

Pablo Felipe Soares de Melo

**Dispositivo de Controle para a Indústria 4.0 baseado no RAMI 4.0 e
OPC UA**

Sorocaba
2020

Pablo Felipe Soares de Melo

**Dispositivo de Controle para a Indústria 4.0 baseado no RAMI 4.0 e
OPC UA**

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica, junto ao Programa de Pós-Graduação em Engenharia Elétrica, Interunidades, entre o Instituto de Ciência e Tecnologia de Sorocaba e o Campus de São João da Boa Vista da Universidade Estadual Paulista “Júlio de Mesquita Filho”.

Orientador: Prof. Dr. Eduardo Paciência Godoy

Sorocaba
2020

M528d

Melo, Pablo Felipe Soares de

Dispositivo de Controle para a Indústria 4.0 baseado no RAMI 4.0 e OPC UA / Pablo Felipe Soares de Melo. -- São João da Boa Vista, 2020

127 p.

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Câmpus Experimental de São João da Boa Vista, São João da Boa Vista

Orientador: Eduardo Paciência Godoy

1. Engenharia Elétrica. 2. Automação. 3. Internet das coisas. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Câmpus Experimental de São João da Boa Vista. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

ATA DA DEFESA PÚBLICA DA DISSERTAÇÃO DE MESTRADO DE PABLO FELIPE SOARES DE MELO, DISCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA, DO INSTITUTO DE CIÊNCIA E TECNOLOGIA - CÂMPUS DE SOROCABA.

Aos 22 dias do mês de maio do ano de 2020, às 09:00 horas, no(a) Sala Virtual, reuniu-se a Comissão Examinadora da Defesa Pública, composta pelos seguintes membros: Prof. Dr. EDUARDO PACIÊNCIA GODOY - Orientador(a) do(a) Departamento de Engenharia de Controle e Automação / Instituto de Ciência e Tecnologia - UNESP - Câmpus de Sorocaba, Prof. Dr. IVANDO SEVERINO DINIZ do(a) Departamento de Engenharia de Controle e Automação / Instituto de Ciência e Tecnologia / UNESP / Sorocaba, Prof. Dr. JOSÉ LUIZ ANTUNES DE ALMEIDA do(a) Coordenação de Eletrônica Automotiva / Faculdade Tecnológica - Sorocaba, sob a presidência do primeiro, a fim de proceder a arguição pública da DISSERTAÇÃO DE MESTRADO de PABLO FELIPE SOARES DE MELO, intitulada **Dispositivo de Controle para a Indústria 4.0 baseado no RAMI 4.0 e OPC UA**. Após a exposição, o discente foi arguido oralmente pelos membros da Comissão Examinadora, tendo recebido o conceito final: APROVADO. Nada mais havendo, foi lavrada a presente ata, que após lida e aprovada, foi assinada pelos membros da Comissão Examinadora.

Prof. Dr. EDUARDO PACIÊNCIA GODOY

Prof. Dr. IVANDO SEVERINO DINIZ

Prof. Dr. JOSÉ LUIZ ANTUNES DE ALMEIDA

Pablo Felipe Soares de Melo

**Dispositivo de Controle para a Indústria 4.0 baseado no RAMI 4.0 e
OPC UA**

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica, junto ao Programa de Pós-Graduação em Engenharia Elétrica, Interunidades, entre o Instituto de Ciência e Tecnologia de Sorocaba e o Campus de São João da Boa Vista da Universidade Estadual Paulista “Júlio de Mesquita Filho”.

Comissão Examinadora

Prof. Dr. Eduardo Paciência Godoy
UNESP – Câmpus de Sorocaba
Orientador

Prof. Dr. Ivando Severino Diniz
UNESP – Câmpus de Sorocaba

Prof. Dr. José Luiz Antunes de Almeida
FATEC – Câmpus de Sorocaba

Sorocaba
22 de maio de 2020

“Ninguém vai bater mais forte do que a vida. Não importa como você bate e sim o quanto aguenta apanhar e continuar lutando; o quanto pode suportar e seguir em frente. É assim que se ganha.”

Rocky Balboa

AGRADECIMENTOS

À minha esposa Maria Vitória Dias Antunes por toda compreensão e apoio incondicional durante os momentos mais difíceis.

Aos meus avós Maria (*in memoriam*) e Oscarlino por terem realizado o papel de pais, ensinando-me a ser uma pessoa honesta e dedicada.

Ao Professor Eduardo Paciência Godoy pela confiança, amizade, dedicação e orientação durante todas as etapas do desenvolvimento deste trabalho.

Aos colegas Rodrigo Rolle e Israel Ferreira pelas trocas de experiências ao longo da realização desta pesquisa.

RESUMO

A Quarta Revolução Industrial ou I4.0 representa a evolução dos sistemas produtivos atuais a partir da união entre tecnologias de automação industrial e tecnologia da informação. A inovação técnica da I4.0 é caracterizada pela integração vertical e horizontal dos sistemas de fabricação, gerenciamento ao longo do ciclo de vida do produto e a descentralização de recursos computacionais. Para que isso ocorra, foi definido o Modelo de Referência de Arquitetura para a Indústria 4.0 (RAMI 4.0) que compatibiliza os elementos da I4.0 em um modelo de camadas tridimensional. Um dos eixos horizontais do RAMI 4.0 define os níveis hierárquicos de elementos de uma indústria integrada em rede, indo desde o “produto” e “Dispositivo de Controle” até o “mundo conectado”. O grande desafio para adoção do RAMI 4.0 se encontra no desenvolvimento de soluções (equipamentos, sistemas e padrões) que suportem as funcionalidades de cada camada e as interações requeridas entre os seus elementos. Dessa forma, apesar de sua aceitação, o uso do RAMI 4.0 ainda é restrito às instituições de pesquisa e primeiros casos de execuções individuais, pois o modelo não é um guia de implementação, o que dificulta sua implementação completa. Com isso, este trabalho foca no estudo deste modelo de referência de arquitetura para o desenvolvimento de um Dispositivo de Controle de código aberto (open-source) para a Indústria 4.0 baseado no RAMI 4.0 e no seu respectivo protocolo de comunicação (padronizado e interoperável), denominado OPC UA. Este Dispositivo de Controle fornece as funcionalidades de programação de processos, identificação de peças por rádio frequência, comunicação em rede e supervisão dos equipamentos, disponibilizando as informações do processo controlado de forma padronizada, para qualquer tipo de plataforma conectada à rede e/ou Internet. O Dispositivo de Controle integra soluções de código aberto de hardware, software, comunicação e programação, cobrindo os relacionamentos entre três camadas do RAMI 4.0 (Ativos, Integração e Comunicação). Experimentos em um cenário de laboratório I4.0 foram utilizados para a validação da operação do Dispositivo de Controle. Finalmente, são discutidas algumas oportunidades e propostas para a atualização do Dispositivo de Controle em um Componente I4.0 da RAMI 4.0.

Palavras-chave: *Indústria 4.0, RAMI 4.0, OPC UA, IIoT, Dispositivo de Controle.*

ABSTRACT

The Fourth Industrial Revolution or I4.0 represents the evolution of current production systems based on the union between industrial automation technologies and information technology. The technical innovation of I4.0 is characterized by the vertical and horizontal integration of the manufacturing systems, management throughout the product life cycle and the decentralization of computational resources. For this to happen, the Reference Model for Industry 4.0 (RAMI 4.0) was defined, which makes the elements of I4.0 compatible in a three-dimensional layered model. One of the horizontal axes of RAMI 4.0 defines the hierarchical levels of elements of an integrated industry, ranging from the “product” and “control device” to the “connected world”. The big challenge for the adoption of RAMI 4.0 is found in the development of solutions (equipment, systems and standards) that support the functionalities of each layer and the required interactions between its elements. Thus, despite its acceptance, the use of RAMI 4.0 is still restricted to research institutions and first cases of individual executions, as the model is not an implementation guide, which makes it difficult to fully implement it. Thus, this work focuses on the study of this reference architecture model for the development of an open source control device (open-source) for Industry 4.0 based on RAMI 4.0 and its respective communication protocol (standardized and interoperable), called OPC UA. This control device provides the functionalities of process programming, identification of parts by radio frequency, network communication and supervision of equipment, providing the information of the controlled process in a standardized way, for any type of platform connected to the network and / or the Internet. The Control Device integrates open source solutions for hardware, software, communication and programming, covering the relationships between three layers of RAMI 4.0 (Assets, Integration and Communication). Experiments in an I4.0 laboratory setting were used to validate the operation of the Control Device. Finally, some opportunities and proposals for updating the Control Device in an I4.0 Component of RAMI 4.0 are discussed.

Keywords: *Industry 4.0, RAMI 4.0, OPC UA, Control Device.*

LISTA DE FIGURAS

Figura 1 - Correlação entre IoT, Indústria e IIoT	17
Figura 2 - Linha do tempo das revoluções industriais	23
Figura 3 - Principais tecnologias para a composição da Indústria 4.0.....	25
Figura 4 - Estrutura Tridimensional do RAMI 4.0	32
Figura 5 - Derivação dos Níveis de Hierarquia do RAMI 4.0	33
Figura 6 - Componente da Indústria 4.0 inserido no RAMI 4.0.....	36
Figura 7 - Modelo de um I4.0C (Objetos x AS x RAMI 4.0).....	37
Figura 8 - Estrutura de Informação de um Shell de Administração	39
Figura 9 - Visão geral das especificações do OPC Clássico.....	41
Figura 10 - Objetos criados por um cliente para acessar dados de um servidor.....	42
Figura 11 - Objetos criados por um cliente OPC para receber eventos	44
Figura 12 - Aplicação do OPC UA dentro da pirâmide de automação	46
Figura 13 - Modelo de objeto unificado para OPC UA	48
Figura 14 - Espaço de endereço de um servidor OPC UA.....	49
Figura 15 - Classe de Nós a partir do Nó Base	50
Figura 16 - Exemplo de alguns serviços utilizados em uma comunicação entre cliente UA e servidor UA.....	57
Figura 17 - Processo de Descoberta com Servidor de Descoberta.....	59
Figura 18 - Arquitetura do sistema OPC UA	62
Figura 19 - Arquitetura proposta para o desenvolvimento de um Dispositivo de Controle para Indústria 4.0	72
Figura 20 - Relação entre as tecnologias do Dispositivo de Controle e o RAMI4.0 ..	73
Figura 21 - Programação em Ladder feita através do PLCopen Editor	78
Figura 22 - Arquitetura do projeto OpenPLC	79
Figura 23 - Componentes de um sistema RFID	82
Figura 24 – Distribuição dos componentes na Unipi1.1.	85
Figura 25 - Tecnologias do Dispositivo de Controle contidas nas camadas de Integração e Comunicação do RAMI4.0.....	87
Figura 26 - Mapeamentos de pinos Modbus para UniPI1.1 no OpenPLC.....	88
Figura 27 - API de comunicação entre OpenPLC e Servidor OPC UA	89
Figura 28- Espaço de endereço para a proposta	92
Figura 29 - Interface Gráfica gerada pelo opcua-client-gui"	93

Figura 30 - Relação entre as camadas de Integração e Ativos Técnicos.....	94
Figura 31 - Etiquetas RFID fixadas nas peças	94
Figura 32 – Interações entre elementos dos eixos RAMI 4.0 no experimento de validação de dispositivos de controle.	98
Figura 33 - Elementos contidos nas camadas de integração e ativos técnicos	100
Figura 34 - Estação Sorting com o ambiente de testes para o Dispositivo de Controle.....	101
Figura 35 - Programa ladder desenvolvido para validação do Dispositivo de Controle.	102
Figura 36 - Upload do programa no servidor do OpenPLC	103
Figura 37 - Seleção da UniPi1.1 e compilação do driver referente ao sistema RFID	103
Figura 38 - Validação da Identificação de Peças pelo sistema RFID no OpenPLC	104
Figura 39 - Estrutura para validar Integração entre OpenPLC e OPC UA	105
Figura 40 - Etapas de configurações para a comunicação com o servidor	106
Figura 41 - Tela de validação do certificado.....	107
Figura 42 - Tela representando conexão estabelecida.	107
Figura 43 - Validação dos Objetos no Espaço de Endereço	108
Figura 44 – Validação do Espaço de Endereço para variáveis do OpenPLC.	108
Figura 45 – Validação das variáveis referentes a identificação de peças pelo sistema RFID.	109
Figura 46 – Validação das variáveis referentes ao hardware durante a execução do proceso controlado.....	109
Figura 47 - Validação dos Atributos no Espaço de Endereço	110
Figura 48 – Assinatura criada para a identificação da peça preta e sua respectiva saída digital.	111
Figura 49 – Monitoramento de todas as variáveis do processo controlado.....	111
Figura 50 – Clientes OPC UA (números de 1 a 5) conectados com o servidor OPC UA (A)	112
Figura 51 - Dados operacionais durante o experimento com várias conexões de usuário (cliente) para o Dispositivo de Controle (servidor)	113
Figura 52 - Cliente OPC UA implementando uma Interface Gráfica com o Usuário.	114

LISTA DE TABELAS

Tabela 1 - Requisitos da Indústria 4.0 e soluções apresentadas pelo OPC UA.....	30
Tabela 2 - Atributo para a classe de Nó <i>Object</i>	51
Tabela 3 - Atributo para a classe de Nó <i>ObjectType</i>	52
Tabela 4 - Atributo para a classe de Nó <i>Variable</i>	53
Tabela 5 - Atributo para a classe de Nó <i>DataType</i>	54
Tabela 6 - Atributo para a classe de Nó <i>VariableType</i>	54
Tabela 7 - Atributo para a classe de Nó <i>Method</i>	55
Tabela 8 - Atributo para a classe de Nó <i>ReferenceType</i>	55
Tabela 9 - Atributo para a classe de Nó <i>View</i>	56
Tabela 10 - Análise Comparativa entre Implementações de Software de Código Aberto do OPCUA	70
Tabela 11- Espaço de endereço de E/S no OpenPLC mapeado para o espaço de endereço Modbus.....	79
Tabela 12- Recursos de Hardware da placa de expansão Unipi1.1.....	84
Tabela 13- Mapeamento Modbus para UniPi1.1	89
Tabela 14- Mapeamento Modbus para sistema RFID dentro do OpenPLC	97
Tabela 15 - Peças e seus respectivos valores de Tag IDs.....	97
Tabela 16 - Sequência de conexão dos clientes com o servidor OPC UA.	113

