp in* do Node-RED denominado *tcp:200.145.26.38:50008* (200.145.26.38 é o endereço de IP do leitor RFID na rede do Laboratório de Automação II da UNESP Campus Sorocaba) faz a conexão com a porta 50008 do leitor, a primeira informação recebida após a conexão é sempre o número do *Connection ID* e toda vez que uma nova conexão é estabelecida, um novo *Connection ID* é fornecido. Visto que pela porta 50008 são recebidas também as informações de chegadas e saídas das *tags* um nó *switch* denominado *Connection ID/Tag Report* separa os dois tipos de dados recebidos pela porta 50008 em duas diferentes saídas. A exata mensagem que o leitor envia é: *event.connection id = (um número de dois dígitos)*. Dois nós *split* e um nó *switch* (denominados respectivamente *Tratamento connection ID 1*, *Tratamento connection ID 2* e *Tratamento connection ID 3*) fragmentam e filtram a mensagem para que a informação passada nesse fluxo seja apenas o número de dois dígitos. Essa informação então chega a um nó *function* denominado *Comando de inicialização*, nele que são gerados e enviados os comandos CLIs de configuração do leitor. Os comandos são enviados para a porta 50007 do leitor através do nó *link call* denominado *tcp:200.145.26.38:50007*, os comandos também são enviados para a Dashboard do Node-RED através do nó *text* denominado *Comando de Inicialização Enviado*.

Figura 15 - Programação feita dentro do nó Comando de inicialização para o envio de comandos de configuração ao leitor RFID.



```

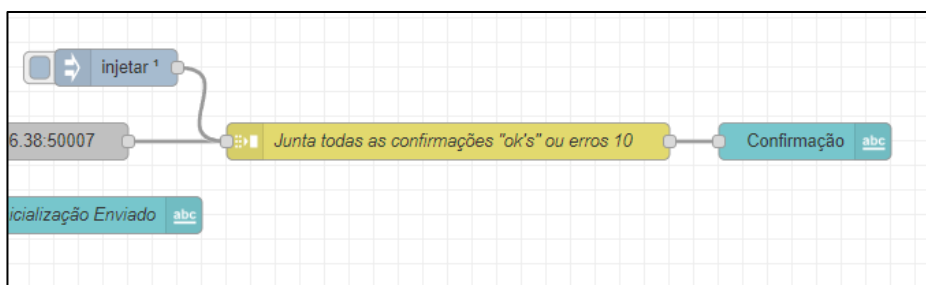
1 msg.payload = "setup.operating_mode=autonomous\r\n" +
2 "modem.protocol.isoc.physical.set(tari=tari_12_50," +
3 "return_link_freq=LF80, data_1_length=d1_len_20," +
4 "rt_modulation=rt_mod_pr, tr_encoding=tr_enc_fm0, "+
5 "interrogator_mode=dense)\r\n" +
6 "antennas.1.conducted_power=260\r\n" +
7 "antennas.2.conducted_power=200\r\n" +
8 "antennas.3.conducted_power=240\r\n" +
9 "antennas.4.conducted_power=260\r\n" +
10 "tag.reporting.depart_fields=tag_id antenna time user_data\r\n" +
11 "tag.reporting.arrive_fields=tag_id antenna time user_data\r\n" +
12 "reader.events.register(" + msg.payload + ", event.tag.arrive)\r\n" +
13 "reader.events.register(" + msg.payload + ", event.tag.depart)\r\n"
14
15
16 return msg;

```

Fonte: Autoria própria.

Todos os CLIs enviados ao leitor geram uma resposta informando o êxito ou o erro de execução de cada comando. Pela Figura 16 é mostrado que essas respostas são coletadas pelo nó *join* (denominado *Junta todas as confirmações "ok's" ou erros 10*) e enviadas através de uma única *string* para o Dashboard no nó *Confirmação*. Caso o comando tenha sido executado com êxito, é retornado um *ok*, caso não, é retornado o código do erro.

Figura 16 - Programação do envio das confirmações ou erros da execução dos comandos utilizados na configuração do leitor RFID para o Dashboard.



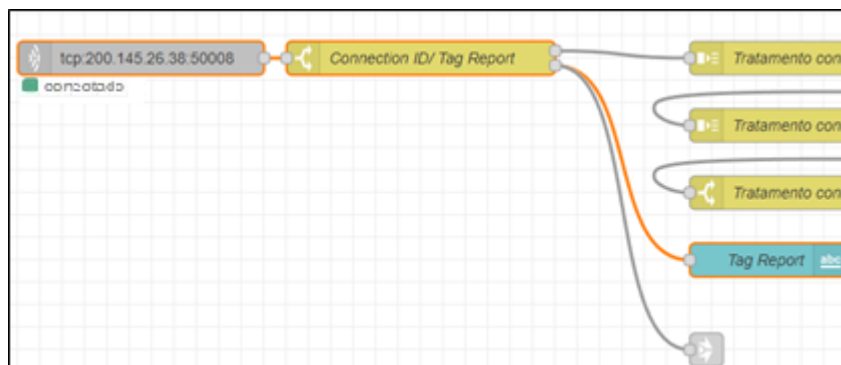
Fonte: Autoria própria.

## 4.2 Leitura e escrita pelo Dashboard do Node-RED

### 4.2.1 Leitura via Dashboard

A leitura pelo Dashboard pode ser feita de duas maneiras, de forma automática pela porta de eventos 50008 e manualmente a partir de um botão no Dashboard. A leitura automática é feita a partir do mesmo nó em que é obtido o *Connection ID*, o *tcp:200.145.26.38:50008*, destacado na Figura 17.

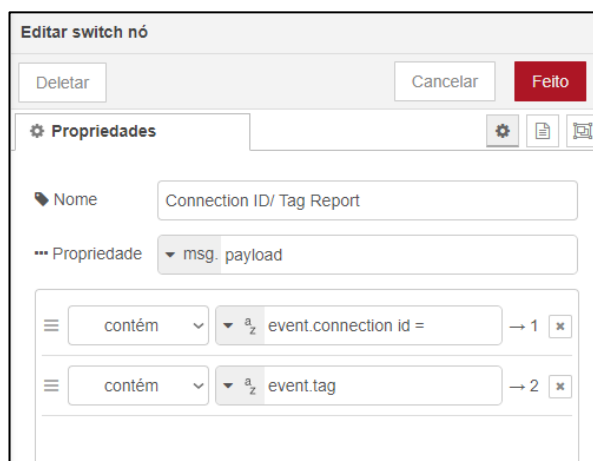
Figura 17 - Programação do Node-RED destacando o envio de leitura de *tags* para o Dashboard.



Fonte: Autoria própria.

Portanto, a segunda saída do nó *Connection ID/Tag Report* envia os dados de leitura de *tags* que entram e saem do alcance das antenas para o nó *text* denominado *Tag Report* para enviar as informações no Dashboard. A Figura 18 apresenta a configuração realizada no nó *Connection ID/Tag Report* para a divisão dos dados através de suas duas saídas.

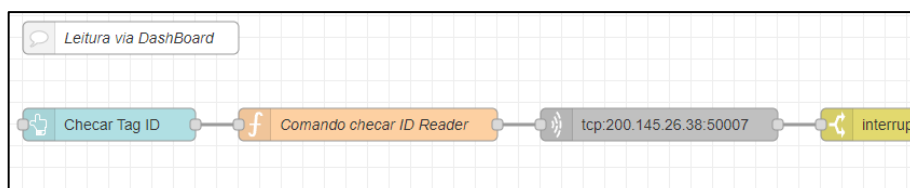
Figura 18 - Configurações do nó Connection ID/Tag Report.



Fonte: Autoria própria.

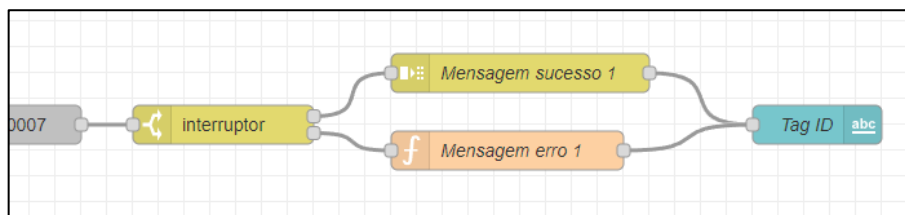
Manualmente a leitura é feita a partir de um botão no Dashboard, a Figura 19 e a Figura 20 apresentam a programação referentes a essa parte.

Figura 19 - Programação da leitura via Dashboard parte I.



Fonte: Autoria própria.

Figura 20 - Programação da leitura via Dashboard parte II.



Fonte: Autoria Própria.

O nó *button* denominado *Checar Tag ID* representa o botão no Dashboard, sendo assim, quando ele é clicado, o comando CLI `tag.read_id()` escrito dentro de um nó *function* (denominado *Comando checar ID Reader*) é enviado ao leitor pelo nó `tcp:200.145.26.38:50007`. Caso tenha alguma *tag* no alcance das antenas, o valor de *Tag ID* é mostrado na Dashboard pelo nó *text Tag ID*, caso nenhuma *tag* esteja no alcance da antena quando pressionado o botão a mensagem “Nenhuma *tag* no alcance da antena” (enviado pelo *function Mensagem de erro 1*) será mostrado pelo nó *text Tag ID*.

Figura 21 - Programação feita dentro do nó *Comando checar ID Reader*.

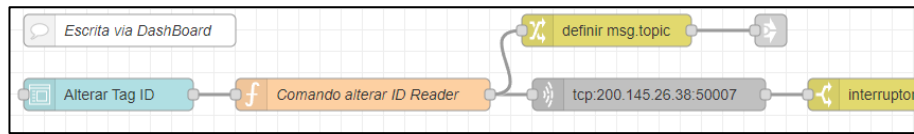
Fonte: Autoria própria.

Outros dados de leitura serão apresentados no Dashboard, como esses dados também tem relação com leitura pelo servidor, seu desenvolvimento será mais detalhado na seção 4.3.2 Leitura pelo Servidor OPC UA.

#### 4.2.2 Escrita via Dashboard

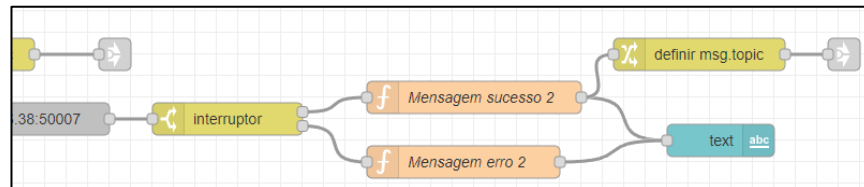
Para realizar a escrita em uma *tag* via Dashboard, foi utilizado um nó que cria um campo de formulário para o usuário escrever, a Figura 22 e Figura 23 apresentam a programação referente a esta funcionalidade.

Figura 22 - Programação da escrita via Dashboard parte I.



Fonte: Autoria própria.

Figura 23 - Programação da escrita via Dashboard parte II.



Fonte: Autoria própria.

O nó *form* denominado *Alterar Tag ID* é o formulário responsável por coletar o valor da *Tag ID* que será escrita na *tag*, a Figura 24 mostra a configuração desse nó.

Figura 24 - Configuração do nó *Alterar Tag ID*.

Label	Nome	Type	Required	UiRows	Remove
Digite a nova TagID	id	Text	<input type="checkbox"/>	1	

Fonte: Autoria própria.

Por padrão, as informações recebidas e enviadas do Node-RED ficam armazenadas no *msg.payload* das mensagens, porém, quando se usa o nó *form* a mensagem ganha uma nova propriedade subdividida dentro do *msg.payload*, neste caso, a informação do valor da nova *Tag ID*, ficará armazenada no *msg.payload.id*, pois como é mostrado na Figura 24 acima, a opção *Nome* das propriedades do *form* foi denominado como *id*.

O nó *function* denominado *Comando alterar ID Reader* possui o comando CLI utilizado para a realização da escrita da *Tag ID*, sendo enviada para o leitor através do nó *tcp:200.145.26.38:50007*. A Figura 25 apresenta o comando CLI configurado dentro deste nó *function*.

Figura 25 - Programação dentro do nó Comando alterar ID Reader.



Fonte: Autoria própria.

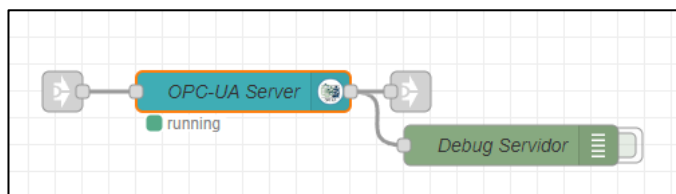
Com isso, o leitor RFID responde com duas possíveis respostas após esse comando de escrita, o êxito ou a falha na execução do comando. Quando ocorre o êxito, a mensagem “*Tag alterada com sucesso*” é enviada por um nó *function* (denominado *Mensagem Sucesso 2*) para a Dashboard por uma caixa de texto (nó *text*). Quando ocorre a falha, a mesma lógica acontece, porém no *function* a mensagem enviada ao Dashboard é “*Erro ao alterar a tag ID*”.

### 4.3 Leitura e escrita pelo servidor/cliente OPC UA

#### 4.3.1 Criação do servidor e das variáveis

Para este projeto, o servidor é criado pelo nó *OpcUa-Server* da paleta de nós *node-red-contrib-opcua*. O servidor será hospedado no mesmo computador em que está funcionando o Node-RED (endereço *localhost*). Basta arrastar o nó para o painel de fluxo que sempre que o Node-RED for iniciado, o servidor estará online.

Figura 26 - Nó responsável pela criação do servidor OPC UA em funcionamento.



Fonte: SOUZA; GODOY; CALDANA (2024).

Nas propriedades do nó é possível fazer as configurações do servidor, tal como definir a porta que ele será conectado no *localhost*, a Figura 27 demonstra essas configurações.

Figura 27 - Configurações do nó responsável pela execução do servidor OPC UA.

**Editar OpcUa-Server nó**

Deletar Cancelar Feito

**Propriedades**

Port: 53880

Name: OPC-UA Server

ResourcePath: UA/SimpleNodeRedServer

Users file: users.json

Custom nodeset directory:

Auto Accept Unknown Certificates ☒

Register to Local Discovery ☒

Build default AddressSpace ☐

Allow anonymous user ☒

Security mode: None ☒

Security mode: Sign ☐

Security mode: Sign & Encrypt ☐

Security policy: Basic128-Rsa15 ☐

Security policy: Basic256 ☐

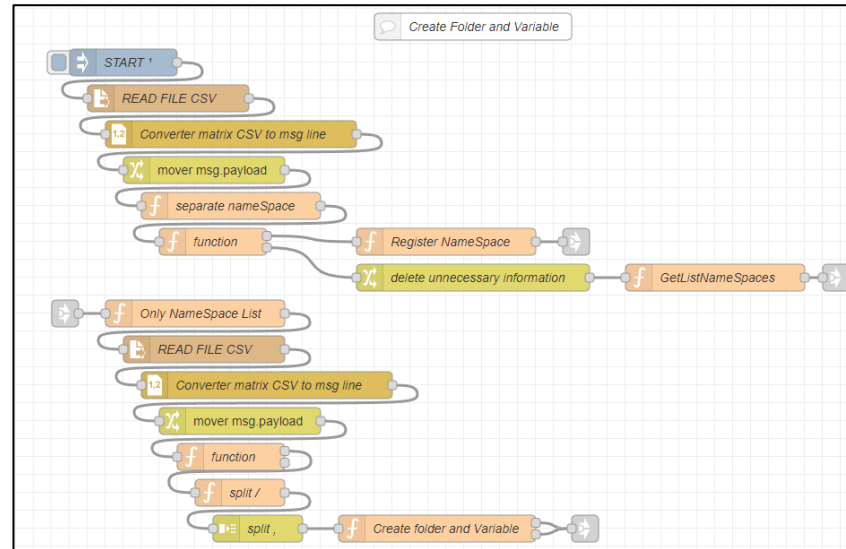
Security policy: Basic256-Sha256 ☐

Fonte: SOUZA; GODOY; CALDANA (2024).

Uma vez que este projeto visa a integração de outros projetos existentes, o servidor OPC UA utilizado entre esses projetos será o mesmo para que todas as variáveis e informações sejam organizadas em um mesmo local. Com isto em vista, o método de criação de variáveis para este projeto se aproveita do método dos projetos existentes para não ocorrer erros e conflitos de dados. Neste método, os *NameSpaces*, as pastas e as variáveis são criadas a partir de um arquivo .CSV (salva dados em estruturas de tabelas), este arquivo é lido pelo Node-RED que através de nós *functions* organiza e envia para o servidor OPC UA. A Figura 28 apresenta uma visão geral da programação desse método aproveitado para este projeto.



Figura 28 - Programação do Node-RED para a criação dos NameSpaces, pastas e variáveis.



Fonte: SOUZA; GODOY; CALDANA (2024).

Sendo assim, o *NameSpace* relacionado aos dados desse projeto no servidor foi definido como 19, a pasta denominada *RFID* e todas as variáveis criadas são apresentadas na Tabela 3. Nota-se que as variáveis foram separadas por antena do leitor RFID e todas são do tipo *String*.

Tabela 3 - Lista de variáveis criadas no servidor OPC UA.

Nome da variável	Função
Antena1_TagReport	Faz a leitura de quando uma <i>tag</i> entra ou sai do alcance da antena 1
Antena1_R_TagID	Faz a leitura da <i>Tag ID</i> de uma <i>tag</i> enquanto ela está no alcance da antena 1.
Antena1_R_UserData	Faz a leitura do <i>User Data</i> de uma <i>tag</i> enquanto ela está no alcance da antena 1.
Antena1_W_TagID	Faz a escrita da <i>Tag ID</i> em uma <i>tag</i> que está no alcance da antena 1.
Antena1_W_UserData	Faz a escrita do <i>User Data</i> em uma <i>tag</i> que está no alcance da antena 1.
Antena2_TagReport	Faz a leitura de quando uma <i>tag</i> entra ou sai do alcance da antena 2.
Antena2_R_TagID	Faz a leitura da <i>Tag ID</i> de uma <i>tag</i> enquanto ela está no alcance da antena 2.
Antena2_R_UserData	Faz a leitura do <i>User Data</i> de uma <i>tag</i> enquanto ela está no alcance da antena 2.
Antena2_W_TagID	Faz a escrita da <i>Tag ID</i> em uma <i>tag</i> que está no alcance da antena 2.
Antena2_W_UserData	Faz a escrita do <i>User Data</i> em uma <i>tag</i> que está no alcance da antena 2.

Antena3_TagReport	Faz a leitura de quando uma <i>tag</i> entra ou sai do alcance da antena 3.
Antena3_R_TagID	Faz a leitura da <i>Tag ID</i> de uma <i>tag</i> enquanto ela está no alcance da antena 3.
Antena3_R_UserData	Faz a leitura do <i>User Data</i> de uma <i>tag</i> enquanto ela está no alcance da antena 3.
Antena3_W_TagID	Faz a escrita da <i>Tag ID</i> em uma <i>tag</i> que está no alcance da antena 3.
Antena3_W_UserData	Faz a escrita do <i>User Data</i> em uma <i>tag</i> que está no alcance da antena 3.
Antena4_TagReport	Faz a leitura de quando uma <i>tag</i> entra ou sai do alcance da antena 4
Antena4_R_TagID	Faz a leitura da <i>Tag ID</i> de uma <i>tag</i> enquanto ela está no alcance da antena 4.
Antena4_R_UserData	Faz a leitura do <i>User Data</i> de uma <i>tag</i> enquanto ela está no alcance da antena 4.
Antena4_W_TagID	Faz a escrita da <i>Tag ID</i> em uma <i>tag</i> que está no alcance da antena 4.
Antena4_W_UserData	Faz a escrita do <i>User Data</i> em uma <i>tag</i> que está no alcance da antena 4.
StatusEscrita	Indica se a escrita pelo servidor ocorreu com sucesso ou não.

Fonte: Autoria própria.

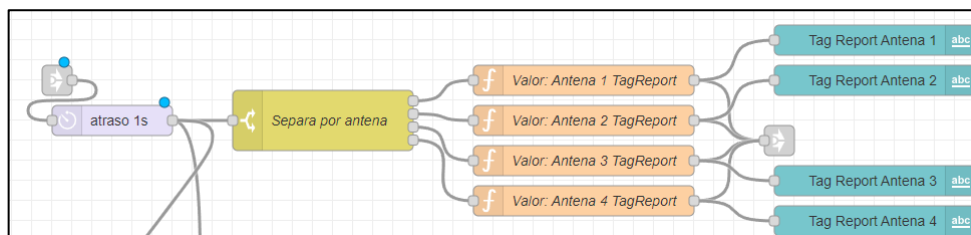
Vale ressaltar que nas variáveis relacionadas à *\_R\_TagID* e *\_R\_UserData* apresentam dados apenas quando uma *tag* estiver no alcance da respectiva antena, quando a *tag* sai do alcance, essas variáveis têm seu valor apagado, sendo então utilizadas para facilitar a identificação da posição atual da *tag* no FMS.

#### 4.3.2 Leitura pelo Servidor OPC UA.

Quando o leitor RFID envia os dados das *tags* captados pelas antenas, existe um padrão, toda vez que uma *tag* entra no alcance, a mensagem com os dados enviados do leitor RFID sempre começa com o termo *event.tag.arrive* e quando sai do alcance a mensagem sempre começa com *event.tag.depart*. É a partir dessa característica que os dados serão organizados e encaminhados pelo Node-RED.

Para as variáveis relacionadas ao *TagReport* a Figura 29 apresenta a programação no Node-RED de como esses dados são enviados ao servidor.

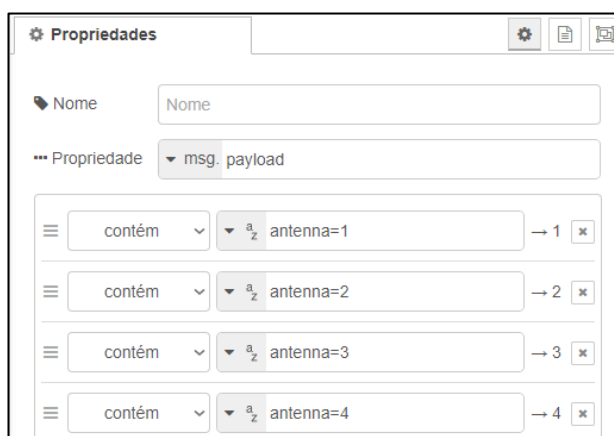
Figura 29 - Programação no Node-RED feita para enviar os dados de TagReport ao servidor OPC UA.



Fonte: Autoria própria.

As informações vêm de um nó *link in* que traz os dados captados pela porta 50008 do leitor RFID (mesmo nó *tcp:200.145.26.38:50008* da Figura 17). Primeiro as informações são passadas através de um nó de atraso de 1 segundo para garantir que não haja uma sobreposição dos dados e logo depois são separadas dependendo do número da antenna pelo nó *Separa por antenna*, a Figura 30 apresenta a configuração desse nó.

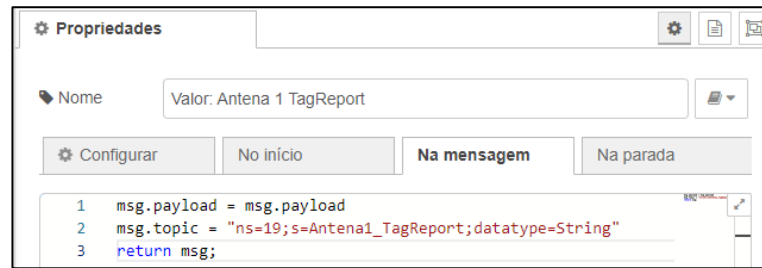
Figura 30 - Configuração do nó para a organização de dados por antenna.



Fonte: Autoria própria.

Depois disso, a informação é encaminhada para um nó *function* que grava em qual variável a determina informação deve ser armazenada no servidor. O nó do servidor OPC UA identifica qual variável deve ser modificada através do *msg.topic* da mensagem. Sendo assim, a Figura 31 apresenta a programação realizada no nó *Valor: Antena 1 TagReport*, os outros 3 nós *function* relacionados a esse valor têm a programação idêntica alterando apenas o nome da sua variável “s” correspondente. Toda essa informação também é encaminhada ao Dashboard pelos nós *texts* denominados *Tag Report Antena X*, sendo X o número da antenna correspondente.

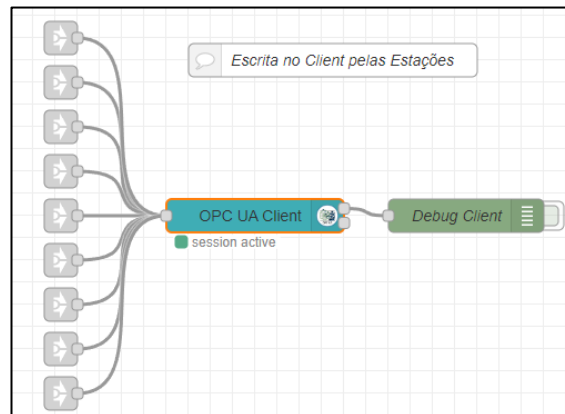
Figura 31 - Programação presente no nó function denominado Valor: Antena 1 TagReport.



Fonte: Autoria própria.

A partir disso, a informação é enviada por um nó *link out* até o nó chamado de *OPC UA Client*, nó este que é o responsável por escrever e alterar as variáveis do servidor. A Figura 32 apresenta o nó *OPC UA Client* quando está funcionando corretamente e a Figura 33 mostra as configurações desse nó, observa-se que são necessários o Endpoint e a ação (*action*) do cliente para seu funcionamento correto.

Figura 32 - Nó OPC UA Client funcionando corretamente.



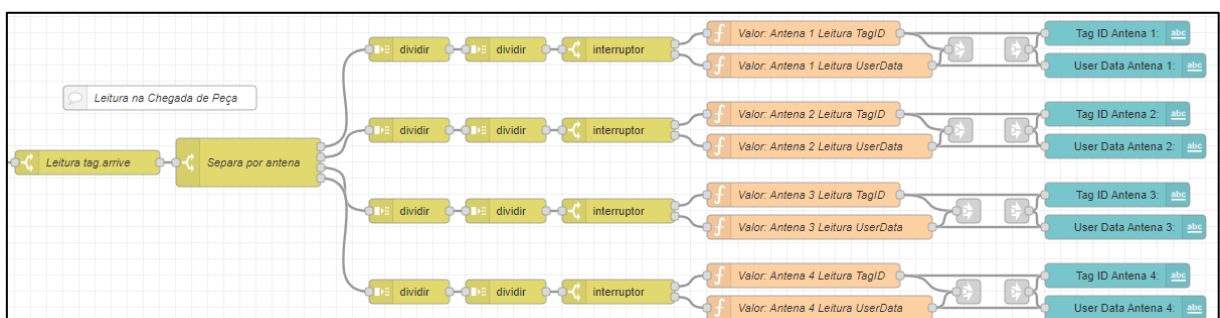
Fonte: SOUZA; GODOY; CALDANA (2024).