LISTA DE SIGLAS

A&E	<i>Alarm & Event</i>
AIM	<i>Association for Automation Data Capture, Identification and Mobility</i>
ANSI	<i>American National Standards Institute</i>
API	<i>Application Program Interface</i>
ARM	<i>Advanced RISC Machine</i>
AS	<i>Administration Shell</i>
B2B	<i>Business to Business</i>
BACnet	<i>Building Automation and Control Networking Protocol</i>
COM	<i>Component Object Model</i>
CPS	<i>Cyber-Physical System</i>
CLI	<i>Command Line Interface</i>
CTT	<i>Compliance Test-Tools</i>
DA	<i>Data Access</i>
DIN	<i>German Industrial Standard</i>
DCOM	<i>Distributed Component Object Model</i>
ERP	<i>Enterprise Resource Planning</i>
FDB	<i>Function Block Diagram</i>
FDI	<i>Field Device Integration</i>
FF	<i>FieldBus Foundation</i>
GASI	<i>Grupo de Automação e Sistemas Integráveis</i>
GPL GNU	<i>General Public License</i>
GUI	<i>Graphical User Interface</i>
HCF	<i>HART Communication Foundation</i>
HDA	<i>Historical Data Access</i>
IHM	<i>Human Machine Interface</i>
HTTP	<i>Hyper Text Transfer Protocol</i>
HTTP	<i>Hyper Text Transfer Protocol Secure</i>
I4.0	<i>Industry 4.0</i>
I4.0C	<i>Industry 4.0 Component</i>
ICT	<i>Industrial Control Technology</i>
IEC	<i>International Electrotechnical Commission</i>
IIoT	<i>Industrial Internet of Things</i>
IL	<i>Instruction List</i>
IoS	<i>Internet of Services</i>
IoT	<i>Internet of Things</i>
ISA	<i>International Society of Automation</i>
JSON	<i>JavaScript Object Notation</i>
LD	<i>Ladder Diagram</i>
LGPL	<i>Lesser General Public License</i>
M2M	<i>Machine to Machine</i>
MIDS	<i>Management Development Institute of Singapore</i>
MES	<i>Manufacturing Execution System</i>

MIT	<i>Massachusetts Institute of Technology</i>
OLE	<i>Object Linking and Embedding</i>
OPC	<i>OLE for Process Control</i>
OPC UA	<i>Open Platform Communication Unified Architecture</i>
PC	<i>Personal Computer</i>
PI	<i>PROFIBUS International</i>
PLC	<i>Programmable Logic Controller</i>
POSIX	<i>Portable Operating System Interface</i>
PPC	<i>Power Personal Computer</i>
QoS	<i>Quality of Services</i>
RAMI	<i>Reference Architecture Model for Industry</i>
RCL	<i>Reciprocal Public License</i>
RFID	<i>Radio Frequency Identification</i>
SBC	<i>Single Board Computer</i>
SCADA	<i>Supervisory Control and Data Acquisition</i>
SDK	<i>Software Development Kit</i>
SFC	<i>Sequential Function Chart</i>
SGAM	<i>Smart Grid Architecture Model</i>
SOA	<i>Service Oriented Architecture</i>
ST	<i>Structured Text</i>
TA	<i>Tecnologia da Automação</i>
TCP/IP	<i>Transmission Control Protocol/Internet Protocol</i>
TI	<i>Tecnologia da Informação</i>
TIC	<i>Tecnologia da Informação e Comunicação</i>
UA	<i>Unified Architecture</i>
URL	<i>Uniform Resource Locator</i>

SUMÁRIO

1	INTRODUÇÃO	17
1.1	Justificativa.....	17
1.2	Objetivos	20
1.3	Estrutura e Conteúdo	20
2	REVISÃO BIBLIOGRÁFICA	22
2.1	Indústria 4.0	22
2.2	Contextualização do RAMI 4.0	26
2.3	Comunicação para Indústria 4.0 e RAMI 4.0.....	29
3	RAMI 4.0 - MODELO DE REFERÊNCIA DE ARQUITETURA PARA A INDÚSTRIA 4.0.....	32
3.1	Eixo de Níveis de Hierarquia.....	32
3.2	Eixo do Ciclo de Vida e Cadeia de Valores.....	34
3.3	Eixo das Camadas	35
3.4	Componente da Indústria 4.0	36
3.5	Shell de Administração.....	37
4	OPC UA - PADRÃO DE COMUNICAÇÃO PARA INDÚSTRIA 4.0	40
4.1	Visão Geral do OPC Clássico	40
4.1.1	OPC DA (Data Access).....	41
4.1.2	OPC HDA (Historical Data Access)	42
4.1.3	OPC A&E (Alarm & Events)	43
4.1.4	OPC XML DA.....	44
4.2	Principais limitações das especificações OPC	45
4.3	OPC UA – Plataforma de Comunicações Aberta de Arquitetura Unificada.....	45
4.4	Modelagem de Informação do OPC UA	48
4.4.1	Espaço de Endereço do Servidor OPC UA.....	49
4.4.2	Classe de Nós do Espaço de Endereço	51
4.4.2.1	Objeto (<i>Object</i>).....	51
4.4.2.2	Tipo de Objeto (<i>ObjectType</i>)	52
4.4.2.3	Variável (<i>Variable</i>).....	52
4.4.2.4	Tipo de Dado (<i>DataType</i>).....	53
4.4.2.5	Tipo de Variável (<i>VariableType</i>)	54
4.4.2.5	Método (<i>Method</i>).....	54
4.4.2.6	Tipo de Referência (<i>ReferenceType</i>)	55
4.4.2.7	Visualização (<i>View</i>).....	55
4.5	Conjunto de Serviços	56
4.5.1	Canal de Segurança (<i>SecureChannel</i>)	57
4.5.2	Descoberta (<i>Discovery</i>)	58
4.5.3	Sessão (<i>Session</i>).....	60
4.5.4	Gerenciamento de Nó (<i>NodeManagement</i>)	60
4.5.5	Visualização (<i>View</i>).....	60
4.5.6	Pergunta (<i>Query</i>)	60
4.5.7	Atributo (<i>Attribute</i>).....	61
4.5.8	Método (<i>Method</i>).....	61
4.5.9	Item Monitorado (<i>MonitoredItem</i>).....	61
4.5.10	Assinatura (<i>Subscription</i>).....	62
4.6	Arquitetura do OPC UA	62

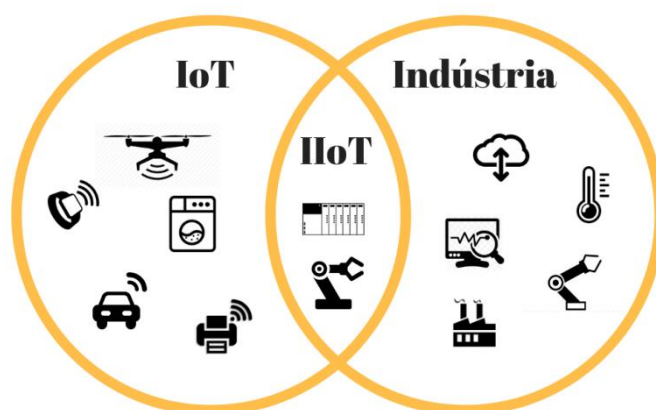
4.6.1	Cliente OPC UA	63
4.6.2	Servidor OPC UA	63
4.7	Mapeamentos de Tecnologias	64
4.7.1	Codificação dos Dados	64
4.7.2	Segurança dos Dados.....	65
4.7.3	Transporte dos Dados.....	65
4.8	Camadas de Software do OPC UA	66
5	ANÁLISE COMPARATIVA ENTRE AS IMPLEMENTAÇÕES OPEN SOURCE DE SOFTWARE DO OPC UA	67
5.1	Node-OPCUA.....	67
5.2	Open62541	68
5.3	Eclipse Milo	68
5.4	ASNeg.....	69
5.5	UA.NET	69
5.6	FreeOPCUA	70
5.7	Análise comparativa com base nos requisitos estabelecidos.....	70
6	MATERIAIS E METÓDOS.....	72
6.1	Proposta do Trabalho.....	72
6.1.1	A camada de Ativos Técnicos para o Dispositivo de Controle	73
6.1.2	A camada de Integração para o Dispositivo de Controle	74
6.1.3	A camada de Comunicação para o Dispositivo de Controle	75
6.2	Tecnologias para o Dispositivo de Controle	77
6.2.1	OpenPLC	77
6.2.2	Node.js.....	80
6.2.3	Sistema RFID	81
7	DESENVOLVIMENTO.....	84
7.1	Conexão entre as camadas de Integração e Comunicação	86
7.1.1	API entre OpenPLC e Servidor OPC UA	87
7.1.2	Desenvolvimento do Servidor OPC UA	90
7.1.3	Comunicação entre IHM (Cliente OPC UA) e Servidor OPC UA.....	92
7.2	Conexão entre as camadas de Ativos Técnicos e Integração.....	93
7.2.1	Identificação de Peças.....	94
8	RESULTADOS	98
8.1	Validação da conexão entre os Ativos Técnicos e a Integração	99
8.1.1	Estação Sorting.....	100
8.1.2	Processo de Manufatura.....	101
8.1.3	Controle do processo pelo OpenPLC	102
8.2	Validação da conexão entre a Integração e a Comunicação	104
8.2.1	Execução do Servidor OPC UA e conexão com o UAExpert.....	105
8.2.2	Servidor OPC UA conectando com diferente tipos de clientes	112
8.3	Discussões e Observações Finais	115
9	CONCLUSÕES	117
10	REFERÊNCIAS	119

1 INTRODUÇÃO

1.1 Justificativa

Com o avanço das tecnologias que norteiam a Internet das Coisas (*Internet of Things* - IoT), Tan e Wang (2010) constatam que no futuro, não serão apenas humanos dialogando com outros humanos, ou pessoas acessando informações; serão humanos conversando com máquinas, sendo que estas poderão conversar entre si, constituindo assim, o conceito de máquina para máquina (*Machine to Machine*- M2M). Mediante a essas possibilidades, a Internet das Coisas ganha atratividade dentro do cenário industrial, abrangendo setores como logística, fabricação, varejo entre outros (XU et al., 2014). A combinação entre a IoT e a Indústria, representada pela Figura 1, derivou para a criação do conceito de Internet das Coisas Industrial (*Industrial Internet of Things* - IIoT), sendo este definido pela utilização de tecnologias que afetam diretamente a automação em infraestrutura crítica e melhoram a produtividade empresarial (HASSANZADEH et al., 2015).

Figura 1 - Correlação entre IoT, Indústria e IIoT



Fonte: Elaborado pelo autor

Outra característica importante no que diz respeito à IIoT é que ela se tornou um dos principais elementos da Fabricação Inteligente (*Smart Manufacturing*) e da Quarta Revolução Industrial (Indústria 4.0) (ERUVANKAI et.al., 2017). O termo Indústria 4.0 (abreviado por I4.0) foi introduzido pela primeira vez na feira de Hannover em 2011, devido a uma iniciativa do governo alemão, adotada como parte do plano de ação estratégica de alta tecnologia para 2020 (KAGERMANN; WAHLSTER; HELBIG, 2013; SNIDERMAN; MAHTO; COTTELEER, 2016). Kagermann, Wahlster e Helbig (2013)

apontam que projetos de demonstração devem ser desenvolvidos e disponibilizados ao mercado assim que possíveis, levando em consideração sua implementação de forma estratégica e a adaptação de tecnologias e experiências básicas já existentes aos requisitos de fabricação. Chantanayingyong (2016) destaca que a I4.0 foi tema do Fórum Econômico Mundial que ocorreu em Davos (Suíça), em 2016, contando com a participação de mais de 2.500 líderes de empresas, governos, organizações internacionais, sociedade civil, academia e mídia, para a discussão dos impactos proporcionados pela Quarta Revolução Industrial. No Brasil, a Indústria 4.0 é conhecida como Manufatura Avançada. Nos Estados Unidos, o termo recebeu o nome de *Smart Industry* (Indústria Inteligente) (A VOZ DA INDÚSTRIA, 2016).

A Indústria 4.0 pode ser definida como um novo conceito que representa a evolução dos sistemas produtivos atuais a partir da união entre novas tecnologias de automação industrial (TA) e tecnologia da informação (TI) (BERGER, 2014). Segundo Bangemann et al. (2016), a I4.0 apresenta um novo cenário com relação a produção industrial que é descrito por três aspectos:

- ✓ Um novo nível de organização e controle de toda cadeia de valor com o ciclo de vida do produto;

- ✓ A disponibilidade de todas as informações relevantes em tempo real, sendo possível através da interconexão de todas as instâncias que participam dos processos de criação de valor;

- ✓ A criação de redes de valores entre empresas auto-organizáveis, dinâmicas e em tempo real por meio da interconectividade de humanos, “coisas” e sistemas de acordo com suas habilidades.

Conforme Wang, Towara e Anderl (2017), analisando as características das três primeiras revoluções industriais, a inovação técnica da quarta revolução, baseia-se nas seguintes condições: integração vertical e horizontal de sistemas de fabricação, engenharia digital contínua ao longo do ciclo de vida do produto e a descentralização de recursos computacionais. Para que isso ocorra, é necessário um modelo comum que agrupe tais condições dentro de uma arquitetura de referência para a I4.0 (ADOLPHS et al., 2015; DORST et al., 2016). Pensando em um processo de desenvolvimento sistemático e sustentável dentro da indústria, é essencial à construção de arquiteturas de sistemas em um conjunto de modelos de referências estáveis, consensuais e padronizados (BANGEMANN et al., 2016). Em consequência

dessas premissas, em abril de 2015 foi apresentado na feira de Hannover, um Modelo de Referência de Arquitetura para a Indústria 4.0, denominado RAMI 4.0 (*Reference Architectural Model for Industry 4.0*) (ADOLPH et al., 2016). O RAMI 4.0 compatibiliza os elementos inseridos na I4.0 em um modelo de camadas tridimensional, onde a partir dessa estrutura, tecnologias baseadas na I4.0 podem ser classificadas e desenvolvidas (HANKEL; REXROTH, 2015). Sendo assim, inter-relações complexas podem ser divididas em clusters menores e mais simples (GOTZE, 2016). Em contrapartida, apesar de sua grande aceitação, alguns estudos relatam algumas limitações em termos de praticidade com relação ao RAMI 4.0. Citando por exemplo, Wang, Towara e Anderl (2017) salientam que o modelo em si é um pouco abstrato, generalizando de forma demasiada a sua aplicação, a qual tem ficado inicialmente restrita às instituições de pesquisa e projetos piloto.