Figura 33 - Configurações do nó OPC UA Client.

Fonte: SOUZA; GODOY; CALDANA (2024).

Com relação as variáveis que possuem apenas as informações de *Tag ID* e *User Data* a Figura 34 apresenta a programação de como esses dados são enviados e gravados no servidor.

Figura 34 - Programação no Node-RED feita para enviar os dados de *Tag ID* e *User Data* ao servidor OPC UA.

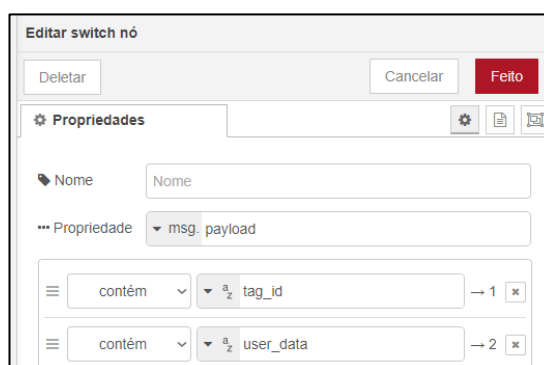


Fonte: Autoria própria.

O nó *Leitura tag arrive* filtra as informações enviadas do leitor RFID, deixando passar no fluxo somente os dados de chegada de *tags*. Novamente tem-se um nó *Separa por antena*, separando os dados em 4 caminhos com base na antena que captou a chegada. Em cada caminho possui dois nós *dividir* e um nó *interruptor*. Os dois nós *dividir* fragmentam a mensagem e eliminam os espaços e as vírgulas, com isso, o nó *interruptor* envia a *Tag ID* para um nó *function* e o *User Data* para um outro *function*. Nesses dois nós *function* a *msg.topic* é devidamente

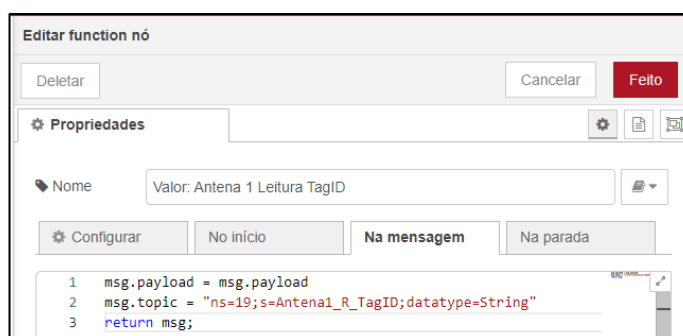
configurada para que os dados sejam enviados ao servidor corretamente. Os valores do *Tag ID* e *User Data* também são encaminhados ao Dashboard através de uma série de nós *texts*. A Figura 35 apresenta a configuração do nó *interruptor*, a Figura 36 a configuração do nó *Valor: Antena 1 Leitura Tag ID* e a Figura 37 do nó *Valor: Antena 1 Leitura UserData*, para os demais nós *functions*, as configurações foram as mesmas diferenciando apenas o valor da variável “s” do *msg.topic*. Com isso, os dados são enviados para o nó *OPC UA Client* e as informações recebidas são gravadas nas respectivas variáveis do servidor.

Figura 35 - Configuração do nó *interruptor* que separa os dados de *Tag ID* e *User Data*.



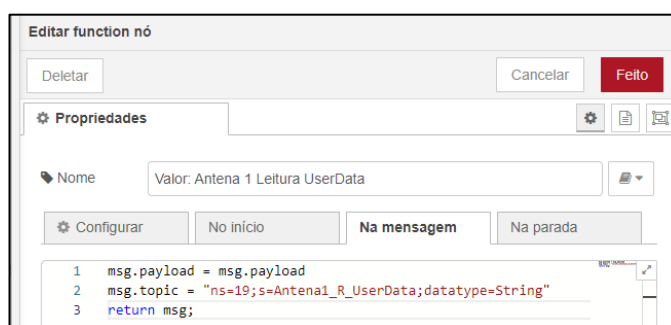
Fonte: Autoria própria.

Figura 36 - Programação do nó *Valor: Antena 1 Leitura TagID*.



Fonte: Autoria própria.

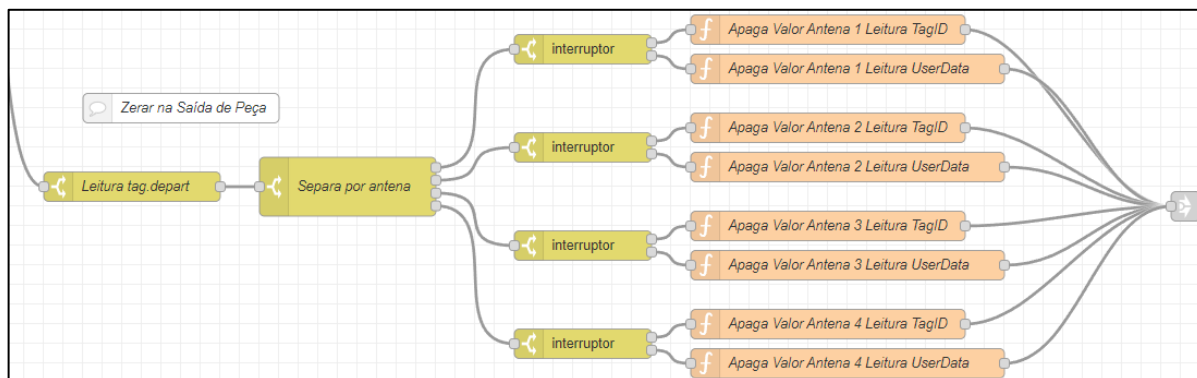
Figura 37 - Programação do nó *Valor: Antena 1 Leitura UserData*.



Fonte: Autoria própria.

Como os dados de *Tag ID* e *User Data* no servidor possuirão valores somente enquanto a *tag* estiver no alcance da antena, quando sair do alcance os valores serão apagados, a Figura 38 apresenta a programação feita para realizar esse procedimento, apagando os dados tanto no servidor quanto no Dashboard.

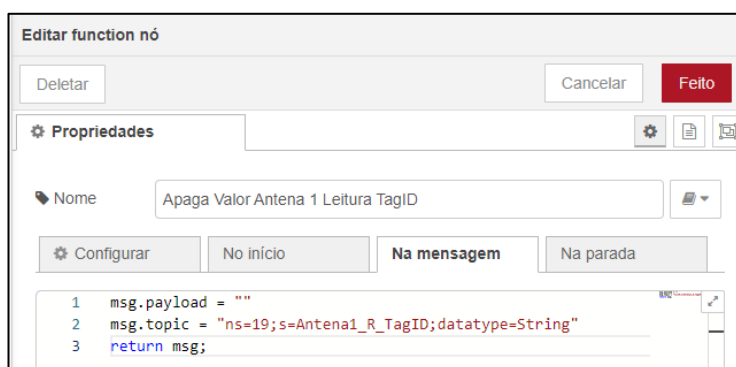
Figura 38 - Programação feita para apagar os dados de TagID e UserData na saída da tag.



Fonte: Autoria própria.

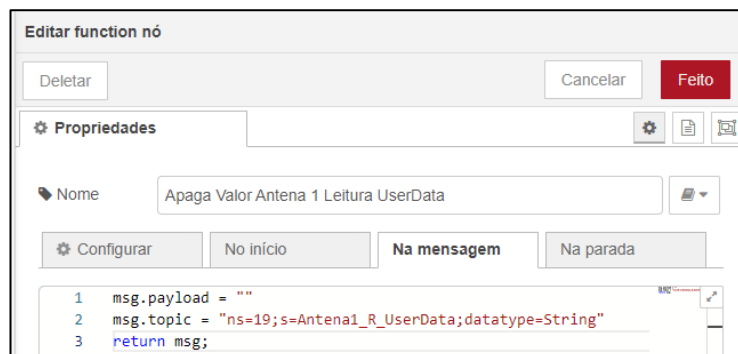
O nó *Leitura tag.depart* filtra as informações, deixando passar apenas os dados relacionados à saída de *tags*. O nó denominado *Separa por antena* organiza os dados com base na antena que captou a saída. E o nó *interruptor* encaminha a informação para os nós *function* que enviam uma mensagem em branco para a respectiva variável através do nó *OPC UA Client*. A Figura 39 apresenta a configuração do nó *Apaga Valor Antena 1 Leitura TagID* e a Figura 40 a configuração do nó *Apaga Valor Antena 1 Leitura UserData*. Para os demais nós *functions* a configuração é a mesma alterando apenas a variável do servidor que o valor será gravado, ou seja, seu respectivo valor “s” no *msg.topic*.

Figura 39 - Programação do nó *Apaga Valor Antena 1 Leitura TagID*.



Fonte: Autoria própria.

Figura 40 - Programação do nó *Apaga Valor Antena 1 Leitura UserData*.



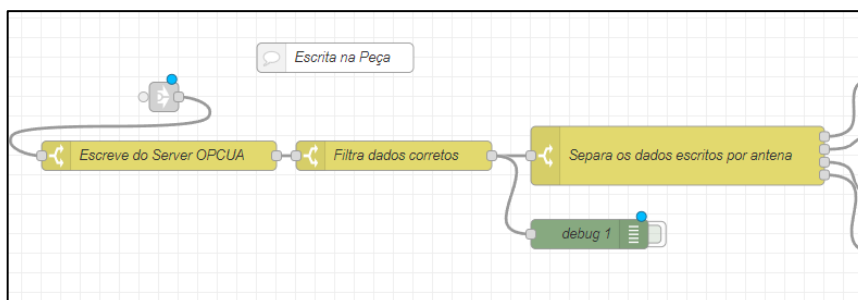
Fonte: Autoria própria.

#### 4.3.3 Escrita pelo Servidor OPC UA.

Quando algo é escrito no servidor, os dados chegam ao Node-RED pela saída do nó *OPC UA Server*, em que, os nomes das variáveis do servidor ficam gravadas na propriedade *msg.payload.variableName* e o valor das variáveis ficam gravadas na propriedade *msg.payload.variableValue* da mensagem.

Pelo servidor OPC UA será possível fazer a escrita da *Tag ID* e do *UserData* de uma peça que possua uma *tag*. A Figura 41 apresenta a programação inicial feita para estas funcionalidades.

Figura 41 - Programação realizada no Node-RED para a escrita via servidor OPC UA parte I.



Fonte: Autoria própria.

O nó *link in* encaminha as informações que saem do nó *OPC UA server* direto para o nó denominado *Escreve do Server OPCUA*. Sua função é filtrar os dados referentes à escrita das *tags*, fazendo isso identificado se a variável possui em seu nome o termo “\_W\_”, já que está presente em todas as variáveis relacionadas a escrita pelo servidor (*Antena1\_W\_TagID*, *Antena2\_W\_UserData*, entre outros). O nó denominado *Filtra dados corretos* impede a passagem de variáveis de escrita em branco. E o nó denominado *Separa os dados escritos por antena* faz o que seu nome sugere, verificando pelo *msg.payload.variableName* a antena em



que a escrita está sendo realizada. A Figura 42 apresenta a configuração do nó *Escreve do Server OPCUA*, a Figura 43 do nó *Filtra dados corretos* e a Figura 44 do nó *Separa os dados escritos por antena*.

Figura 42 - Configurações do nó *Escreve do Server OPCUA*.

The screenshot shows the 'Editar switch nó' (Edit node switch) dialog box. At the top, there are buttons for 'Deletar', 'Cancelar', and 'Feito'. Below is the 'Propriedades' (Properties) section. The 'Nome' (Name) field is set to 'Escreve do Server OPCUA'. The 'Propriedade' (Property) dropdown is set to 'msg. payload.variableName'. At the bottom, there is a filter rule: 'contém' (contains) followed by a dropdown menu showing 'a\_z' and '\_W\_', and a count of '1'.

Fonte: Autoria própria.

Figura 43 - Configuração do nó *Filtra dados corretos*.

The screenshot shows the 'Editar switch nó' dialog box. The 'Nome' field is set to 'Filtra dados corretos'. The 'Propriedade' dropdown is set to 'msg. payload.variableValue'. At the bottom, the filter rule is 'não é nulo' (is not null) with a count of '1'.

Fonte: Autoria própria.

Figura 44 - Configuração do nó *Separa os dados escritos por antena*.

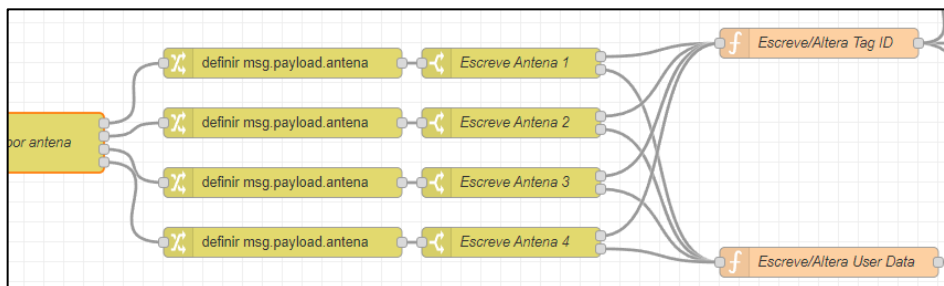
The screenshot shows the 'Editar switch nó' dialog box. The 'Nome' field is set to 'Separa os dados escritos por antena'. The 'Propriedade' dropdown is set to 'msg. payload.variableName'. Below this, there are four filter rules, each with a count:
 

- 'contém' followed by 'Antena1' and count '1'
- 'contém' followed by 'Antena2' and count '2'
- 'contém' followed by 'Antena3' and count '3'
- 'contém' followed by 'Antena4' and count '4'

Fonte: Autoria própria.