Diversos trabalhos recentes têm focado no estudo da estrutura do RAMI 4.0 para o desenvolvimento de soluções compatíveis com os requisitos da aplicação da I4.0 (CONTRENAS; GARCIA; PASTRANA, 2017; PISCHING et al., 2017). O grande desafio para adoção do RAMI 4.0 se encontra no desenvolvimento de soluções (equipamentos, sistemas e padrões) que suportem as funcionalidades de cada camada e as interações requeridas entre os elementos de cada camada. Como atualmente a I4.0 representa um conceito, não especificando por exemplo tecnologias de comunicação, ferramentas de integração e padrões de programação para serem usados, a maioria dos trabalhos devem buscar diferentes tecnologias que possam ser integradas e prover as funcionalidades requeridas por estas novas aplicações industriais.

Fundamentando-se nas necessidades e desafios citados, este projeto foca no estudo das camadas e interações entre elementos do RAMI 4.0 para desenvolver um Dispositivo de Controle para aplicações da Indústria 4.0 que implemente funcionalidades do elemento “Dispositivo de Controle” presente no eixo horizontal de Níveis Hierárquicos do RAMI 4.0.

1.2 Objetivos

Este projeto de pesquisa objetiva o estudo e o desenvolvimento de um Dispositivo de Controle *open-source* para aplicações da Indústria 4.0 utilizando o RAMI 4.0 e OPC UA. Esta interface deverá prover as funcionalidades de programação, identificação por rádio frequência, comunicação em rede e supervisão dos equipamentos, disponibilizando em tempo real as informações do processo, de forma padronizada e interoperável, para qualquer tipo de plataforma conectada à rede e/ou Internet.

1.3 Estrutura e Conteúdo

A estrutura deste trabalho ficou dividida em 9 capítulos considerando esta introdução e desconsiderando as referências bibliográficas.

O capítulo 2 trata de uma revisão bibliográfica que relaciona os principais tópicos acerca do tema como a Indústria 4.0, RAMI4.0 e OPC UA. Dentro dessas contextualizações são apresentadas as mais relevantes pesquisas que já foram publicadas com base no objeto de estudo proposto.

O capítulo 3 traz a primeira parte de uma revisão conceitual, abordando as principais características do RAMI4.0. Nessa revisão são explanados os principais elementos do eixo tridimensional, a composição de um Componente da Indústria 4.0 e o Shell de Administração.

O capítulo 4 apresenta a segunda parte de uma revisão conceitual, esclarecendo os elementos bases do protocolo de comunicação para a Indústria 4.0 e o RAMI 4.0, denominado OPC UA. A descrição e distribuição desses elementos ao longo deste capítulo foi fundamentada a partir da revisão de diversos trabalhos e das especificações oficiais do OPC UA.

O capítulo 5 aborda uma análise comparativa e descritiva entre as principais implementações de software *open-source* do OPC UA. Para isso, com base na apresentação do capítulo 4, foram estabelecidos alguns critérios como conjuntos de serviços, mapeamentos de tecnologias, políticas de segurança, versões dos projetos etc, para definir qual é a implementação mais viável e compatível com os requisitos do Dispositivo de Controle proposto.

O capítulo 6 destaca a Arquitetura para o desenvolvimento de um Dispositivo de Controle para Indústria 4.0, juntamente com as tecnologias necessárias para a implementação da proposta.

O capítulo 7 mostra o relacionamento entre a arquitetura, as tecnologias a serem utilizadas e as camadas selecionadas do RAMI 4.0. Este capítulo detalha o funcionamento das tecnologias dentro do sistema e os principais desenvolvimentos realizados.

O capítulo 8 levanta os resultados dos testes realizados a partir do sistema desenvolvido, sendo este submetido a um caso de estudo real de aplicação da Indústria 4.0. Além disso, neste capítulo são discutidas a possibilidade de tornar o Dispositivo de Controle compatível com um componente da Indústria 4.0.

Por fim, o capítulo 9 apresenta as conclusões e os trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

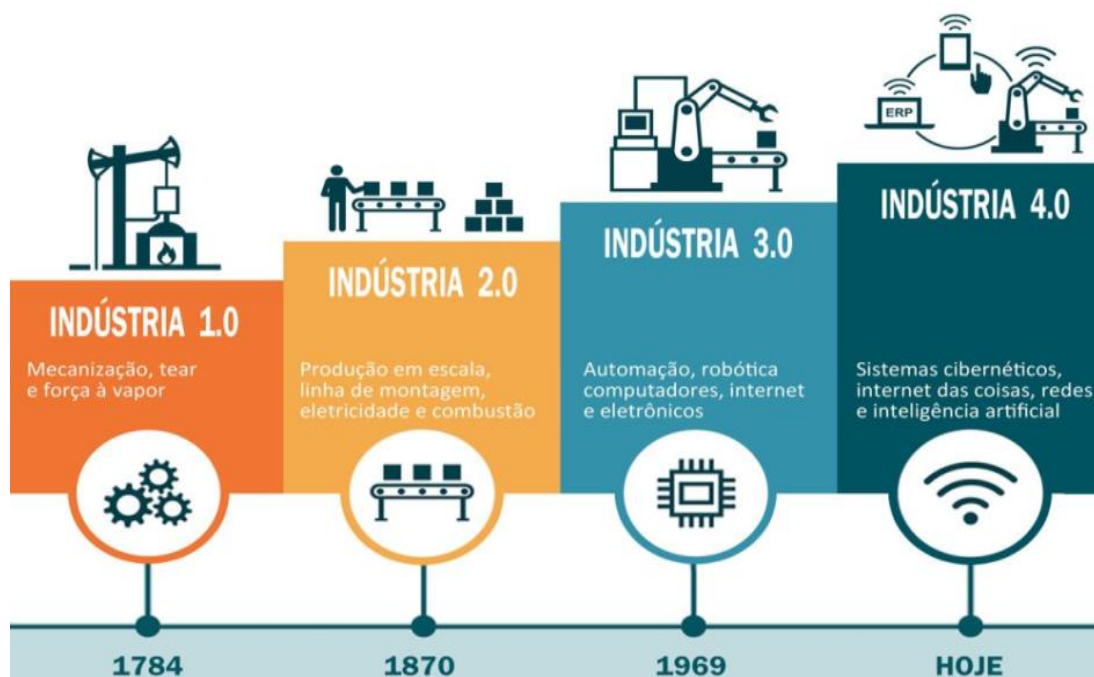
2.1 Indústria 4.0

Ao decorrer de toda evolução histórica, ocorreram grandes avanços e descobertas que mudaram a forma como a indústria se comportava, sendo que essas mudanças indubitavelmente melhoraram a produtividade e o modo de vida das pessoas, consolidando-se para isso três revoluções industriais anteriores que envolvem o uso de diferentes tecnologias.

A primeira revolução industrial foi marcada pela mobilização e mecanização de sistemas produtivos, sendo impulsionada pela construção de ferrovias e pela invenção da máquina a vapor, dando início a produção mecânica (SCHWAB, 2016; BAUERNHASL et al., 2014; TELLO, 2018). A segunda revolução inicia as linhas de montagem de produção em massa auxiliada pela descoberta e desenvolvimento da energia elétrica (SCHWAB, 2016; BAUERNHASL et al., 2014; TELLO, 2018). Já a terceira revolução, também considerada a revolução digital, recebe este nome diante o desenvolvimento de semicondutores, computação de *mainframe* (anos 60), computação pessoal (anos 70 e 80) e internet (década de 90) (SCHWAB, 2016).

Complementarmente, Tello (2018) destaca que essa revolução utiliza uma produção automatizada, com a implementação de eletrônica, PLCs (*Programmable Logic Controller* - Controlador Lógico Programável), robótica e sistemas de TI. No cenário atual, a mais recente evolução industrial, conhecida como Quarta Revolução Industrial implementa tecnologias, como Aprendizado de Máquina, *Big Data*, CPS, IoT, IoS etc que são utilizadas para personalizar e produzir de maneira mais eficiente (TELLO, 2018). A Figura 2 mostra uma linha do tempo, descrevendo as datas aproximadas de cada uma das revoluções industriais.

Figura 2 - Linha do tempo das revoluções industriais



Fonte: Adaptado de Cline (2017)

Por definição, o termo Indústria 4.0 ou Quarta Revolução Industrial representa a próxima etapa na organização e controle de todo o fluxo de valor ao longo do ciclo de vida de um produto (DORST et al., 2016). O termo Indústria 4.0 foi apresentado como estratégia do governo de desenvolver tecnologias de ponta para revolucionar a organização das cadeias globais de valor (SCHWAB, 2016). Ainda, o termo se tornou extremamente importante, especialmente na Alemanha, sendo muito popular e amplamente discutido em grupos sociais, indústria e campo acadêmico (PLESIS, 2017). Segundo Bauer (2014) a razão disso é que se estimou que os benefícios da Indústria 4.0 contribuirão com até 78 bilhões de euros para o produto interno bruto alemão até 2025. No Brasil, primeiras investigações acerca da Indústria 4.0 ou Manufatura Avançada começaram a acontecer em 2013 (A VOZ DA INDÚSTRIA, 2016).

Em se tratando da potencialidade da Indústria 4.0, Kagermann, Wahlster e Helbig (2013) destacam os seguintes pontos:

- **Atender aos requisitos individuais dos clientes:** a I4.0 permite que critérios específicos do cliente sejam incluídos nas respectivas fases: projeto, configuração, pedido, planejamento, fabricação e operação. Dessa maneira, a linha de produção

pode se modificar para atender a critérios individuais, permitindo que até mesmo, modificações de última hora sejam incorporadas ao sistema.

- **Flexibilidade:** configurações dinâmicas de diferentes aspectos dos processos de negócios, como qualidade, tempo, risco, robustez e preço, podem facilitar o corte contínuo de materiais e cadeias de suprimentos. Isso também reflete positivamente nos processos de engenharia, tornando-os mais ágeis. Além do mais, os seguintes itens podem ser modificados: processos de fabricação, compensação de escassez temporária e aumentos enormes na produção em um curto espaço de tempo.

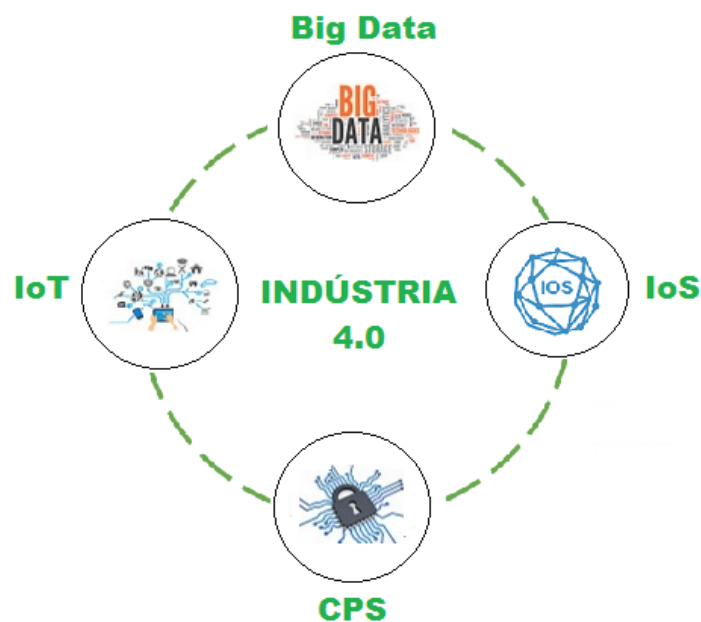
- **Tomada de decisão otimizada:** a Indústria 4.0 permite a verificação antecipada de decisões de design na esfera da engenharia e respostas mais flexíveis à interrupção e otimização global em todos os segmentos da empresa.

- **Produtividade e eficiência dos recursos:** os sistemas podem ser continuamente otimizados durante a produção em termos de consumo de recursos ou reduzindo suas respectivas emissões.

- **Criando oportunidades de valor através de novos serviços:** a I4.0 abre novas formas de criar valor através de serviços. Algoritmos inteligentes podem ser aplicados a grande quantidade de dados gravados por dispositivos inteligentes, a fim de fornecer serviços inovadores. Com isso, por exemplo, existem oportunidades para pequenas empresas e startups desenvolverem serviços B2B para a Indústria.

Para tornar possível a disponibilização desse conjunto de características citadas anteriormente, a I4.0 integra as tecnologias da terceira revolução industrial, juntamente com componentes para processamento de informações que permitem armazenar, processar e gerenciar as informações. Dentre diversas tecnologias que são contempladas para a formação da Indústria 4.0, como por exemplo, a realidade aumentada, computação em nuvem, sistemas integrados etc, algumas se destacam dentro de tal construção, como a Internet das Coisas (IoT), Big Data, Internet dos Serviços (IoS) e os Sistemas Ciber-Físicos (CPS) (TELLO, 2018), conforme ilustra a Figura 3.

Figura 3 - Principais tecnologias para a composição da Indústria 4.0



Fonte: Elaborado pelo autor

A **Internet das Coisas (IoT)** pode ser caracterizada como sendo uma rede de objetos físicos que são capazes de serem acessados através duma rede de comunicação ou a Internet (KUMAR; BOSE, 2015). Segundo Ashton (2010), nossa sociedade, economia e sobrevivência não são fundamentadas em informações ou ideias, mas sim em coisas. Essas “coisas” no que lhes concernem, contêm tecnologias embarcadas, capazes de interagir com o ambiente externo, tornando possível misturar o mundo físico ao mundo da informação (KUMAR; BOSE, 2015). Deste modo, essa junção é realizada por meio da integração de diversas tecnologias, como a Identificação por Radiofrequência (RFID), redes sem fio, Computação em Nuvem e Protocolos de Internet; resultando assim, na viabilização da transferência de informações, análise de dados e aplicações (GUBBI et al., 2013).

Os **Sistemas Ciber Físicos**, também conhecidos por **CPS** (do inglês, *Cyber-Physical Systems*), por definição, representam a integração de sistemas computacionais com processos físicos (LEE; SESHIA, 2016). Em outras palavras, um CPS integra a comunicação e recursos de armazenamentos computacionais com monitoramento e/ou controle de entidades no mundo físico, ocorrendo de maneira confiável, segura, eficiente e em tempo real (SANISLAV; MICLE, 2012). Na sua forma estrutural, um CPS é constituído por meio de uma unidade de controle (geralmente um ou mais microcontroladores) que controlam sensores e atuadores, sendo estes necessários para interagir com o mundo físico. Este conjunto de elementos também

exige uma interface de comunicação para a troca de dados com outros sistemas embarcados ou com a nuvem (JAZDI, 2014).