A Figura 45 apresenta a segunda parte da programação realizada para a escrita nas *tags* pelo servidor.

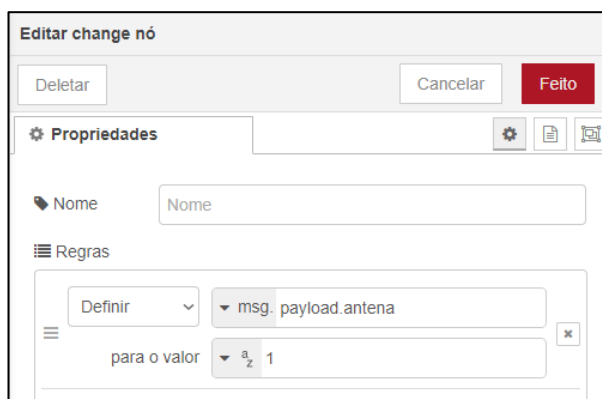
Figura 45 - Programação realizada no Node-RED para a escrita via servidor OPC UA parte II.



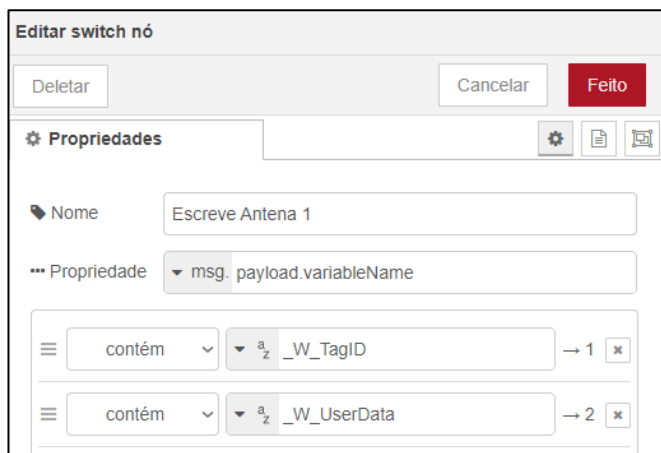
Fonte: Autoria própria

Os nós *changes* denominados *definir msg.payload.antena*, criam a propriedade *msg.payload.antena* nas mensagens e escrevem o valor da respectiva antenna nessa nova propriedade e os nós *switches* denominados *Escreve Antena X* (em que X é o número da antenna) identificam qual propriedade da *tag* está sendo escrita pelo servidor (*Tag ID* ou *User Data*). Portanto, utilizando o *msg.payload.variableValue* (contém o valor escrito pelo servidor do *Tag ID/User Data*) e o *msg.payload.antena*, os nós *functions* enviam o comando CLI de escrita para o leitor RFID. A Figura 46 apresenta a configuração de um dos nós *definir msg.payload.antena*, a Figura 47 do nó *Escreve Antena 1*, a Figura 48 a programação do nó *Escreve/Altera Tag ID* e a Figura 49 do nó *Escreve/Altera User Data*.

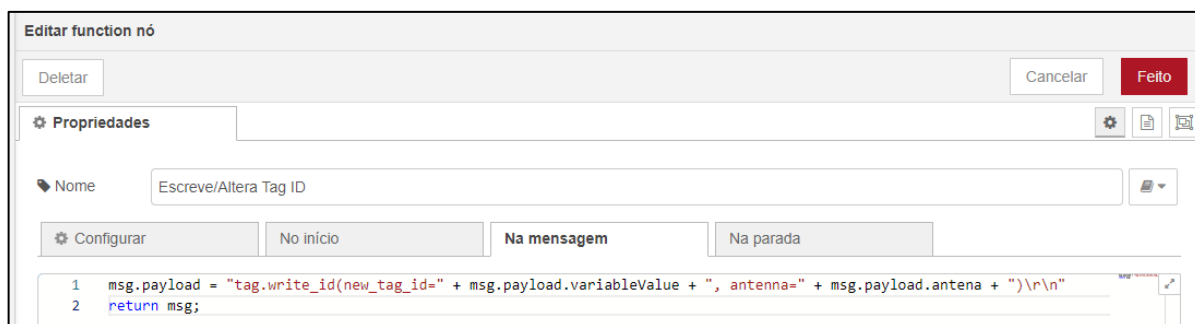
Figura 46 - Configuração do nó *definir msg.payload.antena* relacionado a escrita pela antenna 1.



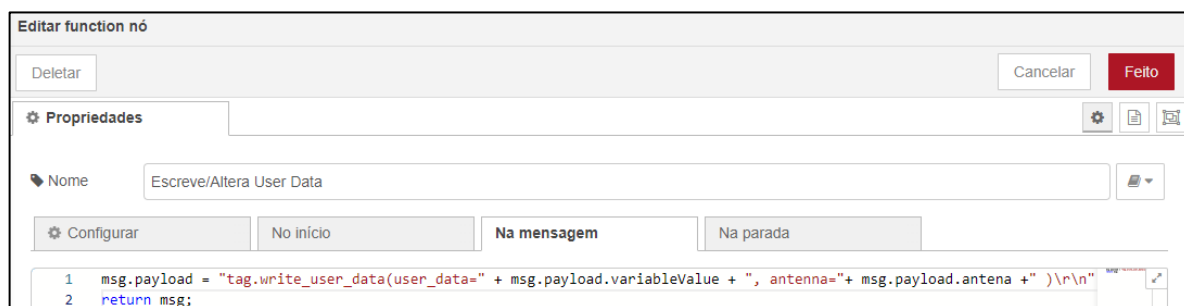
Fonte: Autoria própria

Figura 47 - Configuração do nó *Escreve Antena 1*.

Fonte: Autoria própria

Figura 48 - Programação do nó *Escreve/Altera Tag ID*.

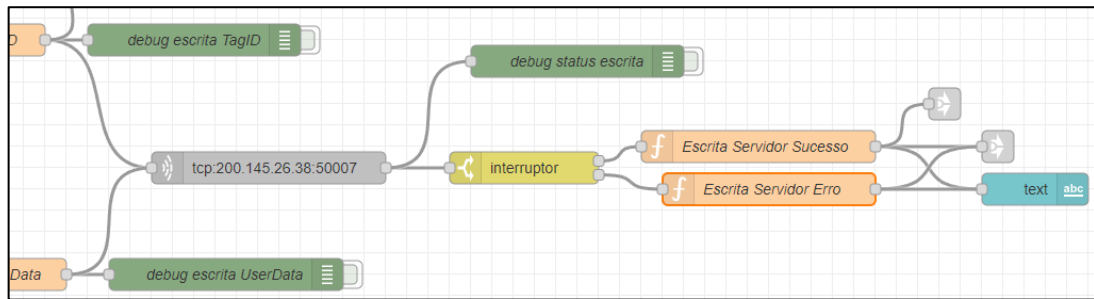
Fonte: Autoria própria

Figura 49 - Programação do nó *Escreve/Altera User Data*.

Fonte: Autoria própria

Por fim, a Figura 50 apresenta a terceira parte da programação relacionada à escrita nas *tags* via servidor OPC UA.

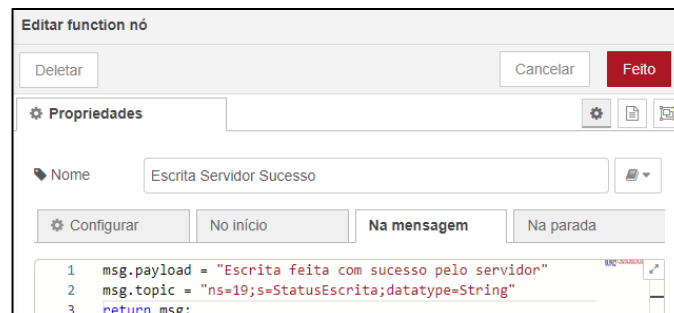
Figura 50 - Programação realizada no Node-RED para a escrita via servidor OPC UA parte III.



Fonte: Autoria própria

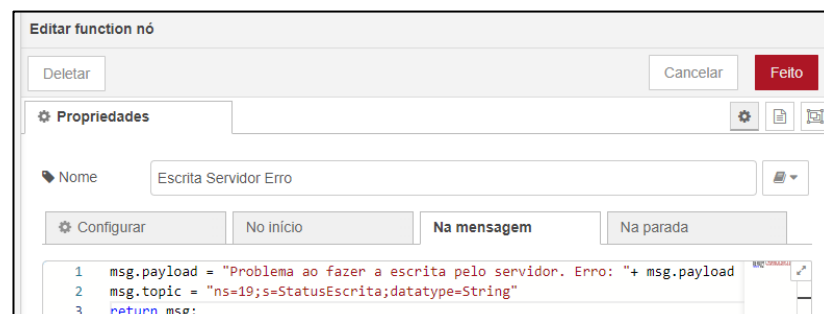
Após o envio do comando, a resposta de êxito ou erro é filtrada pelo nó *interruptor*, quando ocorre o êxito, o nó *function* denominado *Escrita Servidor Sucesso*, envia uma mensagem para a variável *StatusEscrita* do servidor e para o Dashboard (em um nó *text*) que a escrita foi realizada com sucesso. Quando ocorre o erro, é a função *Escrita Servidor Erro* que envia as mensagens indicando o erro na operação da escrita, junto com o código de erro fornecido pelo leitor RFID. A Figura 51 apresenta a programação do nó *Escrita Servidor Sucesso* e a Figura 52 do nó *Escrita Servidor Erro*.

Figura 51 - Programação do nó *Escrita Servidor Sucesso*.



Fonte: Autoria própria

Figura 52 – Programação do nó *Escrita Servidor Erro*.

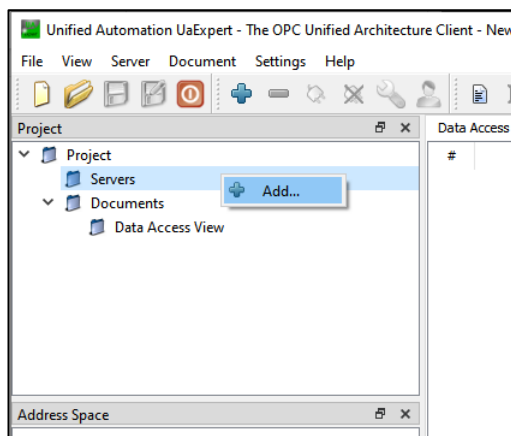


Fonte: Autoria própria

#### 4.3.4 Configuração UA Expert

Na tela inicial do UA Expert na janela *Project* foi adicionado o servidor que deve estar funcionando no Node-RED, ao clicar na opção *Server*, como mostra a Figura 53.

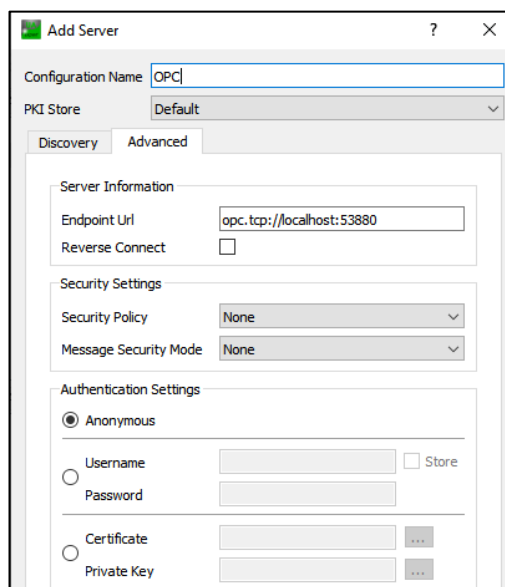
Figura 53 - Adicionando o servidor criado no Node-RED ao UA Expert



Fonte: Autoria própria

Com isso a janela *Add Server* irá abrir, Figura 54. Na aba *Advanced*, *Configuration Name* foi definido como *OPC*. O *Endpoint Url* sempre começa com *opc.tcp://*, o *localhost* é o endereço do servidor, seguido da porta 53880 que foi previamente definida no nó de criação do Servidor OPC UA no Node-RED. As definições de *Security Setting* e *Authentication Settings* também seguem as configurações do nó.

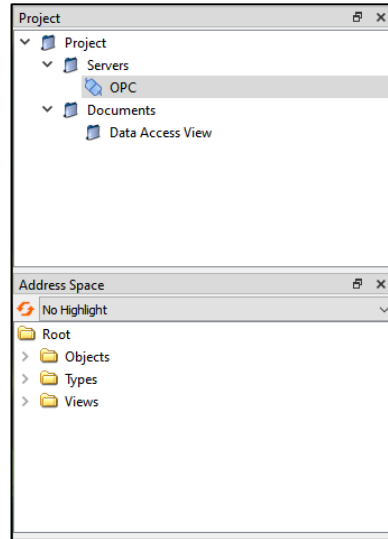
Figura 54 - Informando as configurações do servidor criado no Node-RED ao UA Expert.



Fonte: Autoria própria

Caso a conexão com o servidor seja bem-sucedida, as pastas e caminhos do servidor aparecerão na janela *Address Space*, como é apresentado na Figura 55.

Figura 55 - UA Expert conectado ao servidor OPC UA criado no Node-RED.