O termo **Internet de Serviços** ou **IoS** (do inglês, *Internet of Services*) refere-se à infraestrutura que permite a prestação de serviços universais aos consumidores (SCHROTH; JANNER, 2007). O IoS descreve uma base que usa a internet como meio de oferecer e vender serviços (CARDOSO et al., 2008). Juntos, consumidores e provedores podem negociar os serviços enquanto estão envolvidos em uma interação de negócios, o que representa uma tecnologia de suporte para a visão de IoS (SCHROTH; JANNER, 2007). Diante dessa rede de negócios, os clientes podem ser atendidos através de um sistema em que organizações trabalham juntas, sendo que valores são criados ao longo da cadeia de suprimentos entre uma empresa, seus fornecedores, seus clientes e agregadores. Exemplificando, em uma rede de serviços que inclui pesquisa, desenvolvimento, concepção, marketing e vendas, todas essas seções estão trabalhando de forma intercambiável, visando agregar valor ao serviço geral (KHAKIFIROOZ et al., 2018).

Em relação ao **Big Data**, Witkowski (2017), destaca que referente ao contexto da Internet das Coisas e da Indústria 4.0, enormes quantidades de informações são produzidas e coletadas diariamente. Dessa forma, o processamento e análise dessas informações estão além da capacidade de ferramentas tradicionais. Para resolver essa questão, o autor destaca que existe o conceito conhecido como Big Data, sendo este capaz de permitir o gerenciamento e uso de forma rápida e eficiente de um banco de dados em constante crescimento, permitindo também a análise e separação do que é menos e mais importante.

2.2 Contextualização do RAMI 4.0

Grandes partes das atividades humanas, atualmente, estão inter-relacionadas através de uma infinidade de sistemas de comunicação ligados à Internet. Dentre tais sistemas, destacam-se a Internet de Serviços, Internet de Pessoas e Internet das Coisas. Essa última contém duas importantes derivações, sendo elas: comercial e industrial (MARCON et al., 2017). Essas tendências tecnológicas apresentadas foram amplamente desenvolvidas individualmente, porém integradas no contexto da Indústria 4.0. Tal integração foi estruturada para ser baseada em padrões, abordagens e métodos já existentes, sendo representada em um modelo tridimensional em que

cada eixo representa um aspecto no contexto da Indústria 4.0 (CONTRERAS; GARCIA; PASTRANA, 2017).

Além do mais, diante dessa integração, será possível que entidades ativas dentro da Indústria 4.0 busquem comunicação mútua ao longo de todo o ciclo de vida, independente das fronteiras entre empresas, nações e estados (MARCON et al., 2017). Com isso, todas as entidades dentro da cadeia de produção serão capazes de possuir todos os dados necessários. Vale ainda ressaltar que os participantes na cadeia de produção e comercial, incluindo fabricantes de máquinas industriais e desenvolvedores de softwares, terão a oportunidade de desenvolver seus produtos com base no conhecimento prévio dos componentes mais recentes, ainda não desenvolvidos e testados pelos produtores.

Mediante aos efeitos de cadeias tão abrangentes, as principais empresas e instituições alemãs, incluindo BITCOM, VDI/VDE e ZVEI (MARCON et al., 2017) elaboraram e lançaram em 2015 o RAMI 4.0, sendo este uma estrutura tridimensional (GERMAN STANDARDIZATION ROADMAP, 2016). Este modelo de referência permite a migração passo a passo do mundo de hoje para o da I4.0 e a definição de domínios de aplicação com estipulações e requisitos especiais (HOFFMESITER, 2015).

Para Schulte e Colombo (2017) o RAMI 4.0 tem sido introduzido pela organização alemã de normatização e normalização industrial, DIN¹, com o objetivo de suportar todos os grupos participantes, membros da cadeia de valor do negócio industrial, como fornecedores de controladores, construtores de máquinas, integradores de sistemas, fornecedores e usuários de produtos e serviços. Os autores ainda destacam que o RAMI propõe o uso de tecnologias de controle e comunicação (ICT) no estado da arte para suportar a reconfigurabilidade e evolutibilidade estruturais para sistemas industriais. Complementarmente, Pisching (2017) afirma que o RAMI 4.0 foi elaborado a partir de normas vigentes do setor produtivo visando concentrar diferentes especificações em um único modelo tridimensional que atenda à integração vertical, horizontal e a engenharia fim-a-fim.

A adoção do RAMI 4.0 como padrão de projetos e aplicações na I4.0 fornece alguns benefícios como o uso de uma Arquitetura Orientada a Serviços (SOA), a

¹ DIN (Deutsches Institut für Normung – Instituto Alemão de Normalização) é uma organização nacional alemã para a padronização, com sede em Berlim. O DIN realiza as mesmas funções que órgãos internacionais como o ISO.

combinação de componentes de TI em cada camada e a divisão dos processos em componentes, facilitando comunicação e processamento.

Diante disso, trabalhos recentes têm focado no desenvolvimento de soluções que integrem diferentes tecnologias para implementar as funcionalidades requeridas pelas aplicações da Indústria 4.0, baseando-se no RAMI 4.0. Contrenas, Garcia e Pastrana (2017) abordam as características essenciais que permitem que um sistema de fabricação seja adaptado para se tornar um aplicativo da I4.0, seguindo o modelo RAMI 4.0. Para isso, os autores especificam um sistema de fabricação inteligente baseado em padrões como FDI (*Field Device Integration*), *AutomationML*² e OPC UA (*Open Platform Communications Unified Architecture*). Schleipena et al. (2016) retratam diferentes cenários baseados no OPC UA como tecnologia de comunicação para indicar o papel desta tecnologia como habilitadora para a produção flexível, adaptativa e transparente, seguindo os requisitos da I4.0 com base no RAMI 4.0. Shulte e Colombo (2017) abordam os principais passos para migração da infraestrutura de ICT associada ao gerenciamento de controle de uma linha de extrusora de chapas industriais para um sistema industrial digitalizado em conformidade com os requisitos do RAMI 4.0. Em outro estudo, Wang, Towara e Anderl (2017) criam um modelo universal orientado a aplicações para promover o uso do RAMI 4.0 na prática e para apoiar novas pesquisas em torno da I4.0. Langmann e Rojas-Peña (2016) retratam a utilização de um PLC como componente da I4.0, seguindo as especificações definidas pelo RAMI 4.0, demonstrando que os controladores industriais podem ser facilmente integrados nos ambientes de produção da I4.0 usando o paradigma de serviço. Pisching et al. (2017) apresentam uma arquitetura baseada em camadas no RAMI 4.0 para descobrir equipamentos para processar operações de acordo com os requisitos do produto. Fleischmann et al. (2017) desenvolvem e validam um conceito para realização de uma IHM (Interface Homem Máquina) configurável e adaptável para monitorar plantas de produção discretas, baseando-se nas tecnologias OPC UA e *AutomationML*, aplicadas dentro da estrutura do RAMI 4.0.

² *AutomationML* é um formato de dados neutro baseado em XML para armazenamento e troca de informações de engenharia de fábrica, que é fornecido como padrão aberto. O objetivo do *AutomationML* é interconectar o cenário de ferramentas heterogêneas e de engenharia moderna em seus diferentes segmentos, como engenharia mecânica, projetos elétricos, desenvolvimento de IHM, CLP etc.

2.3 Comunicação para Indústria 4.0 e RAMI 4.0

A Indústria 4.0 é impulsionada por tecnologias avançadas de informação e comunicação (ZEZULKA et al., 2016; PEREIRA et al., 2017; BLAZEK et al., 2016). O estudo de tecnologias de comunicação entre equipamentos e sistemas compatíveis com a I4.0 tem sido alvo de discussão (WOLLSCHLAEGGER; SAUTER; JASPERNEITE, 2017). Isso ocorre, devido ao fato da I4.0 precisar de uma linguagem comum, confiável, estável e principalmente, poderosa, para poder impulsionar a revolução, especialmente se ela se espalhar pelo mundo através de tecnologias de nuvem. Além disso, a comunicação dentro do contexto da I4.0 também requer interconexões flexíveis entre camadas de integração e camadas de informação (DERHAMY et al., 2017). Complementando, Sauer (2014) discute que em sistemas industriais, os fluxos de informações crescem em todos os segmentos da planta e a partir disso, a imprescindibilidade está voltada para a organização dos dados entre a planta vertical, horizontal e o ciclo de vida útil dos equipamentos de produção.

Em virtude de tais requisições, para saná-las, tem-se o protocolo OPC UA, sendo este extremamente conveniente para a infraestrutura de informação e comunicação da I4.0, pois ele permite comunicações padronizadas, rápidas e seguras (ZEZULKA et al, 2018). Aliás, Hoppe (2017) deixa evidente que não há nenhuma possibilidade de I4.0 sem o OPC UA, destacando que em abril de 2015, o RAMI 4.0 recomendou apenas o OPC UA para implementação das funcionalidades da camada de comunicação em relação às outras tecnologias existentes para a I4.0 e IIoT. Sendo assim, Zezulka et al. (2018) destacam que o OPC UA será aceito como um protocolo de comunicação comum para a revolução da I4.0 nos países industriais mais desenvolvidos, num futuro próximo.

Diante deste quadro, é imprescindível um esforço considerável para implementar com sucesso a visão da Indústria 4.0, uma vez que as demandas variam consideravelmente. Visando a redução do grau de complexidade, é preciso haver modularização, padronização e digitalização consistentes (BURKE, 2017). Em tal caso, a I4.0 estabelece uma série de requisitos em termos de comunicações e o OPC UA apresenta soluções para tais exigências, conforme demonstrado através da Tabela 1.

Tabela 1 - Requisitos da Indústria 4.0 e soluções apresentadas pelo OPC UA

Requisitos da Indústria 4.0	Soluções através do OPC UA
Escalabilidade para redes integradas incluindo os menores sensores, dispositivos embarcados e controladores PLC, PCs, <i>smartphones</i> , <i>mainframes</i> e aplicativos em nuvem. Comunicação horizontal e vertical em todas as camadas.	O OPC UA pode ser implementado através de <i>hardware single</i> e <i>multi-core</i> com uma ampla gama de arquitetura de CPU (Intel, ARM, PPC etc). O OPC UA é usado em dispositivos de campo embarcados como leitores RFID, conversores de protocolo etc e em praticamente todos os controladores e produtos SCADA/IHM, bem como sistemas MES/ERP. Os projetos já foram realizados com sucesso na <i>Amazon</i> e no <i>Microsoft Azure Cloud</i> .
Independência da tecnologia de comunicação do fabricante, setor, sistema operacional e linguagem de programação.	A <i>OPC Foundation</i> é uma organização independente sem fins lucrativos. A associação não é necessária para usar a tecnologia OPC UA ou para desenvolver produtos OPC UA. O OPC Clássico é amplamente utilizado em automação, mas é tecnologicamente neutro em termos setoriais. Já o OPC UA é executado em todos os sistemas operacionais e até mesmo em implementações de camada de chip sem um sistema operacional. O OPC UA pode ser implementado em todas as linguagens de programação. Atualmente pilhas em ANSI C/C++, .NET e Java estão disponíveis.
Transferência segura e autenticação no nível do usuário e do aplicativo.	O OPC UA usa certificados X.509, Kerberos ou usuário/senha para autenticação do aplicativos. A transferência assinada e criptografada, bem como um conceito de direitos no nível de ponto de dados, com funcionalidades de auditoria está disponível na pilha.
SOA, transporte através de padrões estabelecidos como TCP/IP para troca de dados ao vivo históricos, comandos e eventos (evento /call-back).	O OPC UA é independente do método de transporte. Atualmente, duas ligações de protocolo estão disponíveis: protocolo binário baseado em TCP otimizado para aplicativos de alto desempenho e serviço da Web HTTP/HTTPS com mensagens binárias ou XML codificadas. Além disso, o modelo de comunicação Publicar/Assinar pode ser integrado. As pilhas garantem o transporte consistente de todos os dados. Além dos dados ao vivo e em tempo real, os dados históricos e sua agregação matemática são padronizados no OPC UA. Ademais, são possíveis mais chamadas de método com argumentos complexos.

Mapeamento de conteúdo de informações com qualquer grau de complexidade para modelagem de objetos virtuais para representar os produtos reais e suas etapas de produção.	O OPC UA fornece um conceito totalmente em rede para um espaço de endereço orientado a objeto (não apenas rede hierárquica, mas com malha completa), incluindo descrição de Metadados e de objetos. Estruturas de objetos podem ser geradas via referência das instâncias entre si e seus tipos e um modelo de tipo que pode ser estendido através de herança. Como os servidores carregam suas instâncias e sistemas de tipos, os clientes podem navegar por essa rede e obter todas as informações que precisam, mesmo para tipos antes desconhecidos.
Integração em engenharia e extensão semântica.	A <i>OPC Foundation</i> já colabora com sucesso em parceria com outras organização (PLCopen, BACnet, FDI, AIM etc) e atualmente está expandindo suas atividades de cooperação, por exemplo, MES, DACH, ISA95, MDIS. Uma nova iniciativa de cooperação é com a <i>AutomationML</i> com o objetivo de otimizar a interoperabilidade entre as ferramentas de engenharia.
Verificação de conformidade com o padrão definido	O OPC UA já é um padrão e estão disponíveis ferramentas e laboratórios de teste para testar e certificar a conformidade. Eventos de teste adicionais aumentam a qualidade e garantem a compatibilidade. Testes expandidos são necessários para extensões/emendas (padrões complementares, semântica).