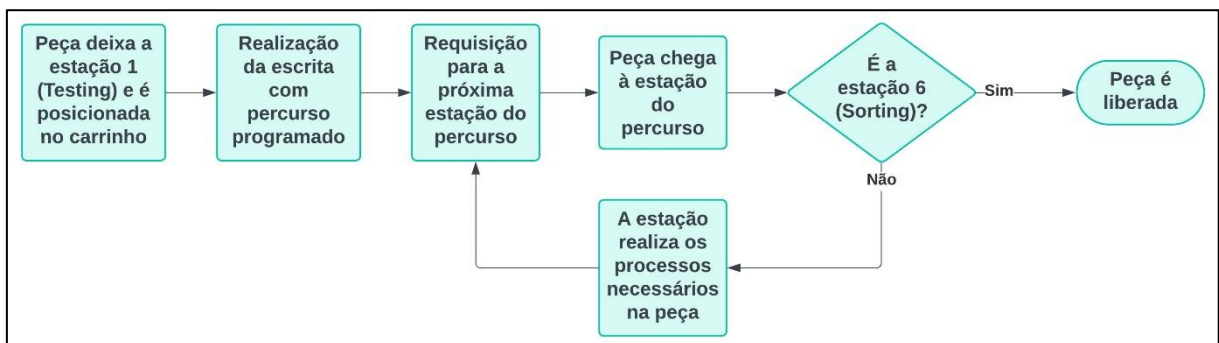


Fonte: Autoria própria

#### 4.4 Configuração do percurso da peça no FMS com base escrita.

A Figura 56 apresenta um fluxograma que idealiza o funcionamento lógico da programação do sistema com relação ao percurso da peça no FMS conforme a escrita no valor do *Tag ID*.

Figura 56 - Fluxograma do funcionamento do percurso da peça conforme escrita.



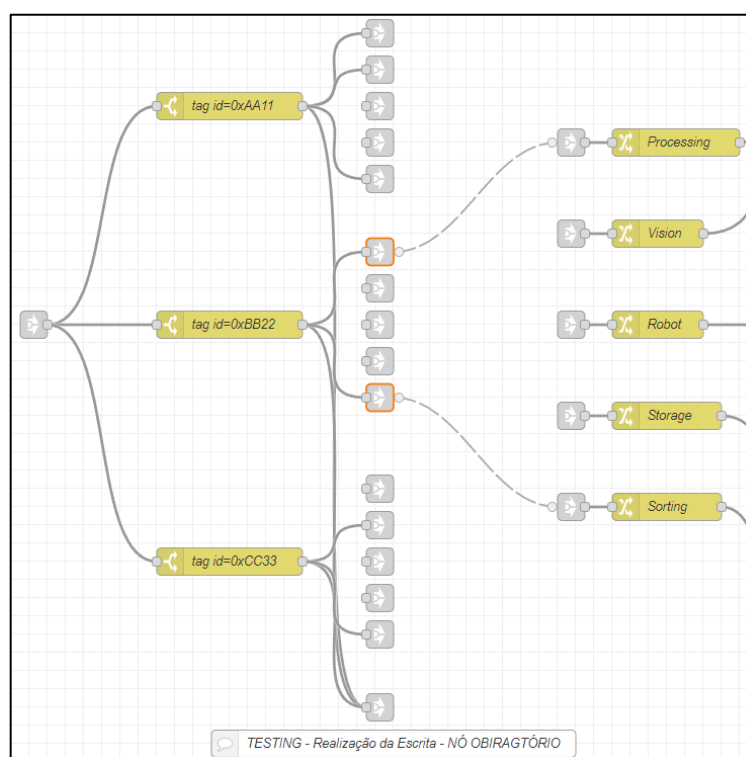
Fonte: Autoria própria

##### 4.4.1 Confirmação da realização da escrita.

Com base na escrita de *Tags IDs* específicas, a peça fará paradas em determinadas estações no FMS. A programação relacionada a essa funcionalidade visou facilitar a forma da definição dessas *Tags IDs* específicas e facilitar a definição em qual estação essas *tags* farão uma parada. A Figura 57 apresenta a programação relacionada a essas definições. Um nó *link*

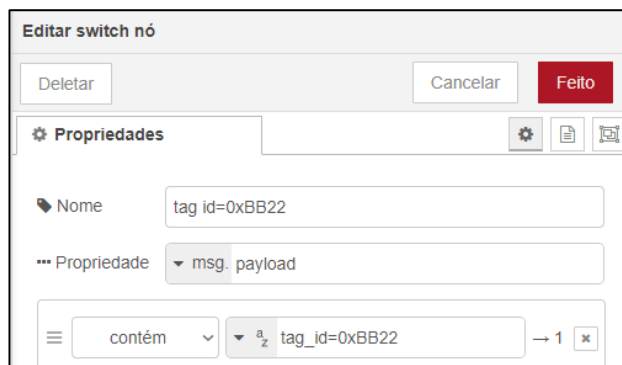
*in* inicia o fluxo recebendo informações de três fontes: dos eventos de entrada e saídas de *tags*; das escritas de *Tag ID* realizadas na Dashboard; e das escritas de *Tag ID* realizadas pelo servidor OPC UA. As informações são filtradas por nós *switches*, fazendo com que o fluxo de dados apenas continue caso a escrita realizada tenha um dos valores que foram previamente definidos por estes nós (neste caso 0xAA11, 0xBB22 e 0xCC33). A escrita é realizada pela antena 1, logo depois quando a peça deixa a estação *Testing* (Estação 1). Depois da filtragem a programação requisita a parada em uma estação dependendo das ligações de fluxo realizadas: *Processing* (Estação 2), *Vision* (Estação 3), *Robot Arm* (Estação 4), *Storage* (Estação 5) e *Sorting* (Estação 6). No caso da Figura 57 destaca-se as paradas que serão realizadas pela *Tag ID* 0xBB22. Todos os nós *switches* da definição da *Tag ID* deverão ter um fluxo conectado com o nó indicado em “*TESTING – Realização da Escrtia – NÓ OBRIGATÓRIO*”. O desenvolvimento realizado da programação para a requisição de parada em cada estação (do *Processing* ao *Sorting*) será detalhado mais adiante nas próximas seções.

Figura 57 - Programação da definição das *Tags IDs* e e suas paradas.



Fonte: Autoria própria

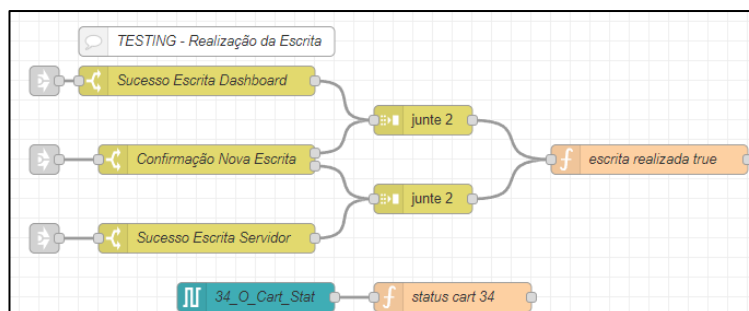
A Figura 58 apresenta a configuração de um dos nós *switches* que realiza a filtragem dos dados, os demais nós *switches* nesta parte também seguem a mesma lógica diferenciando apenas o valor da *Tag ID* pertinente.

Figura 58 - Configuração do nó *tag id=0xBB22*.

Fonte: Autoria própria

O nó indicando “*TESTING – Realização da Escrtia – NÓ OBRIGATÓRIO*” encaminha as informações para a programação presente na Figura 59, parte esta responsável na identificação de uma nova escrita de *Tag ID* pelo Dashboard ou servidor e se essa escrita foi realizada com sucesso.

Figura 59 - Parte da programação que faz a confirmação da escrita de uma Tag ID que possui as paradas no FMS configuradas.



Fonte: Autoria própria

Neste caso, três nós *link in* trazem os dados necessários. O primeiro *link in* traz a resposta do comando de escrita via Dashboard, o segundo traz a confirmação se valores escritos no Dashboard ou servidor possuem um percurso programado para realizar no FMS (ligado ao nó obrigatório da Figura 57) e o terceiro *link in* traz a resposta do comando de escrita via servidor OPC UA. Os nós denominados *Sucesso Escrita Dashboard* e *Sucesso Escrita Servidor* filtram os dados deixando o fluxo continuar quando houver êxito nos comandos de escrita. E o nó denominado *Confirmação Nova Escrita* separa se o valor escrito (valor de *Tag ID* com percurso programado) foi realizado pela Dashboard ou pelo servidor. Os dois nós *join* denominados *junte 2* funcionam como uma porta lógica E, ou seja, o fluxo continua apenas quando o valor da escrita possui um percurso programado e a confirmação de escrita foi realizada com sucesso. O nó *function* chamado de *escrita realizada true* utiliza a função *flow.set()* do Node-RED, que permite criar uma variável com um valor que é possível ser utilizada e resgatada por diferentes



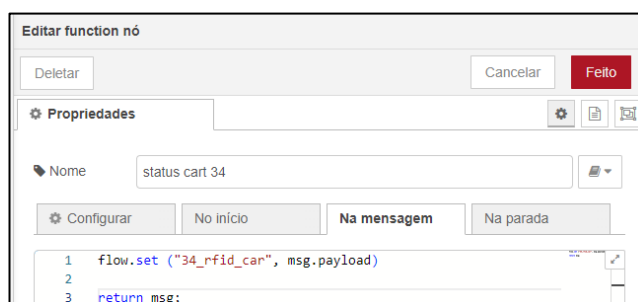
nós *functions*. Portanto, nesse nó uma variável é criada que indica quando uma operação de escrita com percurso configurado está ativa. Utilizando a mesma lógica, o nó *status cart 34* cria uma variável para sempre salvar se o carrinho do FMS está ou não parado na estação de Testing (Estação 1). A Figura 60 apresenta a programação do nó *escrita realizada true*, Figura 61 a programação do nó *status cart 34* e a Figura 62 a configuração de um nó *junte 2*.

Figura 60 - Programação do nó *escrita realizada true*.



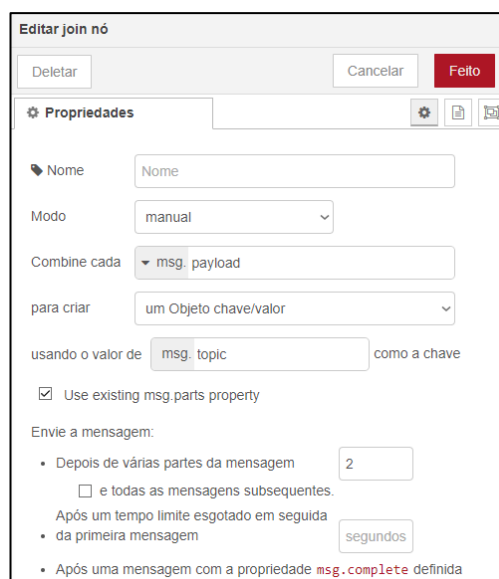
Fonte: Autoria própria

Figura 61 - Programação do nó *status cart 34*.



Fonte: Autoria própria

Figura 62 - Configuração do nó *junte 2*.



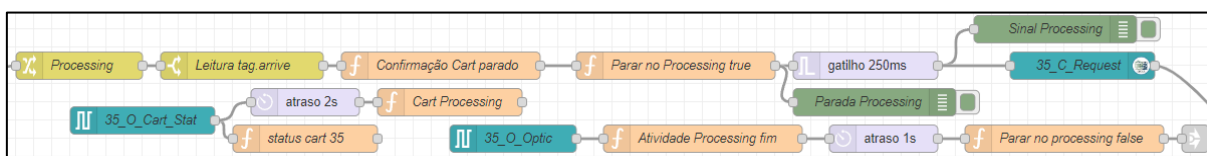
Fonte: Autoria própria

#### 4.4.2 Configuração da requisição com manipulação da peça.

Dependendo da estação em que a peça for requisitada no FMS, pode haver a manipulação dela pela estação ou não, como por exemplo na estação 3, em que a peça é apenas fotografada não acontecendo a manipulação. Com base nestas duas situações, foram feitas duas lógicas de programação para a requisição da peça em uma estação. Primeiro será apresentado o desenvolvimento do caso em que existe a manipulação da peça.

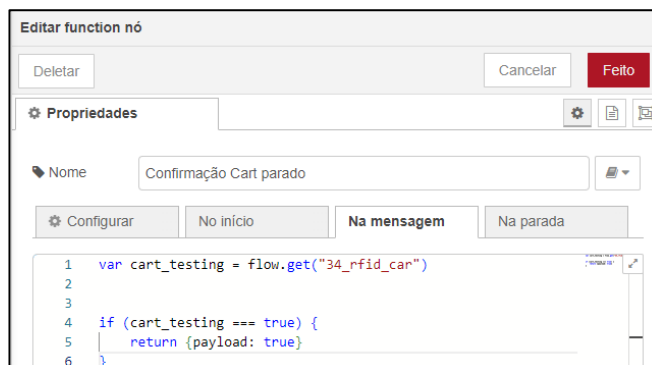
A Figura 63 apresenta a programação feita para a requisição com a manipulação da peça na estação 2 *Processing*. O nó denominado *Processing* é apenas para identificação referente a estação, não exercendo nenhuma função. O nó denominado *Leitura tag arrive* exerce a mesma função do nó de mesmo nome da Figura 34, pois no momento em que a escrita na *Tag ID* é realiza e o valor é atualizado na *tag*, o leitor identifica essas atualização do valor como uma chegada nova peça. O nó *function* denominado *Confirmação Cart parado* permite que o fluxo das informações continue apenas se o carrinho estiver parado na estação 1 onde é realizado a escrita. A função *Parar no Processing true* confirma se a ação da escrita foi realizada por uma função *if* e por meio da função *flow.set* cria uma variável (chamada de *rfid\_processing*) que será o indicativo que uma parada no *Processing* precisa ser realizada. Por fim, um o nó *gatilho 250ms* envia um sinal em pulso para a requisição do carrinho na estação por meio nó *35\_C\_Request*.