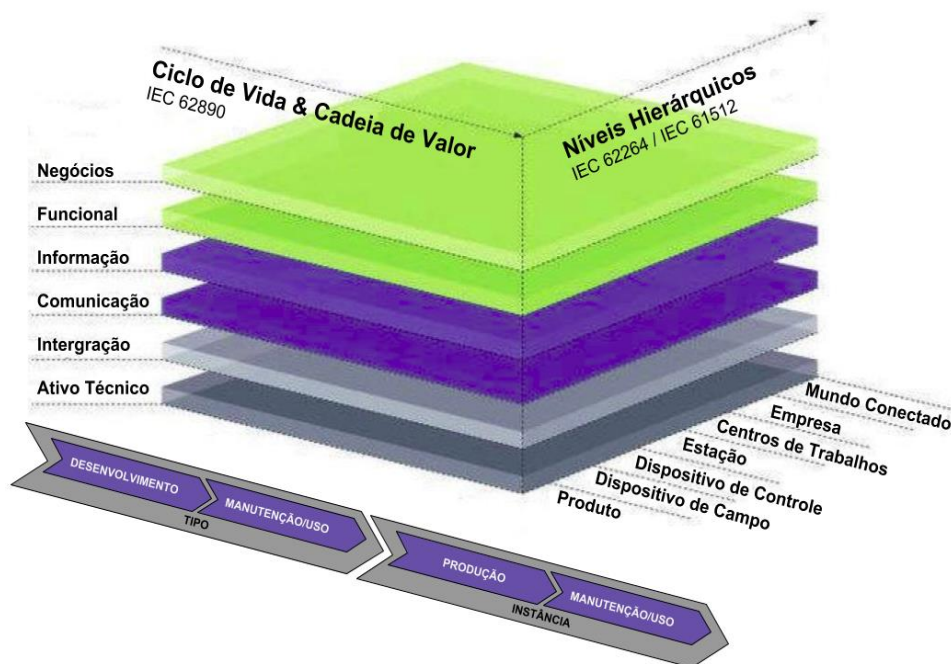
Fonte: Burke, 2017.

Neste contexto, recentes estudos demonstram as possibilidades de implementações da comunicação usando o protocolo OPC UA para aplicações da Indústria 4.0. Ghazivakili, Demartini e Zuzino (2018) detalham a arquitetura para a criação de um coletor de dados industrial, habilitando o OPC UA para a Indústria 4.0. Este coletor é capaz de adquirir os dados do chão de fábrica e convertê-los no padrão OPC UA, objetivando uma integração vertical ágil e totalmente eficaz. Panda et al. (2018) propõem uma abordagem para integrar novas máquinas e dispositivos em um sistema de produção existente sem qualquer intervenção manual, fornecendo assim, uma arquitetura de sistema *Plug & Produce*. Imtiaz e Jaspertine (2014) investigam a utilização do OPC UA como uma solução de *middleware* para dispositivos com recursos limitado, propondo para isso, um estudo de caso que contempla o cenário para a IIoT e I.40.

3 RAMI 4.0 - MODELO DE REFERÊNCIA DE ARQUITETURA PARA A INDÚSTRIA 4.0

No que diz respeito aos eixos estruturais do RAMI 4.0, o modelo de referência pode ser dividido, conforme mostrado na Figura 4, nos seguintes eixos: Níveis Hierárquicos (*Hierarchy Levels*), Ciclo de Vida Cadeia de Valor (*Life Cycle & Value Stream*) e Camadas (*Layers*).

Figura 4 - Estrutura Tridimensional do RAMI 4.0



Fonte: Adaptado de Adolphs (2015).

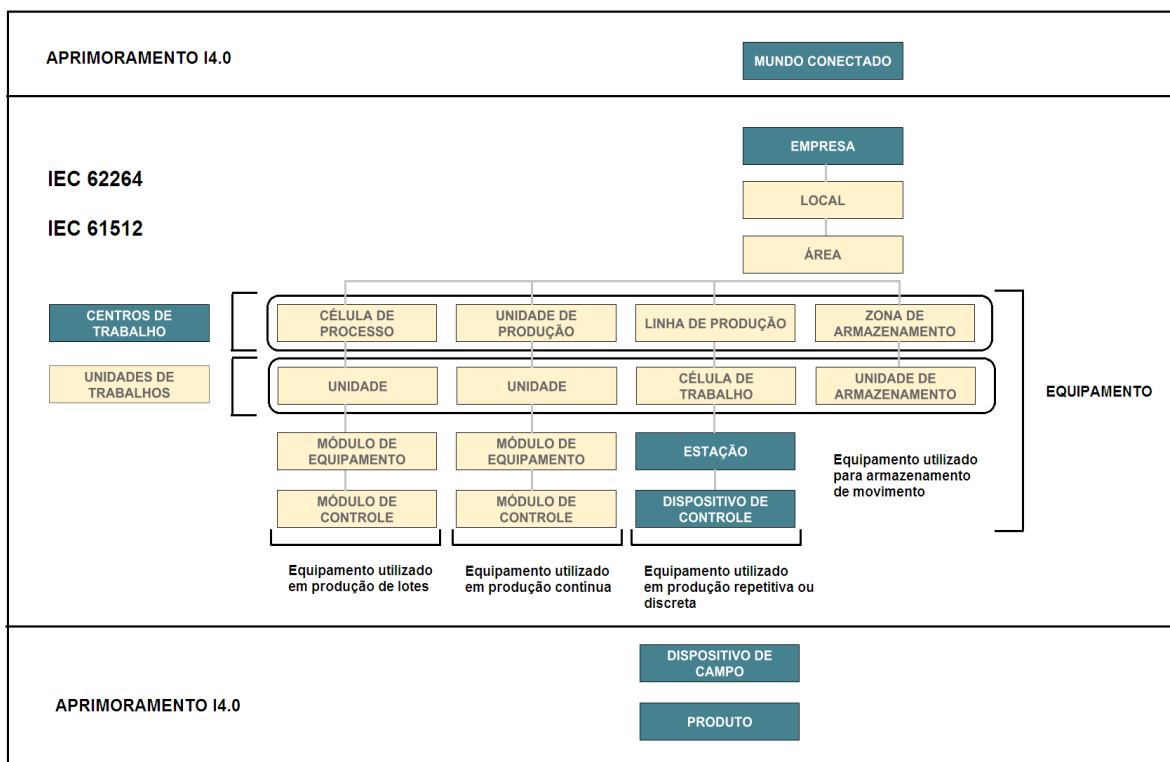
3.1 Eixo de Níveis de Hierarquia

O primeiro eixo ou também conhecido como eixo dos níveis hierárquicos, define o modelo de interconexão de todos os elementos do sistema produtivo, incluindo informações, pessoas e máquinas (STEVAN; LEME; SANTOS, 2018). As funcionalidades, representadas nos níveis de hierarquia foram expandidas para representar o ambiente da Indústria 4.0, incluindo peças de trabalhos, rotuladas como “Produto” e a conexão com a Internet das Coisas e Serviços, rotulada como “Mundo Conectado” (GOTZE, 2016).

A estrutura dos Níveis Hierárquicos, localizada à direita no eixo horizontal da Figura 4, foi constituída com base nas normas IEC 62264 e 61512. Essas normas contemplam os elementos referenciados na Figura 5 (WANG; TOWARA; ANDERL,

2017; HANKEL; REXROTH, 2015) e padronizam sistemas corporativos de TI e controle, em que os níveis hierárquicos representam diferentes funcionalidades dentro das fábricas (HANKE; REXROTH, 2015). Os níveis hierárquicos podem ser subdivididos em sete categorias: Produto (*Product*), Dispositivo de campo (*Field Device*), Dispositivo de Controle (*Control Device*), Estação (*Station*), Centros de Trabalho (*Work Centers*), Empresas (*Enterprise*) e Mundo Conectado (*Connected World*) (WANG; TOWARA; ANDERL, 2017). A seguir, há uma explanação sobre cada um destes elementos.

Figura 5 - Derivação dos Níveis de Hierarquia do RAMI 4.0



Fonte: Adaptado de Adolphs (2015)

- **Produto:** permite uma visão homogênea do produto que será fabricado, a facilidade de produção e as interdependências entre eles (ADOLPHS et al., 2016).

- **Dispositivo de Campo:** representa o nível funcional de um dispositivo inteligente, por exemplo, um sensor inteligente (ADOLPHS et al., 2016). Basicamente, são dispositivos eletrônicos usados para detectar e identificar componentes e tecnologias de sensores (ZAHEER, 2017).

- **Dispositivo de Controle:** considerado como o cérebro da fabricação. Geralmente são máquinas/sensores usados para gerenciar comandos de entrada/saída, como por exemplo, Controlador Lógico Programável, sistemas de controle distribuído, Interface Gráfica com Usuário etc (ZAHEER, 2017).

- **Estação:** onde os operadores executam atividades administrativas para examinar a operação de eventos e processos, como por exemplo, a utilização do SCADA (ZAHEER, 2017).

- **Centros de Trabalho:** mantém informações de fabricação, definem o estado de produção e supervisionam a renovação de matérias primas para produtos refinados (ZAHEER, 2017).

- **Empresas:** geralmente são definidas em termos de ERP, sendo este um software de gerenciamento de negócios, utilizados em processos empresariais, como, por exemplo, planejamento de produção, prestação de serviços, marketing, vendas, módulos financeiros, varejo e outras despesas (ZAHEER, 2017).

- **Mundo Conectado:** este item foi adicionado na extremidade superior dos níveis de hierarquia para atender às necessidades da Indústria 4.0 que descreve o grupo de fábricas e a colaboração com firmas de engenharia externas, fornecedores de componentes e clientes (ADOLPHS et al., 2016).

3.2 Eixo do Ciclo de Vida e Cadeia de Valores

Devido a I4.0 oferecer um grande potencial de melhoria ao longo do ciclo de vida dos produtos, máquinas e fábricas, Dorst et al. (2016) apontam que para visualizar e padronizar relacionamentos, o eixo localizado à esquerda da Figura 4, refere-se ao Ciclo de Vida e Cadeia de Valores. Este eixo foi baseado na norma IEC 62890 que cuida da gestão do ciclo de vida para sistemas e produtos utilizados em processos industriais, medições, controle e automação (GOTZE, 2016; CHUANG, 2015). Neste eixo, são estruturados os tipos e instâncias.

Segundo Adolphs et al. (2015), um **tipo** é sempre criado com a ideia inicial. Isso abrange a colocação de ordens de projeto, desenvolvimento e testes que vão desde a primeira amostra até a produção de protótipos. O tipo de produto, máquina etc, pode ser criado durante essa fase. Os autores destacam também que os

produtos são fabricados industrialmente com base no tipo geral, sendo que cada produto manufaturado representa uma **instância** desse tipo, podendo essa possuir, por exemplo, um número de série exclusivo.

3.3 Eixo das Camadas

Em relação às Camadas, elas servem para descrever o mapeamento virtual de uma máquina ou processo. Esses tipos de representações são originárias da Tecnologia da Informação e Comunicação (TIC), em que as propriedades de sistemas complexos na maioria dos casos são divididas em camadas (HANKEL; REXROTH, 2015). Dentro da estrutura do RAMI 4.0, têm-se seis camadas, sendo elas: Regra de Negócios (*Business*), Funcional (*Functional*), Informação (*Information*), Comunicação (*Communication*), Integração (*Integration*) e Ativo Técnico (*Asset*) que pode ser ilustrada conforme a Figura. Para Wang, Towara e Anderl (2017) as camadas do RAMI 4.0 podem ser descritas da seguinte maneira:

- **Regra de Negócios:** cobre os modelos de negócios abstratos e a lógica do processo, além de garantir a integridade das funções ao longo da cadeia de valor;

- **Funcional:** representa o ambiente de tempo de execução para serviços e aplicativos, sendo também responsável pela integração horizontal de várias funções, geração de regras e lógicas de aplicação.

- **Informação:** essa camada é incumbida de realizar o pré-processamento dos eventos que são recebidos da camada de comunicação. Para isso, os dados são verificados quanto à integridade, resumidos em dados novos e de maior qualidade e disponibilizados para as camadas superordenadas, como a funcional.

- **Comunicação:** permite a comunicação entre os diferentes elementos da rede com base em protocolos de comunicação uniformes. Essa camada fornece serviços para controlar a camada de integração.

- **Integração:** fornecimento de informações relacionadas aos ativos técnicos, como hardware e software para as camadas sobrepostas. Os autores ressaltam que essa camada contém todos os elementos associados à TI, incluindo IHM gerando eventos com base nas informações adquiridas e executando o controle final dos processos. Adolphs et al. (2015) afirmam que a camada de

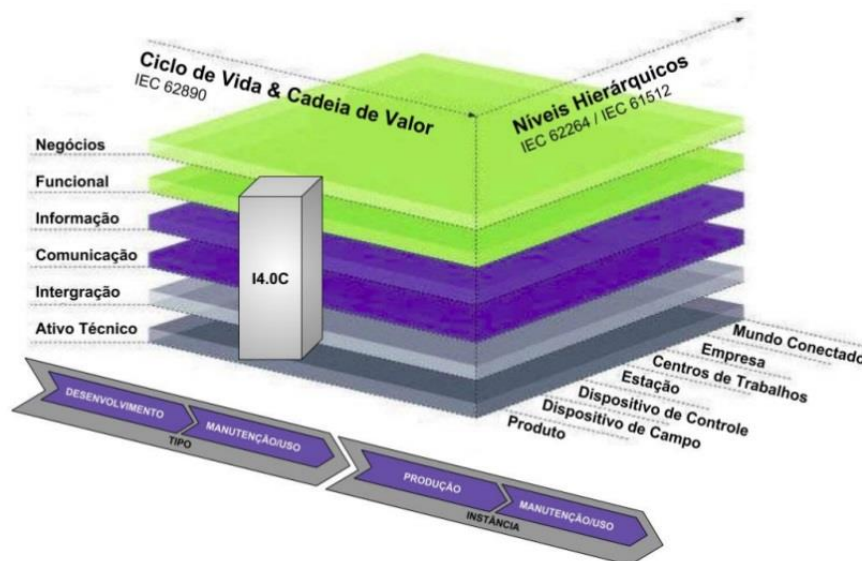
integração contém elementos associados à identificação de matérias e produtos, tais como leitores RFID e código de barras.

- **Ativo Técnico:** é a camada mais baixa dentro da estrutura do RAMI 4.0. Representa a realidade física, contendo objetos como processos, máquinas, sensores, atuadores e documentação. Nessa camada realiza-se a identificação de produtos e peças em relação aos requisitos de processamento por máquinas e estações de trabalho.

3.4 Componente da Indústria 4.0

Um outro importante elemento inserido no RAMI 4.0 é o Componente da Indústria 4.0 (MARSEU et al., 2017). Ele é caracterizado como um modelo que descreve com detalhes as propriedades que um CPS deve conhecer em termos de Indústria 4.0 (BANGEMANN et al., 2016). A Figura 6 demonstra o I4.0C inserido na estrutura do RAMI 4.0.

Figura 6 - Componente da Indústria 4.0 inserido no RAMI 4.0

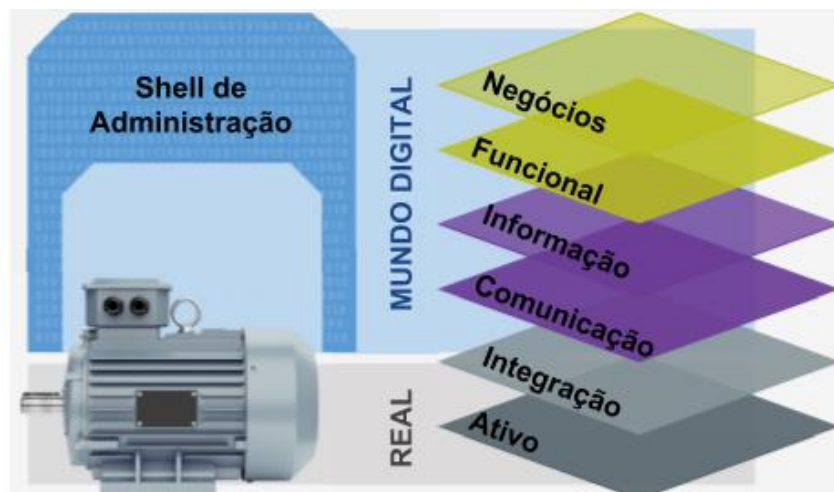


Fonte: Elaborado pelo autor

O I4.0C complementa o trabalho da *Plattform Industrie 4.0* na conexão de produtos, equipamentos e processos, corrigindo assim, a lacuna entre padrões e tecnologias no nível de produção para a I4.0 (MOSH, 2017). O Componente da I4.0 também pode ser definido como: “Participante globalmente identificável com capacidade de comunicação que consiste em um AS (*Administration Shell* – Shell de Administração) e ativo dentro de um sistema da I4.0 que oferece serviços com

características de QoS (*Quality of Service* – Qualidade do Serviço) definidas (BANGEMANN et al., 2016). Em termos estruturais, um I4.0C consiste em duas partes únicas: objetos (físicos ou lógicos) e uma representação virtual (AS - Shell de Administração). A Figura 7 mostra o relacionamento entre o componente da I4.0 e as camadas RAMI 4.0.

Figura 7 - Modelo de um I4.0C (Objetos x AS x RAMI 4.0)



Fonte: Elaborado pelo autor

Os objetos são referidos como elementos usados para se referir a cada parte física ou não física (ADOLPHS et.al, 2015). Esses objetos podem ser definidos como máquinas inteiras, componentes de automação ou uma plataforma de software. Para se tornar compatível com um I4.0C, esses objetos precisam ser combinados com uma representação virtual, definida como AS. Com isso, todos os dados relevantes de hardware ou software podem ser expandidos conforme desejado. Assim, Hoffmeister (2015) afirma que fornecedores e integradores de sistemas podem implementar serviços inteligentes criando novos tipos de informação, modelos de informação e funções técnicas. Portanto, informações relevantes para I4.0C, como manuais, parâmetros de configuração e informações históricas, podem ser fornecidas a muitos usuários em uma rede de informações, tornando a fábrica inteligente uma realidade.

3.5 Shell de Administração

Cada I4.0C possui um Shell de Administração que se destina a unir todos os dados gerados no ciclo de vida (MARSEU et al., 2017) apresentados na estrutura do RAMI 4.0. É importante destacar que esse conjunto de dados podem ser

armazenados no próprio I4.0C ou em um sistema de TI localizado em outro local, mas conectado à rede destinada para I4.0 (VDI, 2015). Além disso, Adolphs et al (2016) destacam que o Shell de Administração é descrito como um elemento chave, que transforma um objeto ou ativo em um I4.0C, permitindo o armazenamento de propriedades e estados no mundo virtual e a capacidade de compartilhar esses dados com outros componentes. É importante ressaltar que o ativo técnico ou “coisa” pode ser desacoplado do seu respectivo Shell de Administração. Um exemplo claro disso é de uma peça com uma tag RFID (ativo) que deve ter seu Shell de Administração hospedado em um servidor de sistema de TIC, estando este presente nas camadas de Negócio, Funcional e Informação. O Shell de Administração basicamente é formado por duas estruturas: Manifesto e Gerenciador de Recursos.

O **Manifesto** é definido como um diretório do conteúdo de dados individuais da representação virtual (ADOLPHS et al., 2016). Em outras palavras, ele pode ser considerado como uma interface de acesso a todas as informações importantes do componente, tanto da parte física, como da parte virtual/digital (STEVAN; LEME; SANTOS, 2018). Para Adolphs et al. (2016), as principais características contempladas dentro do manifesto são:

- ✓ Recursos característicos dos componentes reais;
- ✓ Informações sobre relacionamentos entre recursos;
- ✓ Relacionamentos entre componentes da I4.0 que são relevantes para a produção e processos;
- ✓ Descrição formal das funções relevantes de máquinas e seus fluxos de trabalho.

Visualizando de uma forma prática, Stevan, Leme e Santos (2018) destacam que o Manifesto atua como uma tabela de conteúdo, sendo esta claramente localizável para todos os dados, bem como funções contidas no Shell de Administração.

O **Gerenciador de Recursos (Body – Corpo)** ou também, Gerenciador de Componentes é o caminho de comunicação do I4.0C para serviços técnicos de TIC, permitindo também acesso externo à representação virtual e às funcionalidades técnicas (PEDONE; MEZGÁR, 2018). Esse Gerenciador de Recursos lida com pedidos de ações relacionadas aos recursos e responde a eles, constituindo dessa maneira um serviço expandido que diretamente ou indiretamente é utilizado para

realizar a manutenção das informações incluídas ao longo da vida do componente (STEVAN; LEME; SANTOS, 2018). A Figura 8 ilustra a estrutura de um Shell de Administração, expandindo a visualização para o Manifesto e o Gerenciador de Recursos.

Figura 8 - Estrutura de Informação de um Shell de Administração



Fonte: Elaborado pelo autor

4 OPC UA - PADRÃO DE COMUNICAÇÃO PARA INDÚSTRIA 4.0

A popularidade do OPC UA não se restringe pelo fato do protocolo atender aos requisitos citados na revisão bibliográfica, mas também por facilitar a IIoT através de sua escalável e segura arquitetura (ERUVANKAI; MUTHUKRISMAN; MYSORE, 2017). Essas características se deram pelo fato do OPC UA ter conseguido unificar todas as especificações do seu antecessor, o OPC Clássico, e as levado para o estado da arte usando SOA (LEHNHOFF; ROHJANS; MAHNKE, 2011). Os autores ainda destacam que o OPC Clássico, é bem aceito no cenário de automação industrial, pois até o ano de publicação de seu artigo, 2011, existiam mais de 22.000 produtos OPC Clássico disponíveis no mercado, distribuídos em mais de 3.500 empresas, incluindo todos os principais fornecedores de soluções para automação. Em vista desses fatores, faz-se necessária uma abordagem sobre os principais tipos de tecnologias OPC e o que levou a OPC Foundation a criar as novas especificações, tendo por consequência a elaboração de um padrão de arquitetura unificada, totalmente compatível com a Indústria 4.0 e RAMI 4.0.

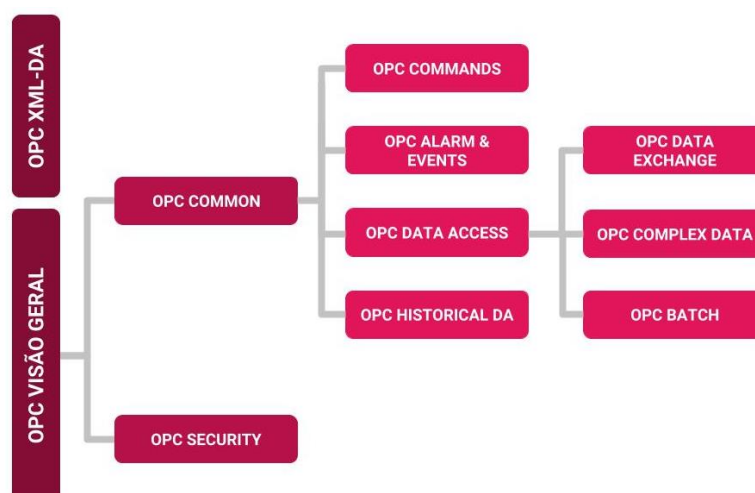
4.1 Visão Geral do OPC Clássico

A *OPC Foundation*, responsável pelo desenvolvimento e manutenção do OPC, define este, como sendo um padrão de interoperabilidade para a troca segura e confiável de dados no espaço de automação industrial. Ao longo dos anos, o acrônimo OPC sofreu algumas variações. Inicialmente, a sigla OPC significava “*OLE for Process Control*”, ou “*Object Linking and Embedding for Process Control*”, porém devido ao seu estado atual, o acrograma OPC corresponde à “*Open Platform Communications*” (OPC FOUNDATION, 2017a).

Segundo Mahnk, Leitner e Damm (2009) as interfaces OPC são baseadas nas tecnologias COM e DCOM da Microsoft. Na época em que o OPC foi desenvolvido, este embasamento trouxe a redução do trabalho de especificação para definição de diferentes APIs, sem o requisito de definir o protocolo de rede ou mecanismo para intercomunicação. Usando essas tecnologias disponíveis em todos os PCs baseados nos sistemas operacionais *Windows*, a redução do trabalho proporcionou o avanço rápido do desenvolvimento das especificações, sendo que este coeficiente acarretou na diminuição do tempo de colocação dos produtos OPC disponíveis para o mercado.

A estrutura do OPC Clássico originou-se por volta dos anos noventa do século passado, quando a empresa *OPC Foundation* lançou sua primeira especificação para acesso de dados (*OPC DA – Data Access*) que promoveu uma série de métodos entre servidores e clientes, incluindo algumas funcionalidades, como, por exemplo, leitura, escrita e monitoramento de variáveis (HUIMING; ZHIFENG, 2010). Posteriormente, a OPC desenvolveu outras especificações. Entre elas estão o *Historical Data Access (HDA)* e *Alarm & Event (A&E)*. Essas especificações se popularizaram no âmbito industrial, pois resultaram na interoperabilidade entre componentes de automação, como controladores, IHM etc (LEHNHOFF; ROHJANS; MAHNKE, 2011). A partir delas, existem extensões, como a título de exemplo: *OPC Complex Data*, *OPC Batch*, *OPC Data Exchange* e as primeiras especificações para serviços baseados na web, neste caso, *OPC-XML-DA*. A Figura 9 fornece uma visão geral das especificações do OPC Clássico.

Figura 9 - Visão geral das especificações do OPC Clássico



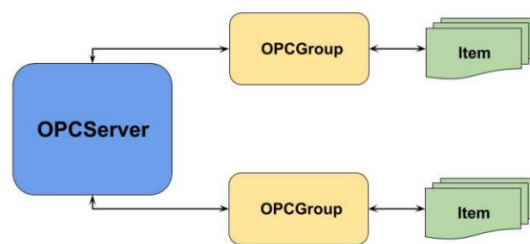
Fonte: Elaborado pelo autor

4.1.1 OPC DA (Data Access)

Além da criação dos métodos citados anteriormente, o OPC DA foi designado para padronizar o mecanismo de comunicação de inúmeras fontes de dados, como, dispositivos de entradas e saídas de hardware no chão da planta ou banco de dados em salas de controle (SON; YI, 2010). Para a efetivação do OPC DA, os clientes devem selecionar explicitamente as variáveis, também chamadas de itens OPC as quais eles desejam ler, escrever ou monitorar no servidor. Tendo em vista essas

execuções, inicialmente, o cliente precisa estabelecer uma conexão com um servidor, criando o objeto *OPCServer*, que é o objeto de nível superior da hierarquia. Posteriormente, para acessar os dados, o cliente agrupa os itens OPC com configurações idênticas, como o tempo de atualização em um objeto *OPCGroup* (SON; YI, 2010; BANGEMANN et al., 2014). Segundo Son e Yi (2010) na aplicação real, vários valores são lidos e escritos ao mesmo tempo. Devido a isto, é mais eficiente gerenciar vários objetos *OPCItem* com uma única chamada via métodos fornecidos pelo *OPCGroup*. A Figura 10 mostra os diferentes objetos que o cliente OPC cria no servidor.

Figura 10 - Objetos criados por um cliente para acessar dados de um servidor



Fonte: MAHNKE, LEITNER & DAMM, 2009

Segundo Mahnke, Leitner e Damm (2009) quando os itens são adicionados a um grupo, eles podem ser lidos ou escritos pelo cliente. Porém, a maneira preferida para a leitura cíclica dos dados pelo cliente é através de mudanças de valores no servidor. O cliente define uma taxa de atualização no grupo que contém os itens de interesse, sendo que ela é usada no servidor para verificar ciclicamente os valores de mudanças. Ao término de cada ciclo, o servidor envia apenas os valores alterados ao cliente. Em suma, o OPC DA é a mais importante especificação do OPC Clássico. Ele é utilizado em 99% dos produtos que usam as tecnologias OPC.

4.1.2 OPC HDA (Historical Data Access)

Enquanto o OPC DA fornece acesso aos dados em tempo real, com estes, mudando continuamente, o OPC HDA disponibiliza acesso aos dados já armazenados (MAHNKE; LEITNER; DAMM, 2009). Os autores ainda destacam que os servidores OPC HDA permitem que os clientes acessem dois tipos básicos de dados classificados como brutos (dados históricos armazenados) ou agregados (extraídos dos dados brutos). Para Son e Yi (2010) dependendo da implementação, os

servidores OPC HDA são divididos em simples (estabelecem algumas interfaces opcionais e acesso aos dados brutos) ou complexos (capazes de realizar processamento, como também a análise dos dados). Em se tratando dos modos de funcionamento do OPC HDA, o cliente durante a conexão com o servidor, cria um objeto denominado *OPCHDAServer*, sendo que este oferece todas as interfaces e métodos para a leitura e atualização de dados históricos. Também é definido um segundo objeto, classificado por *OPCHDABrowser*. Ele é responsável pela navegação no espaço de endereços do servidor HDA (MAHNKE; LEITNER; DAMM, 2009).