Figura 63 - Programação feita para a requisição da peça conforme a escrita com manipulação da peça.



Fonte: Autoria própria

A Figura 64 apresenta a configuração do nó *Confirmação Cart Parado* e a Figura 65 a configuração do nó *Parar no Processing true*.

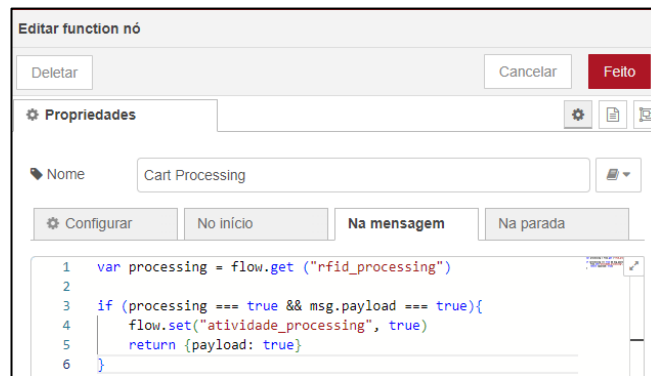
Figura 64 - Programação do nó *Confirmação Cart parado*.

Fonte: Autoria própria

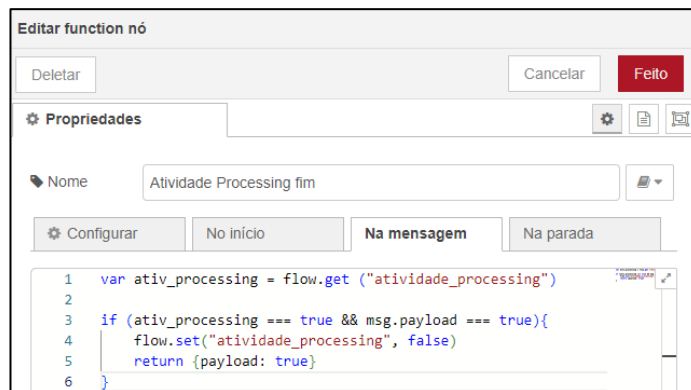
Figura 65 - Programação do nó *Parar no Processing true*.

Fonte: Autoria própria

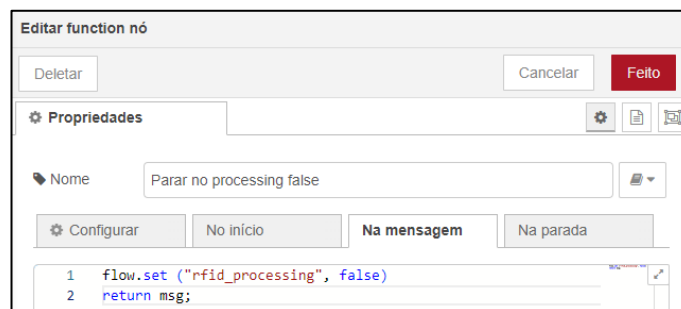
Ainda na Figura 63 os fluxos conectados aos nós *35\_O\_Cart\_Stat* e *35\_O\_Optic* são relacionados a ações quando a peça chega na estação 2 *Processing*. A função *status cart 35*, assim como a função *status cart 36* Figura 61, salva em uma variável *flow.get()* indicando se o carrinho está ou não parado (valores verdadeiro ou falso respectivamente) na estação 2. O nó denominado *Cart Processing* verifica se o carrinho está parado na estação 2 e se o indicativo de parada da variável *rfid\_processing* é verdadeiro, caso essas duas condições se satisfaçam, a função configura uma variável por meio da *flow.set()* denominada *atividade\_processing* que indica ao sistema que a estação 2 está manipulando a peça. A função *Atividade Processing fim* tem a funcionalidade de indicar que a peça foi devolvida ao carrinho pela estação 2, como o nó *35\_O\_Optic* é referente ao sensor óptico que indica quando a peça está no carrinho, a devolução é identificada quando o sensor óptico envia um sinal verdadeiro e a variável *atividade\_processing* também possui um valor verdadeiro. Por fim, a função *Parar no processing false* configura um valor falso à variável *rfid\_processing* que libera o carrinho para a próxima estação requisitada. A Figura 66 mostra a configuração do nó *Cart Processing*, a Figura 67 do nó *Atividade Processing fim* e a Figura 68 do nó *Parar no processing false*.

Figura 66 - Programação do nó *Cart Processing*

Fonte: Autoria própria

Figura 67 - Programação do nó *Atividade Processing fim*.

Fonte: Autoria própria

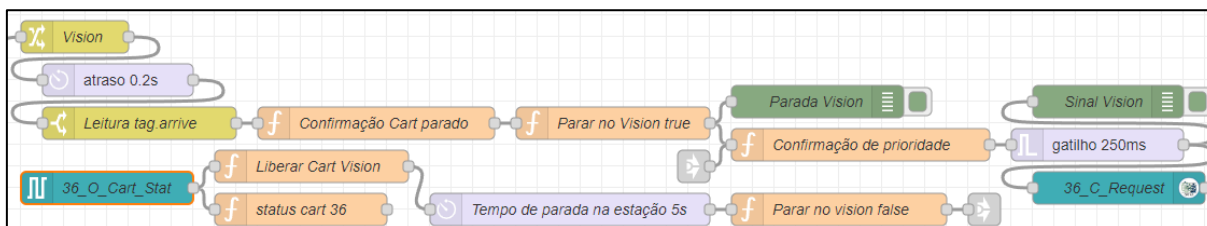
Figura 68 - Programação do nó *Parar no processing false*.

Fonte: Autoria própria

#### 4.4.3 Configuração da requisição com parada por tempo

A Figura 69 apresenta a programação feita para a requisição do carrinho para a estação 3 Vision, como se pode observar, a configuração funciona de forma parecida apresentada anteriormente na Figura 63, porém com a mudança de nome de variáveis em algumas funções e a adição de um nó *function* chamado de *Confirmação de prioridade*.

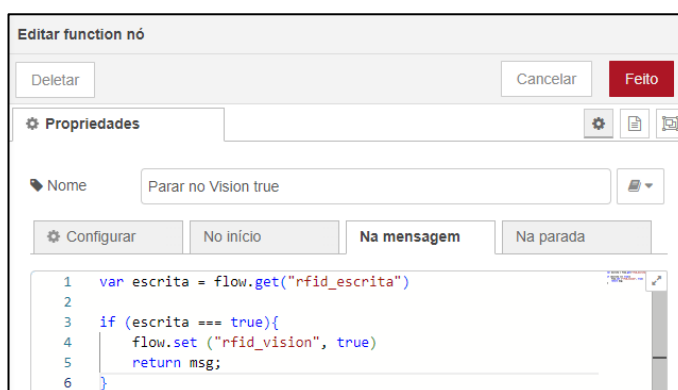
Figura 69 – Programação da requisição da peça conforme escrita com parada por tempo na estação.



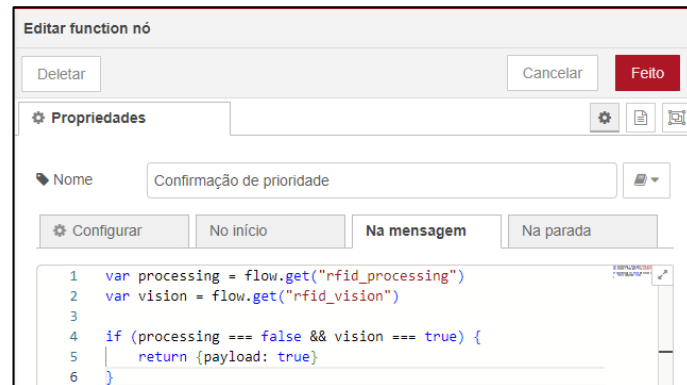
Fonte: Autoria própria

O nó *Confirmação Cart parado* continua idêntico comparado a configuração na estação 2 do *Processing*. O nó *Parar no Vision true* possui a mesma funcionalidade do *Parar no Processing true*, porém nesta situação indica ao sistema que uma parada na estação 3 é necessária por meio da variável *rfid\_vision* criada pela função *flow.set()*. O nó *Confirmação de prioridade* tem a função de identificar a ordem de prioridade para a requisição do carrinho entre as estações, como a estação 2 é a primeira estação depois da realização da escrita na *Tag ID* ela sempre vai ter prioridade sobre as outras, por isso essa função não é necessária na sua programação de requisição. Este nó sempre verifica a prioridade quando ocorre a ação da escrita e quando o carrinho é liberado por uma estação (através do nó *link in* ligado a entrada nó *Confirmação de prioridade*). Com isso, apenas depois da confirmação da prioridade o sinal é enviado ao nó *36\_C\_Request* para a requisição do carrinho. A Figura 70 apresenta a programação do nó *Parar no Vision true* e a Figura 71 a programação do nó *Confirmação de prioridade*.

Figura 70 - Programação do nó *Parar no Vision true*.



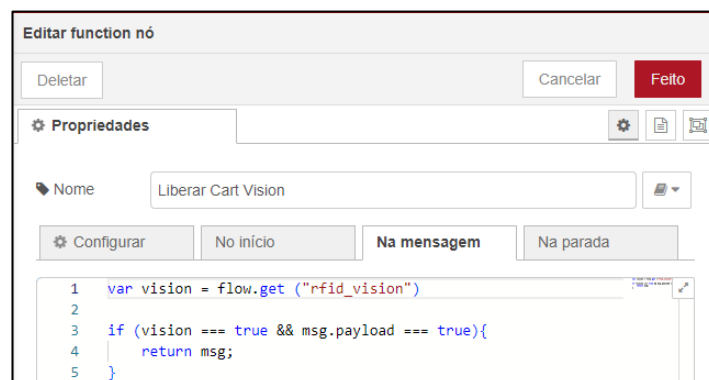
Fonte: Autoria própria

Figura 71 - Programação do nó *Confirmação de propriedade*

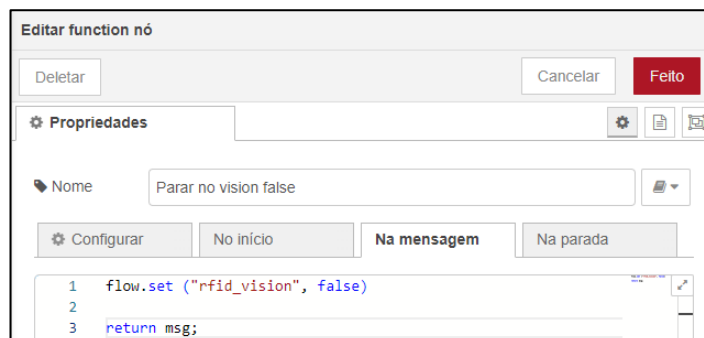
Fonte: Autoria própria

Vale ressaltar que a verificação da prioridade muda dependendo da estação, como foi visto, a estação 3 (*Vision*) precisa dar a prioridade a estação 2, já a estação 4 (*Robot*) precisa dar a prioridade as estações 2 e 3 e assim sucessivamente até a estação 6.

Retornando a Figura 69, o nó *function* denominado *Liberar Cart Vision* tem a função de continuar o fluxo da programação somente quando o carrinho estiver parado na estação do *Vision* e quando a variável de indicação de parada na estação *rfid\_vision* possuir um valor verdadeiro. Sendo assim, satisfazendo essas duas condições, um nó *delay* de 5 segundos responsável pelo tempo de parada na estação é acionado. Depois do seu tempo esgotar, a variável *rfid\_vision* recebe um valor falso pelo nó *Parar no vision false* e libera o carrinho para a próxima estação em que foi requisitado. A Figura 72 apresenta a configuração do nó *Liberar Cart Vvision*.

Figura 72 - Programação do nó *Liberar Cart Vvision*.

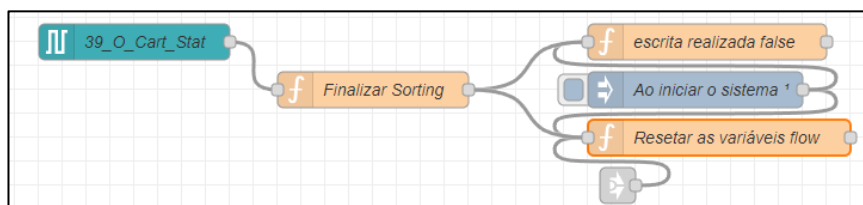
Fonte: Autoria própria

Figura 73 - Programação do nó *Parar no vision false*.

Fonte: Autoria própria

Para a requisição de parada para as demais estações, as lógicas de programação foram replicadas, obviamente levando em consideração se na estação existe a manipulação da peça ou a parada será realizada por tempo.

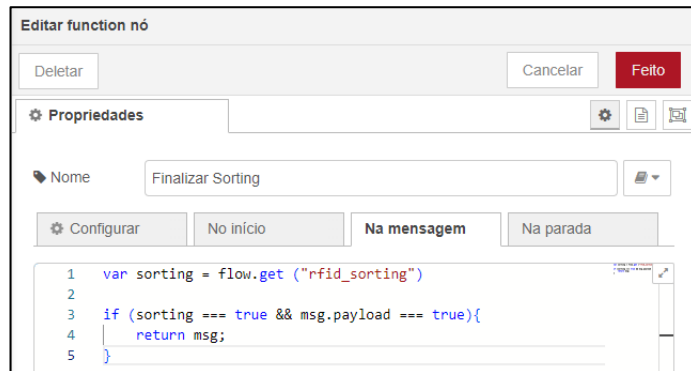
Como as variáveis criadas e manipuladas pelas funções *flow.set()* e *flow.get()* mantêm os valores salvos fora dos nós é necessário reiniciar seus valores em certos momentos para o funcionamento correto da programação, a Figura 74 apresenta a lógica para esta ação.