Mahnke, Leitner e Damm (2009) apontam que a principal funcionalidade do OPC HDA é a leitura de dados históricos, podendo esta, ocorrer de três formas diferentes. A primeira forma está concatenada com a leitura dos dados brutos do arquivo, em que o cliente é capaz de definir uma ou mais variáveis e o domínio do tempo que ele deseja ler. Com isso, o servidor retorna todos os valores arquivados para o intervalo de tempo especificado. Os autores apontam de forma sucinta que a segunda maneira lê valores de uma ou mais variáveis para *timestamps*³ especificados. Por fim, a terceira maneira calcula valores agregados de dados no histórico do banco de dados para o domínio do tempo especificado para uma ou mais variáveis.

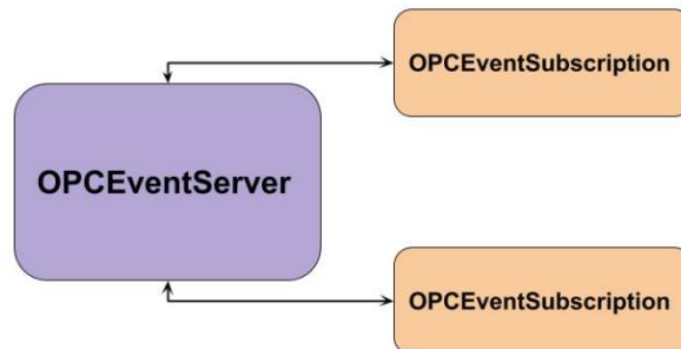
4.1.3 OPC A&E (Alarm & Events)

Esta especificação fornece notificações de alarmes e eventos sob demanda (em contraste com o fluxo contínuo de dados do OPC DA). Dentro dessas notificações estão inclusos: alarmes de processos, ações do operador, mensagens informativas e mensagens de rastreamento/auditoria (BHATT; BALDEVIA, 2005). Para transmitir alarmes e eventos do processo, o cliente OPC A&E precisa estabilizar uma conexão com o servidor, criando inicialmente um objeto *OPCEventServer*. Na próxima etapa, um objeto *OPCEventSubscription* é criado para receber mensagens do evento. Conforme Son e Yi (2010) a quantidade de eventos de retorno pode ser reduzida, definindo certos critérios de filtros, como, por exemplo, filtros por tipos de eventos,

³ Timestamp ou estampa de tempo é uma cadeia de caracteres denotando a hora ou data que certo evento ocorreu. A cadeia é geralmente apresentada num formato consistente, permitindo comparação entre duas marcas temporais distintas.

prioridade ou por fonte de evento. A Figura 11 ilustra os diferentes objetos criados pelo cliente OPC no servidor para receber eventos.

Figura 11 - Objetos criados por um cliente OPC para receber eventos



Fonte: MAHNKE, LEITNER & DAMM, 2009

4.1.4 OPC XML DA

O OPC XML DA foi a primeira plataforma independente da especificação OPC, substituindo o COM/DCOM pelas tecnologias HTTP/SOAP e *Web Service* HDA (MAHNKE; LEITNER; DAMM, 2009). O OPC XML DA foi definido a fim de garantir uma melhor interoperabilidade com plataformas não Windows e acesso à internet mais flexível (SON; YI, 2010).

A utilização de serviços Web permitiu a redução de funcionalidades para um conjunto mínimo de métodos, visando a troca de informações do OPC DA, sem a necessidade de métodos para criar e modificar um contexto de comunicação (MAHNKE; LEITNER; DAMM, 2009). Para isso, foram elaborados alguns métodos que cobrem os principais recursos do OPC DA, sendo eles: *GetStatus* (verifica o status do servidor), *Read* (lê um ou mais valores de Item), *Write* (escreve um ou mais valores de item), *Subscribe* (cria uma assinatura para uma lista de itens), *SubscriptionPolledRefresh* (troca de valores modificados de uma assinatura) e *SubscriptionCancel* (apaga uma assinatura).

Apesar da criação dos métodos citados anteriormente, a transmissão XML entre clientes e servidores é menos eficiente que a transmissão de dados binários utilizando o DCOM (SON; YI, 2010).

4.2 Principais limitações das especificações OPC

Independente de o OPC Clássico ser bem aceito no cenário industrial, algumas restrições tornam-se aparentes pelo fato do padrão ser baseado na tecnologia COM/DCOM da *Microsoft* (HUIMING; ZHIFENG, 2010). Dentre essas limitações, destacam-se:

- ✓ Utilização do protocolo em plataformas independentes, pois o OPC Clássico pode ser adequado apenas para plataformas da *Microsoft*;
- ✓ DCOM é difícil de configurar, apresentando também tempos de espera muito longos e não configuráveis. Os pacotes gerados pelo DCOM são complexos e difíceis de serem enviados pela internet, devido ao firewall;
- ✓ Limitação de gerenciamento dos independentes servidores DA, HDA e A&E;

Adicionalmente, Son e Yi (2010) relatam que os serviços baseados em tecnologia web XML fornecem uma solução viável para permitir a interoperabilidade de dados e sistemas. Porém, o OPC XML-DA não foi bem sucedido, pois apresentava um alto consumo de recursos e um limitado desempenho devido ao grande tamanho da estrutura de sua mensagem. Para sanar todos estes problemas, a OPC Foundation aproveitou algumas tecnologias (tendo como exemplo XML e .NET) para a elaboração de um padrão de objeto e arquitetura unificada, completamente orientada a serviços, visando o modelo de fabricação empresarial (HUIMING; ZHIFENG, 2010). Tal elaboração acarretou na criação do OPC UA.

4.3 OPC UA – Plataforma de Comunicações Aberta de Arquitetura Unificada

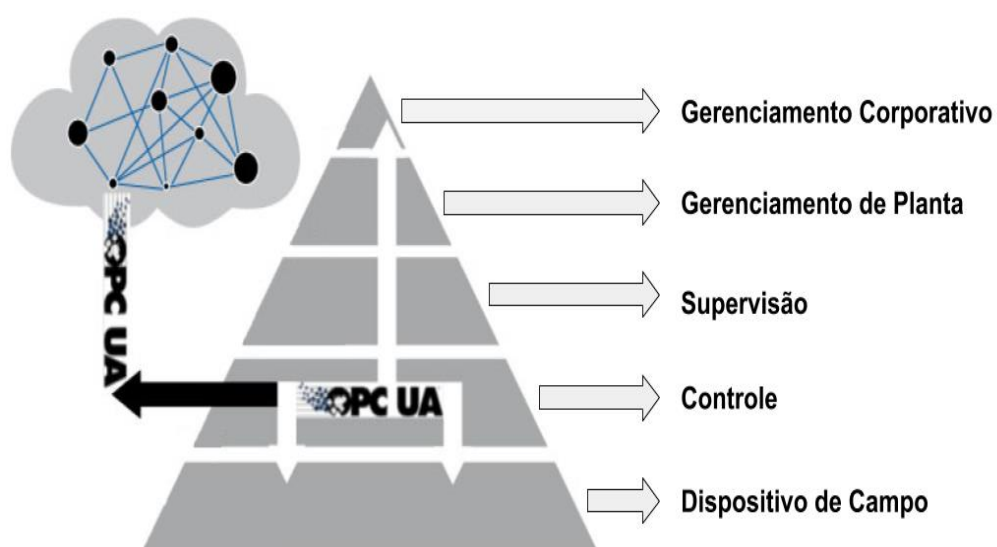
Além de introduzir um paradigma de rede distribuída baseada em SOA, Hannelius (2008) aponta que o OPC UA apresenta a independência de plataformas, não sendo dependente apenas de um fornecedor, neste caso, a *Microsoft*. Dessa maneira, tais mudanças permitem uma nova abordagem para a integração de informações e serviços, tornando o padrão mais aplicável que as especificações antigas.

Em relação a isso, Bangemann et al. (2014) relata que várias organizações como *PROFIBUS International (PI)*, *Fieldbus Foundation (FF)*, *HART Communication Foundation (HCF)*, entre outras, começaram a investigar a potencial implementação

do OPC UA. A título de exemplo, a PLCopen e a OPC Foundation estão desenvolvendo atividades comuns para definir um modelo de informação (GRUNER; PFROMMER; PALM, 2016).

Lançado oficialmente em 2008, O OPC UA é um protocolo para comunicação industrial, padronizado na IEC 62541 (GRUNER; PFROMMER; PALM, 2016), definido como um padrão independente de uma determinada plataforma, sendo que diversos tipos de dispositivos e sistemas podem se comunicar por meio de mensagens trocadas entre clientes e servidores nos mais variados tipos de redes (DURKOP et al., 2012), mantendo assim, uma arquitetura baseada em padrões de tecnologias Web. Em função disso, o protocolo busca assegurar a interoperabilidade entre seus componentes (ROHJANS; USLAR; APPELRATH, 2010) e apresenta como principal característica a definição de poderosos modelos de informações (GROSSMANN; BENDER; DANZER, 2008). De acordo com Grossmann, Bender e Danzer (2008) estes modelos de informações organizam os dados dentro do espaço de endereço (*AddressSpace*) de um servidor OPC UA em termos de estrutura e semântica, o que define o padrão, como sendo uma perfeita escolha para integração vertical dos sistemas de automação. Dessa forma, o OPC UA pode ser aplicável a diversos componentes, em todos os domínios industriais, objetivando a troca de informações entre sensores, atuadores, sistemas de controle, sistemas de fabricação e sistemas de planejamento, conforme se constata a partir da Figura 12.

Figura 12 - Aplicação do OPC UA dentro da pirâmide de automação



Fonte: Elaborado pelo autor

O OPC UA oferece uma série de recursos se comparado ao OPC Clássico (HUIMING; ZHIFENG, 2010). Dentre estes benefícios, destacam-se: espaço de endereço unificado, complexidade dos dados, múltiplas plataformas e mecanismos de segurança. A seguir serão elencadas algumas características de tais recursos.

Suporte a estruturas de dados complexos: conforme Huiming e Zhifeng (2010) as especificações anteriores do OPC fornecem uma simples organização hierárquica para dados, enquanto o OPC UA dispõe modelos de informações Metadados⁴ que podem se estender facilmente. Os autores sustentam que com isso, o cliente não precisa entender os significados com as descrições detalhadas do modelo de dados. Sendo assim, tal medida facilita à construção do software do cliente.

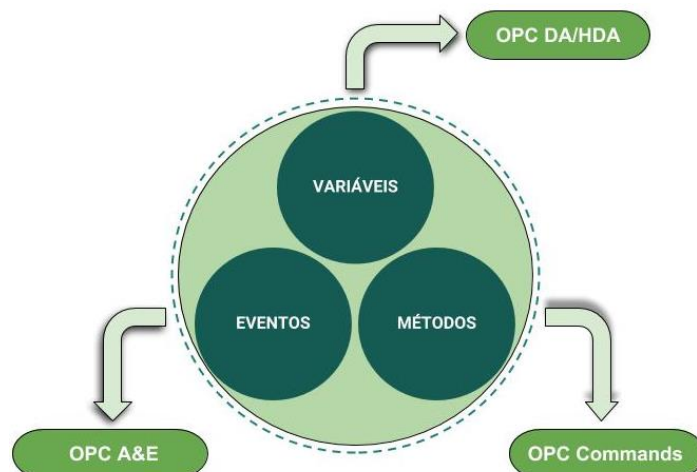
Múltiplas plataformas: o OPC UA pode ser desenvolvido em *Windows*, *Linux* e sistemas embarcados. Tal desenvolvimento só é possível pelo fato de que a *OPC Foundation* disponibiliza alguns SDKs (*Software Development Kit*) estruturados em algumas linguagens de programação, como, C/C++, C# e Java.

Mecanismos de segurança: a segurança do OPC Clássico depende exclusivamente da segurança da tecnologia COM/DCOM. Por outro lado, o OPC UA fornece um modelo flexível de segurança que pode ser utilizado nos diferentes níveis da pirâmide de automação, atendendo às exigências para cada ambiente.

Modo de acesso de dados unificado: no OPC clássico cada especificação lida com um aspecto diferente os dados (SON; YI, 2010). O OPC UA por sua vez integra os dados, alarmes e eventos em um único espaço de endereço. Com isso, o cliente OPC UA pode obter um bom conjunto de informações com apenas uma chamada (HUIMING; ZHIFENG, 2010). Conforme Son e Yi (2010), dentro do OPC UA todos os dados são tratados de forma padronizada, resultando na diminuição do tempo e desenvolvimento de aplicativos. A Figura 13 caracteriza o objeto do modelo unificado para OPC UA.

⁴ Metadados são dados sobre outros dados. Um item de um metadado pode dizer do que se trata aquele dado, geralmente uma informação inteligível por um computador. Os metadados facilitam o entendimento dos relacionamentos e a utilidade das informações dos dados.

Figura 13 - Modelo de objeto unificado para OPC UA



Fonte: Elaborado pelo autor

Os dados atuais e históricos podem ser armazenados em **variáveis**; os comandos de controle dos dispositivos são considerados como **métodos de serviços** para execução e a ocorrência de um **alarme ou evento** que provém de um dispositivo de hardware, pode ser classificada como um serviço de evento (SON; YI, 2010).

4.4 Modelagem de Informação do OPC UA

Segundo Son e Yi (2010), um dos componentes fundamentais do OPC UA é a modelagem de informação (melhorada em relação ao OPC Clássico), sendo esta de maneira geral, uma questão muito importante para o desenvolvimento de aplicativos de sistemas de informação. Tal afirmativa consiste pelo fato de que a modelagem de informação permite o enriquecimento dos dados, através da utilização de Metadados. Ela também torna possível a troca de informação com semântica conhecida, diferentemente da troca de dados puros (LEHNHOFF et al., 2011). Com isso, ao expor muito mais semântica a respeito de dispositivos, os servidores OPC UA permitem que os clientes processem tarefas sofisticadas, interpretando assim a semântica dos dados fornecidos (CHUANYING; HE; ZHILONG, 2012).