Figura 74 - Reinicialização dos valores das variáveis *flow*.

Fonte: Autoria própria

O nó *39\_O\_Cart\_Stat* envia um sinal verdadeiro quando o carrinho está na última estação do FMS, a estação 6 (*Sorting*). Portanto a variável relacionada a ação de escrita é reiniciada em duas situações, quando o carrinho chega a última estação e quando o sistema do Node-RED é iniciado. As demais variáveis são reiniciadas em três momentos: carrinho na última estação; quando o sistema é iniciado; e quando uma nova escrita é realizada pela Dashboard/servidor (informação esta que chega pelo nó *link in*). A Figura 75 apresenta a programação do nó *Finalizar Sorting*, a Figura 76 do nó *escrita realizada false* e a Figura 77 do nó *Resetar as variáveis flow*.

Figura 75 - Programação do nó *Finalizar Sorting*.



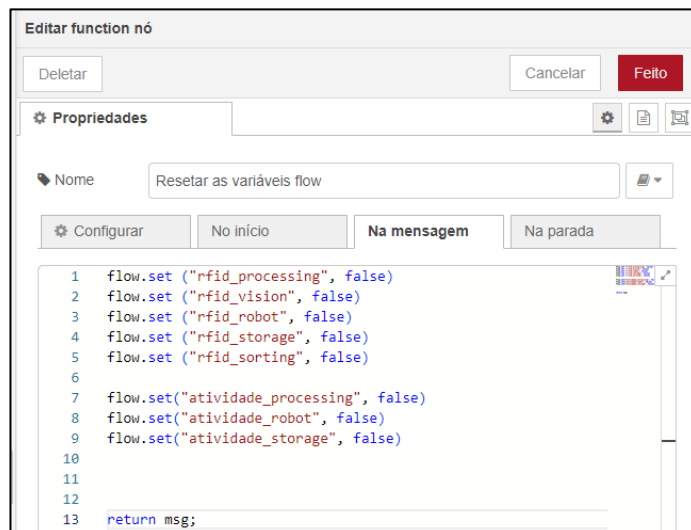
Fonte: Autoria própria

Figura 76 - Programação do nó *escrita realizada false*.



Fonte: Autoria própria

Figura 77 - Programação do nó *Resetar as variáveis flow*.



Fonte: Autoria própria

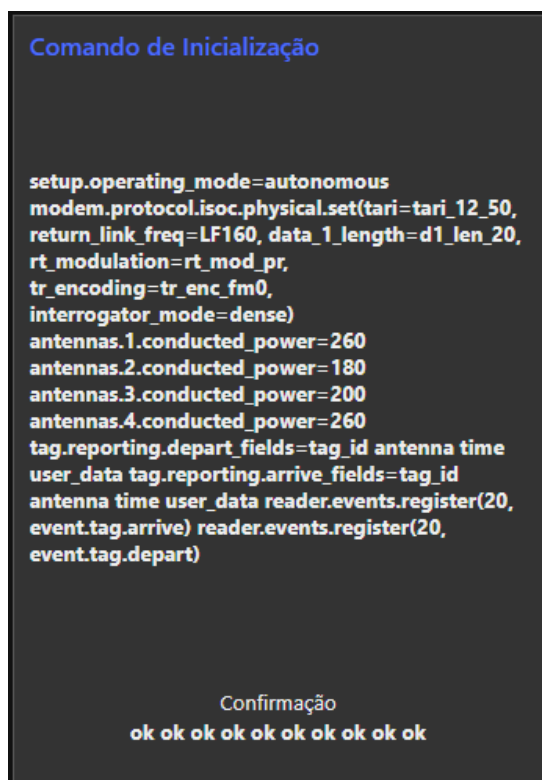


## 5 RESULTADOS

### 5.1 Inicialização do leitor RFID

A Figura 78 apresenta a tela do Dashboard referente as configurações de inicialização de leitor RFID feitas automaticamente pela programação no Node-RED. No centro apresenta a lista de comandos CLIs enviados e a parte inferior a resposta do leitor para cada comando. Quando ocorre êxito na execução o leitor responde com um *ok* para cada comando, caso ocorra a falha o leitor responde com uma mensagem de erro para cada comando em que houve o problema.

Figura 78 - Dashboard com os comandos de inicialização do leitor RFID e suas confirmações.



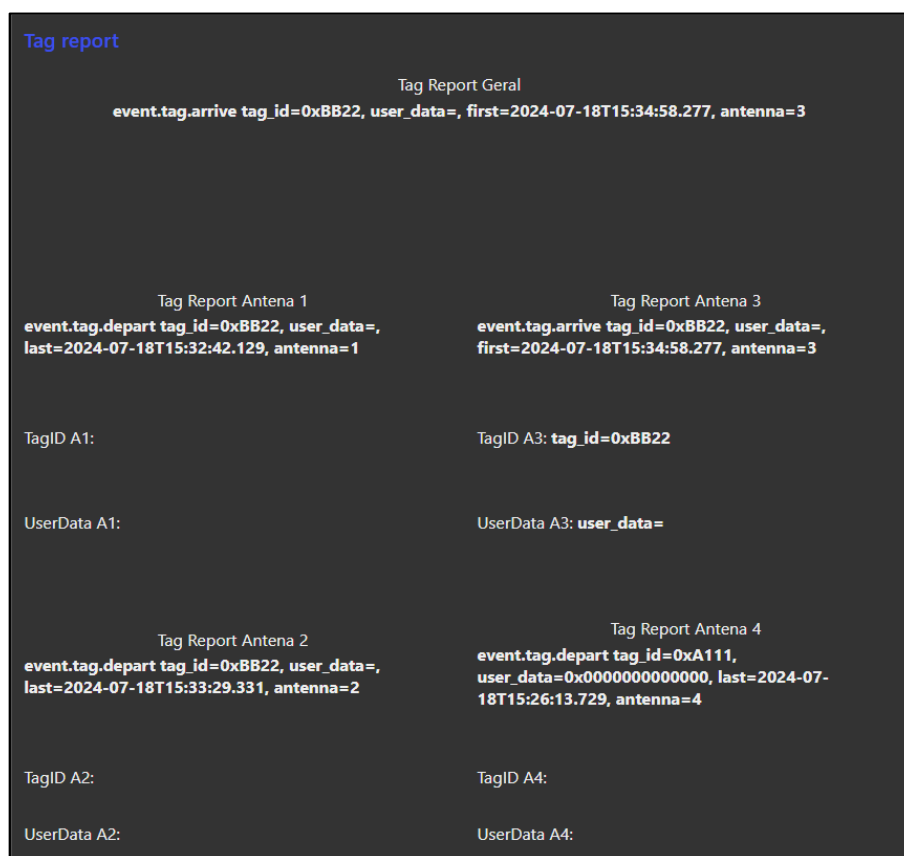
Fonte: Autoria própria

Em todos os testes e execuções, em nenhum momento ocorreu um erro referente à execução dos comandos CLIs para a inicialização do leitor RFID. Os testes incluíram interromper e reconectar a comunicação entre o leitor e o Node-RED, como desligar e ligar o leitor e desconectar e reconectar o cabo Ethernet do leitor.

## 5.2 Leitura e Escrita Dashboard

A Figura 79 apresenta a tela do Dashboard com a janela referente a leitura de *tags*. No quadro de Tag report, tem-se Tag Report Geral que apresenta o último evento de entrada ou saídas de uma *tag* englobando todas as antenas do sistema e logo em sequência a apresentação de eventos separados por antena. Como foi dito, os campos *Tag ID* e *User Data* apresentam dados somente se uma *tag* permanece no alcance da respectiva antena, sendo assim, sabe-se que a peça com a *tag* está na antena 3, estação 4 do FMS (*Robot Arm*). Nesta tela também é possível obter a informação do tempo que a peça leva para transitar entre as estações, devido a presença da informação dos tempos de chegada e saída nos campos de *Tag Report*

Figura 79 - Leitura das peças com *tag* apresentadas na Dashboard do Node-RED.



Fonte: Autoria própria

A Figura 80 apresenta a tela da leitura manual e a escrita da *Tag ID* pela Dashboard. O botão *CHECAR TAG ID* faz uma leitura manual da *tag* no alcance das antenas, apresentando o resultado da leitura logo abaixo deste botão. A escrita é realizada no campo *Alterar Tag ID*, assim que uma nova *Tag ID* for escrita uma mensagem de sucesso ou erro na operação aparecerá logo abaixo. Outro indicativo de escrita na *tag* é pelo quadro de Tag report, o valor no Tag Report geral e da antena 1 atualizará com a *Tag ID* recém escrita.

Figura 80 – Leitura manual e escrita da Tag ID pela Dashboard do Node-RED



Fonte: Autoria própria

5.3 Leitura e Escrita Servidor OPC UA

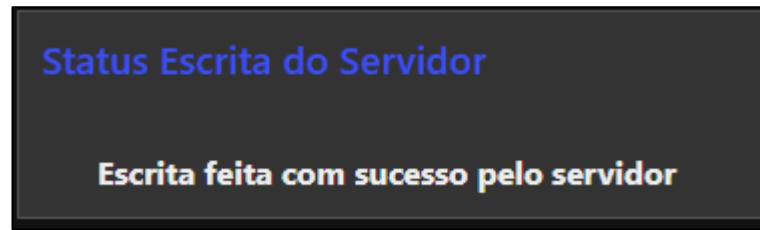
A Figura 81 apresenta todas as variáveis criadas na janela de *Data Acess* do UA Expert, o nome de cada variável é visto na coluna *Display Name*. Com características semelhantes a leitura e escrita via Dashboard, porém as opções de escrita são mais específicas. No servidor OPC UA a escrita é feita por antena, o que possibilita uma futura aplicação de múltiplas peças com *tags* ao mesmo tempo no FMS. E pelo servidor é possível fazer a escrita da *Tag ID* e do *User Data*. A variável *StatusEscrita* informa se a escrita pelo servidor foi um sucesso ou contém erros. A informação de ação da escrita pelo servidor também é encaminhada ao Dashboard, como é mostrado na Figura 82.

Figura 81 - Varáveis de leitura e escrita do servidor OPC UA apresentadas na janela de Data Access View do software UA Expert

Data Access View					
#	Server	Node Id	Display Name	Value	Datatype
1	OPC	NS4[String]...	Antena1_TagReport	event.tag.arrive tag_id=0xA111 user_data=0x000000000000000000000000, last=2024-07-18T15:45:07.661, antenna=1	String
2	OPC	NS4[String]...	Antena1_R_TagID	tag_id=0xA111	String
3	OPC	NS4[String]...	Antena1_R_UserData	user_data=0x00000000000000000000000000000000	String
4	OPC	NS4[String]...	Antena2_TagReport	event.tag.depart tag_id=0xBB22, user_data=, last=2024-07-18T15:33:29.331, antenna=2	String
5	OPC	NS4[String]...	Antena2_R_TagID		String
6	OPC	NS4[String]...	Antena2_R_UserData		String
7	OPC	NS4[String]...	Antena3_TagReport	event.tag.arrive tag_id=0xBB22, user_data=, first=2024-07-18T15:35:20.124, antenna=3	String
8	OPC	NS4[String]...	Antena3_R_TagID		String
9	OPC	NS4[String]...	Antena3_R_UserData		String
10	OPC	NS4[String]...	Antena4_TagReport	event.tag.depart tag_id=0xBB22, user_data=, last=2024-07-18T15:36:13.472, antenna=4	String
11	OPC	NS4[String]...	Antena4_R_TagID		String
12	OPC	NS4[String]...	Antena4_R_UserData		String
13	OPC	NS2[String]...	StatusEscrita	Escrita feita com sucesso pelo servidor	String
14	OPC	NS4[String]...	Antena1_W_TagID	0xA111	String
15	OPC	NS4[String]...	Antena1_W_UserData		String
16	OPC	NS4[String]...	Antena2_W_TagID		String
17	OPC	NS4[String]...	Antena2_W_UserData		String
18	OPC	NS4[String]...	Antena3_W_TagID		String
19	OPC	NS4[String]...	Antena3_W_UserData		String
20	OPC	NS4[String]...	Antena4_W_TagID		String
21	OPC	NS4[String]...	Antena4_W_UserData		String

Fonte: Autoria própria

Figura 82 - Indicação na Dashboard de sucesso na escrita pelo servidor.



Fonte: Autoria própria

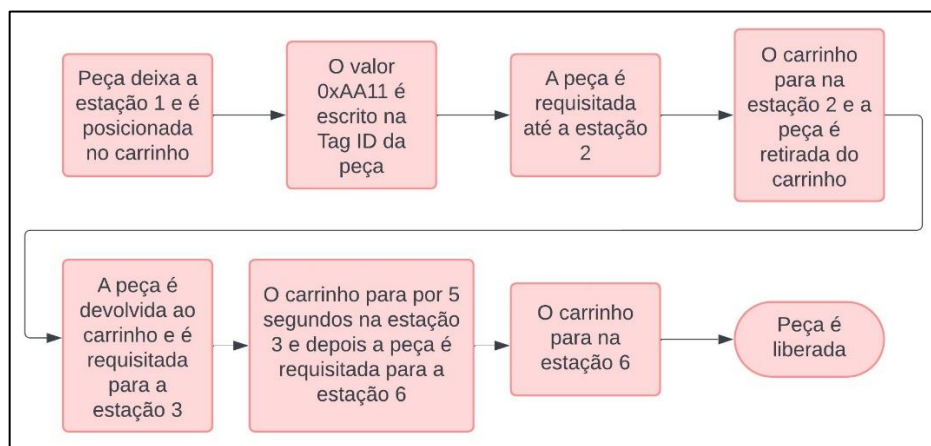
#### 5.4 Percurso da peça FMS.

Dois testes de validação do percurso da peça no FMS conforme a escrita foram realizados e gravados em vídeo, um com valor escrito de 0xAA11 no *Tag ID* e o outro com o valor de 0xBB22.

##### 5.4.1 Peça 0xAA11

A Figura 83 apresenta o fluxo do comportamento do sistema quando o valor da escrita na *Tag ID* é 0xAA11.

Figura 83 - Funcionamento do sistema com o valor 0xAA11.



Fonte: Autoria própria

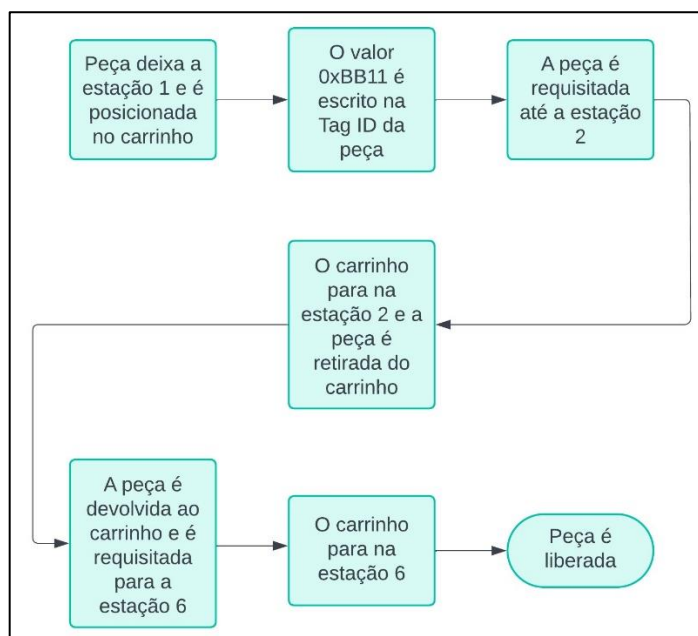
Vídeo do funcionamento do sistema com o valor da escrita de 0xAA11: [vídeo 0xAA11](#)

No vídeo, após a realização da escrita na *tag* pela antena 1, ocorre paradas na estação 2 *Processing* (Manipulação da peça) e estação 3 *Vision* (Parada por tempo) e então é a peça encaminhada para a última estação.

### 5.4.2 Peça 0xBB22

A Figura 84 apresenta o fluxo do comportamento do sistema quando o valor da escrita na *Tag ID* é 0xBB22.

Figura 84 - Funcionamento do sistema com o valor 0xBB11



Fonte: Autoria própria

Vídeo do funcionamento do sistema com o valor da escrita de 0xBB222: [vídeo 0xBB22](#)

No vídeo, após a realização da escrita na *tag* pela antena 1, ocorre uma parada apenas na estação 2 *Processing* e então a peça é encaminhada direto para a última estação.

## 5.5 Desafios e Discussões

Nas questões do desenvolvimento relacionados ao servidor OPC UA e a apresentação dos dados na Dashboard, não aconteceram grandes dificuldades, toda sua realização aconteceu em um progresso constante com resultados satisfatórios desde o início. Ainda assim, dois desafios foram encontrados durante o desenvolvimento do trabalho.

Um desafio encontrado foi na realização da configuração das antenas do leitor, inicialmente, mesmo com a peça parada em frente as antenas 3 e 4 o sinal oscilava entre eventos de saída e chegada, principalmente da antena 4. E ocasionalmente também ocorriam erros na execução da escrita do *Tag ID* pela antena 1. A primeira solução pensada para estes problemas foi aumentar o consumo dessas antenas para consequentemente aumentar seu alcance, porém isto fez com que a antena 3 começassem a captar sinais de peças que não estavam perto da sua

respectiva estação. Esses problemas foram resolvidos quando a configuração no RFID referente a frequência em que a *tag* responde ao leitor, o comando *return\_link\_freq*, que inicialmente estava com o valor 80 Kbps foi aumentado para o valor atual de 160 Kbps. Esta ação além de resolver os problemas de leitura trouxe uma estabilidade muito maior para as ações de escrita.

Outro desafio foi na realização da funcionalidade relacionada ao percurso da peça conforme a escrita, nas primeiras versões e testes realizados desta funcionalidade, o sinal contínuo recebido das antenas do leitor RFID causava interações indesejadas na programação no Node-RED, como interromper e travar o funcionamento do FMS, assim como mandar sinais de requisição para a estação errada. Este problema foi contornado com a utilização das funções *flow.set()* e *flow.get()* do Node-RED. Com elas foi possível criar variáveis que indicavam condições e ações para fluxos que não estavam ligados diretamente entre si, conseguindo assim criar uma ordem bem definida de prioridade de ações.

## 6 CONCLUSÃO

Este trabalho visava realizar a integração entre a tecnologia RFID e o padrão de comunicação OPC UA aplicado a um sistema de manufatura utilizando o ambiente do Node-RED para sua programação, acompanhando os seguintes objetivos: configuração automática do leitor RFID e suas antenas; leitura e escrita via Dashboard do Node-RED e via servidor OPC UA; e definição de percurso pelo sistema de manufatura com base na escrita realizada. Para a sua completa realização foram necessários diversos estudos das funcionalidades e características do leitor RFID, somados a estudos das funções e dos nós do Node-RED e entender suas interações entre o leitor e a comunicação OPC UA.

Foram necessários diversos estudos do funcionamento da tecnologia e do leitor RFID, dos padrões da comunicação OPC UA e das funcionalidades do ambiente do Node-RED para poder compreender como realizar a integração dessas três ferramentas de forma funcional e atender aos objetivos propostos.

A utilização do Node-RED para o desenvolvimento do projeto foi, sem dúvida, muito positiva, pois seu sistema de fluxos e funções predefinidas faziam que a realização da programação da interação com o leitor RFID e o servidor OPC UA fosse feita de uma forma bem intuitiva. O fluxo de informações era facilmente visualizado na programação, o que tornava simples a identificação dos erros e dos acertos.

Apesar de muitos resultados iniciais satisfatórios, também houve desafios na realização do projeto, fazendo com que fosse necessário a retomada nos estudos no comportamento do leitor RFID e uma pesquisa mais profunda das funcionalidades menos intuitivas do Node-RED para o funcionamento ideal do percurso da peça pelo FMS conforme escrita.

Com os principais desafios superados, pode-se concluir que o objetivo principal de integrar a tecnologia RFID e comunicação OPC UA aplicado a um sistema de manufatura foi atingido. Contudo, como recomendação para trabalhos futuros, o projeto ainda pode ser aprimorado tornando-o mais flexível, com a possibilidade da alteração da escrita durante a produção, fazendo assim o sistema reconfigurar o trajeto da peça pelo FMS automaticamente, levando em conta se é possível fazer a alteração, por exemplo, caso a peça já tenha passado pela estação 2 em produção, não permitir a alteração para peças que não passam nessa estação.

## REFERÊNCIAS

ALIEN TECHNOLOGY. **ALR-9610-AL linearly polarized RFID antenna & ALR-9610-BC circularly polarized RFID antenna**. Califórnia. 2004.

ALTUS. **Como o sistema supervisório ajuda na produtividade de uma empresa?** 2023. Disponível em: <https://www.altus.com.br/blog/busca/pagina/1?palavrachave=supervisórios&validadorantispa m=8185456>. Acesso em 24 jun. 2024.

FERENCZ, Katalin; DOMOKOS, József. **Using Node-RED platform in an industrial environment**. XXXV. Jubileumi Kandó Konferencia, Budapest, p. 52-63, 2019. Disponível em: [https://www.researchgate.net/publication/339596157\\_Using\\_Node-RED\\_platform\\_in\\_an\\_industrial\\_environment](https://www.researchgate.net/publication/339596157_Using_Node-RED_platform_in_an_industrial_environment). Acesso em 26 jun. 2024.

FIAIDHI, Jinan; MOHAMMED, Sabah. **Virtual care for cyber-physical systems (VH\_CPS): NODE-RED, community of practice and thick data analytics ecosystem**. Computer communications, v. 170, p. 84-94, 2021. Disponível em: <https://doi.org/10.1016/j.comcom.2021.01.029>. Acesso em 26 jun. 2024.

MEDEIROS, Maria Janienne Alves Pereira de et al. **Tecnologia RFID: contribuições de uso/aplicação em Unidades de Informação (Bibliotecas)**. 2011. Disponível em: <https://repositorio.ufpb.br/jspui/handle/123456789/2247>. Acesso em 25 jun. 2024.

MELO, Pablo Felipe Soares de. **Dispositivo de controle para a indústria 4.0 baseado no RAMI 4.0 e OPC UA**. São João da Boa Vista: Dissertação (mestrado) - Universidade Estadual Paulista “Júlio de Mesquita Filho”, Câmpus Experimental de São João da Boa Vista, 2020. Disponível em: <http://hdl.handle.net/11449/192851>. Acesso em 25 jun. 2024.

MELO, Pablo. **Introdução ao OPC UA (Open Platform Communications Unified Architecture)**. 2017. Disponível em: <https://embarcados.com.br/introducao-ao-opc-ua/>. Acesso em 25 jun. 2024

OPC FOUNDATION. **Unified Architecture**. 2017. Disponível em: <https://opcfoundation.org/about/opc-technologies/opc-ua/> Acesso em 25 jun. 2024.

OPENJS FOUNDATION & CONTRIBUTORS. **Node-RED**. 2019. Disponível em: <https://nodered.org>. Acesso em 26 de jun. 2024

PINHEIRO, José Maurício dos Santos. **Identificação por Radiofrequência: Aplicações e Vulnerabilidades da Tecnologia RFID**. Cadernos UniFOA, Volta Redonda, v. 1, n. 2, p. 18–32, 2017. DOI: 10.47385/cadunifoa. Disponível em: <https://revistas.unifoa.edu.br/cadernos/article/view/889>. Acesso em: 25 jun. 2024.



SANCHES, Heverton Bacca. **Monitoramento da produção e da eficiência de processos de manufatura usando RFID e internet das coisas**. Bauru: Dissertação (mestrado) - Universidade Estadual Paulista “Júlio de Mesquita Filho”, Faculdade de Engenharia, Bauru, 2018. Disponível em: <http://hdl.handle.net/11449/157236>. Acesso em 24 jul. 2024.

SACOMANO, José B.; GONÇALVES, Rodrigo F.; BONILLA, Sílvia H. **Indústria 4.0: conceitos e fundamentos**. São Paulo: Editora Blucher, 2018. E-book. ISBN 9788521213710. Disponível em: <https://app.minhabiblioteca.com.br/#/books/9788521213710/>. Acesso em: 24 jun. 2024.

SIRIT INC. **INfinity 510 Quick Start Guide**. Ontário, v.3.1, 2010.

SIRIT INC. **INfinity 510 Protocol Reference Guide**. Ontário, v.2.02, 2008.

SOUSA, Daniel Ribeiro de; GODOY, Eduardo Paciência; CALDANA, Vitor Mendes. **Upgrading legacy systems for Industry 4.0 with Node-RED and OPC-UA**. 2024. Disponível em: [https://www.researchgate.net/publication/378654188\\_Upgrading\\_legacy\\_systems\\_for\\_Industry\\_40\\_with\\_Node-RED\\_and\\_OPC-UA](https://www.researchgate.net/publication/378654188_Upgrading_legacy_systems_for_Industry_40_with_Node-RED_and_OPC-UA). Acesso em: 24 jul. 2024.

TT ELECTRONICS. **RFID: The Technology Making Industries Smarter**. 2021. Disponível em: <https://www.ttelectronics.com/blog/rfid-technology/> Acesso em: 25 jun. 2024.

UNIFIED AUTOMATION GMBH. **UaExpert—A Full-Featured OPC UA Client**. 2023. Disponível em: [https://www.unified-automation.com/products/development-tools/uaexpert.html?gad\\_source=1&gclid=Cj0KCQjw1qO0BhDwARIsANfnkv-iuVF\\_3VajmJv4Bq5hfFvZx0adfLD664F\\_9HABfTHT90tv6OPW34EaAiRmEALw\\_wcB](https://www.unified-automation.com/products/development-tools/uaexpert.html?gad_source=1&gclid=Cj0KCQjw1qO0BhDwARIsANfnkv-iuVF_3VajmJv4Bq5hfFvZx0adfLD664F_9HABfTHT90tv6OPW34EaAiRmEALw_wcB). Acesso em: 28 jun. 2024

WEINSTEIN, Ron. **RFID: a technical overview and its application to the enterprise. IT professional**, v. 7, n. 3, p. 27-33, 2005. Disponível em: <https://ieeexplore.ieee.org/abstract/document/1490473>. Acesso em 25 jun. 2024.

XAVIER, Andreza Batista et al. **A Automação Industrial Como Solução e Não Como Ameaça Aos Trabalhadores**. GeSe: Revista de Gestão e Secretariado. v.14, n.6, p. 9019–9032, 2023. Disponível em: [https://unesp.primo.exlibrisgroup.com/permalink/55UNESP\\_INST/1j0ltst/cdi\\_proquest\\_journals\\_2834510477](https://unesp.primo.exlibrisgroup.com/permalink/55UNESP_INST/1j0ltst/cdi_proquest_journals_2834510477). Acesso em: 24 jun. 2024.