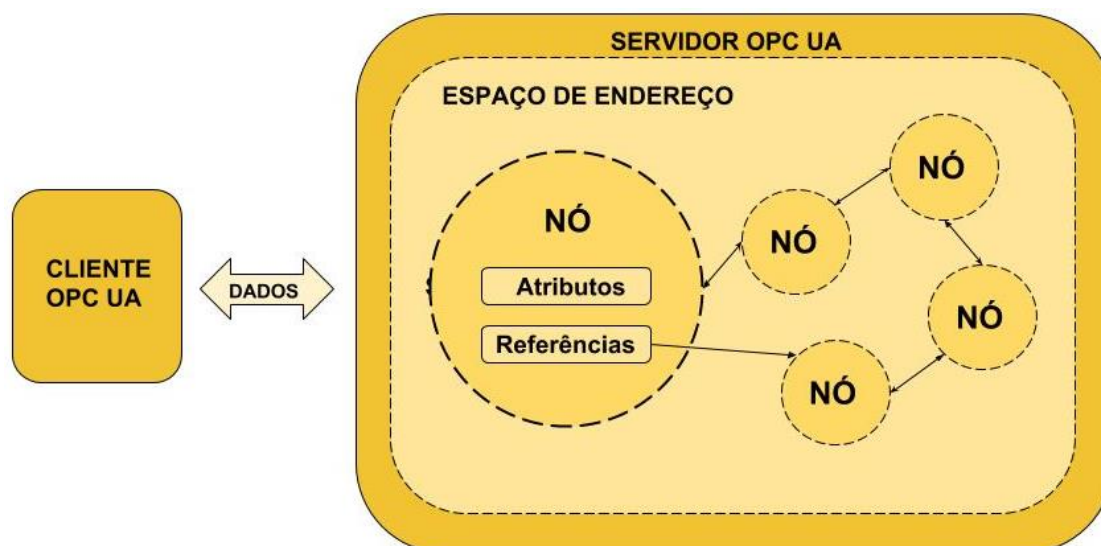
Para definir a modelagem de informação, o protocolo OPC UA estrutura um documento, também conhecido como a especificação de número 5 (*Information Model*) que fornece com detalhes as descrições de como o OPC UA integra o espaço de endereço (*Address Space*), Nós (*Nodes*) e Referências (*References*) para definir um modelo de informação.

4.4.1 Espaço de Endereço do Servidor OPC UA

De acordo com a especificação de número 3 (*AddressSpace*), um conjunto de objetos que um servidor OPC UA disponibiliza aos clientes como dados que representam um sistema em tempo real subjacente é chamado de Espaço de Endereço. Huiming e Zhifeng (2010) afirmam que o espaço de endereço é o mais importante conceito do OPC UA, sendo capaz de integrar todos os blocos funcionais.

O espaço de endereço pode ser representado por um conjunto de **Nós**, sendo que esses são descritos por **Atributos** e interconectados com outros Nós por meio de **Referências**. A Figura 14 ilustra de forma geral o espaço de endereço de um servidor OPC UA.

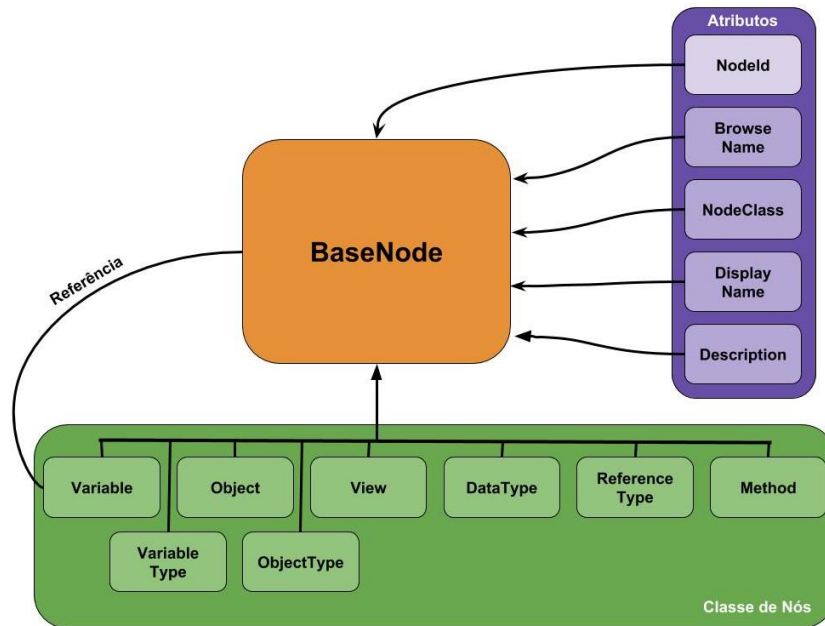
Figura 14 - Espaço de endereço de um servidor OPC UA



Fonte: Elaborado pelo autor

Conforme afirmam Leitner e Mahnke (2006), o **Nó** é o conceito base para o espaço de endereço. Diversas classes de Nós são definidas (conforme Figura 15). Os Nós têm apenas propriedades públicas chamadas de Nó Base (*BaseNode*). O Nó Base é um MetaDado e um tipo de nó abstrato que não pode ser instanciado. Além disso, todos os Nós do espaço de endereço são derivados do Nó base, como as respectivas classes de Nós: *Variable*, *Object*, *View*, *DataType*, *ReferenceType*, *Method*, *VariableType* e *ObjectType* (HUIMING; ZHIFENG, 2010).

Figura 15 - Classe de Nós a partir do Nó Base



Fonte: Elaborado pelo autor

Cada Nó pode deter um conjunto fixo de atributos, dependendo da classe do Nó. Esses atributos podem ser obrigatórios ou opcionais. Citando por exemplo, cada classe de Nó tem um ID de identificação exclusivo e obrigatório, enquanto o atributo de descrição é opcional.

Os **Atributos** são definidos como elementos de dados que descrevem os Nós. Com isso, os clientes OPC UA acessam valores de atributos através de serviços, como: leitura, escrita, pergunta e assinatura/item monitorado. Vale destacar que os atributos são determinados pelo tipo de Nó. No entanto, alguns atributos são compartilhados por todos os nós (HUIMING; ZHIFENG, 2010). Esses atributos compartilhados são citados abaixo:

- **NodeId**: é o identificador exclusivo de um nó. Tem por função identificar e localizar um nó do espaço de endereço;
- **BrowseName**: identificar um nó quando navegando no espaço de endereço;
- **NodeClass**: enumeração que define a categoria de nós;
- **Description**: uma descrição textual localizada do nó;
- **DisplayName**: contém o nome do nó que deve ser usado para exibir em uma interface de usuário.

As **Referências** são responsáveis por interligar os Nós. Suas construções são simplificadas, uma vez que elas não são Nós e não possuem atributos (LEITNER; MAHNKE, 2006). Complementarmente, uma referência descreve a relação entre dois Nós. Huiming e Zhifeng (2010) salientam que a referência não pode ser acessada diretamente, somente navegada através do Nó de origem para o Nó de destino, sendo assim, viável a organização do espaço de endereço do OPC UA como uma estrutura de malha por referência de Nós. Porém, para melhorar a interoperabilidade entre servidor e cliente, geralmente é recomendado que o espaço de endereço do servidor fosse hierarquizado, o que é semelhante à estrutura de pastas do *Windows*.

4.4.2 Classe de Nós do Espaço de Endereço

A seguir serão explanadas as principais características da Classe de Nós do espaço de endereço de um servidor OPC UA. É importante destacar que todas as classes citadas a seguir, contemplam todos os atributos da classe do Nó Base, destacados anteriormente.

4.4.2.1 Objeto (*Object*)

De acordo com *OPC Foundation* (2018), o objeto pode ser descrito como um Nó que representa um elemento físico ou abstrato de um sistema. Com isso, os objetos podem ser caracterizados como sistemas, subsistemas e dispositivos. Huiming e Zhifeng (2010) apontam que o objeto é a mais importante unidade de função no espaço de endereço, pois os Nós de objetos usam tipos de referências, organizam as variáveis e métodos, sendo também responsáveis por produzirem eventos. Um objeto pode ser definido como uma instância de um *ObjectType*. A seguinte Tabela 2 demonstra o atributo para a classe de nó *Object*:

Tabela 2 - Atributo para a classe de Nó *Object*

Atributo	Utilização	Tipo de Dado	Descrição
<i>EventNotifier</i>	Obrigatório	Byte	Este atributo representa uma máscara de bits que identifica se o objeto pode ser usado para assinar eventos e se o histórico de eventos é acessível e modificável.

Fonte: Elaborado pelo autor

4.4.2.2 Tipo de Objeto (*ObjectType*)

A classe de nó *ObjectType* fornece definições para objetos. Eles podem ser simples ou complexos. Esses expõem a estrutura do objeto, enquanto tipos simples definem apenas a semântica para o objeto (POSTÓL, 2016). A Tabela 3 destaca o atributo específico para a classe do Nó *ObjectType*.

Tabela 3 - Atributo para a classe de Nó *ObjectType*

Atributo	Utilização	Tipo de Dado	Descrição
<i>IsAbstract</i>	Obrigatório	Boolean	Indica se o <i>ObjectType</i> é concreto ou abstrato e, portanto, não pode ser usado diretamente como definição de tipo.

Fonte: Elaborado pelo autor

4.4.2.3 Variável (*Variable*)

As variáveis são usadas para representar atributos de dados do objeto. Elas contêm valor, unidade de engenharia, *timestamp*, selo de qualidade entre outras propriedades (HUIMING; ZHIFENG, 2010). Essas variáveis são subdivididas em duas categorias: *Properties* e *DataVariables* (OPC FOUNDATION, 2018).

Properties (Propriedades): são características dos objetos que podem ser definidas pelo servidor. As propriedades são diferentes dos atributos, pois caracterizam o que o nó representa, já os atributos definem Metadados que são instanciados para todos os nós de uma *NodeClass*. A *OPC Foundation* cita que um atributo pode definir o *DataType* de variáveis, à medida que uma propriedade pode ser usada para especificar a unidade de engenharia de algumas variáveis (OPC FOUNDATION, 2018). Enquanto os **DataVariables** representam o conteúdo de um objeto. Citando por exemplo, um objeto de arquivo pode ser definido contendo um fluxo de bytes. Esse fluxo é definido como um *DataVariable*. Neste caso, as propriedades podem ser usadas para expor características como, o tempo de criação e o proprietário do arquivo objeto. Outro caso exemplificado pela especificação “*Address Space Model*” é de que em sistemas de controle, os blocos de função podem ser caracterizados como objetos e os parâmetros (*setpoints*) representados como *DataVariable*. Os seguintes atributos são especificados para a classe de nó *Variable*, conforme demonstra a Tabela 4.

Tabela 4 - Atributo para a classe de Nó *Variable*

Atributo	Utilização	Tipo de Dado	Descrição
<i>DataType</i>	Obrigatório	NodeId	São representados como nós no espaço de endereço. Esse atributo contém o <i>NodeId</i> de tal Nó e, portanto define o tipo de dado do atributo valor, citado acima.
<i>ValueRank</i>	Obrigatório	Int32	Identifica se o valor é uma matriz e, quando é uma matriz, permite especificar as dimensões da matriz.
<i>ArrayDimensions</i>	Opcional	UInt32[]	Permite especificar o tamanho de uma matriz e só pode ser usado se o atributo for uma matriz.
<i>AccessLevel</i>	Obrigatório	Byte	Uma máscara de bits que indica se o valor atual do atributo de valor é legível e gravável, bem como se o histórico do valor é legível e alterável.
<i>UserAccess Level</i>	Obrigatório	Byte	Contém as mesmas informações que o <i>AccessLevel</i> , porém leva em consideração os direitos de acesso do usuário
<i>MinimumSampling Interval</i>	Opcional	Double	Fornece as informações sobre a velocidade com que o servidor OPC UA pode detectar alterações do atributo de valor
<i>Historizing</i>	Obrigatório	Boolean	Indica se o servidor atualmente coleta histórico para o valor

Fonte: Elaborado pelo autor

4.4.2.4 Tipo de Dado (*DataType*)

A classe de Nó *DataType* define o tipo do valor da variável. Essa classe fornece tipos básicos como *Int32*, *Boolean*, *Double* e também específicos, como tipo de dados de enumeração e dados estruturados (OPC FOUNDATION, 2018). Esses tipos citados têm uma missão especial, pois descrevem os dados fornecidos pelo servidor OPC UA para os clientes OPC UA (POSTÓL, 2016). Por exemplo, um Nó do *DataType* pode disponibilizar informações aos clientes de que os dados possuem um valor numérico e os clientes que o leiam podem usar esse conhecimento para interpretar e processar o valor. O atributo do nó *DataType* é descrito segundo a Tabela 5.

Tabela 5 - Atributo para a classe de Nó *DataType*

Atributo	Utilização	Tipo de Dado	Descrição
<i>IsAbstract</i>	Obrigatório	Boolean	Indica se um <i>DataType</i> é ou não abstrato

Fonte: Elaborado pelo autor

4.4.2.5 Tipo de Variável (*VariableType*)

Essa classe define o tipo de uma variável (POSTOL, 2016). *VariableType* são geralmente usadas para definir quais propriedades estão disponíveis na instância da variável (UNIFIED AUTOMATION, 2018). A Tabela 6 a seguir, ilustra os atributos ligados a Classe de Nó *VariableType*:

Tabela 6 - Atributo para a classe de Nó *VariableType*

Atributo	Utilização	Tipo de Dado	Descrição
<i>Value</i>	Opcional	Tipo de dados base	Define um valor padrão para instâncias desta classe de Nó.
<i>DataType</i>	Obrigatório	NodeId	Define o <i>DataType</i> do atributo de valor para instâncias desse tipo, bem como para o atributo de valor da classe <i>VariableType</i> (se fornecido)
<i>ValueRank</i>	Obrigatório	Int32	Identifica se o valor é uma matriz e, quando é uma matriz, permite especificar as dimensões dessa matriz
<i>Array Dimensions</i>	Opcional	UInt32[]	Permite especificar o tamanho de uma matriz e só pode ser usado se o valor for uma matriz
<i>IsAbstract</i>	Obrigatório	Boolean	Indica se o <i>VariableType</i> é abstrato e, portanto, não pode ser usado diretamente como definição do tipo

Fonte: Elaborado pelo autor

4.4.2.5 Método (*Method*)

Conforme a OPC Foundation (2018) “os métodos são funções “leves”, cujo escopo é delimitado por um objeto próprio, semelhante aos métodos de uma classe em programação orientada a objetos”. Adicionalmente, Huiming e Zhifeng (2010) apontam que o método é chamado por um cliente e retorna um resultado que é

implementado pelo servidor. Cada método é descrito por um nó de método *NodeClass*, sendo que este nó contém Metadados que identificam e descrevem o método. A Tabela 7 abaixo ilustra os atributos ligados a Classe de Nó *Method*.

Tabela 7 - Atributo para a classe de Nó *Method*

Atributo	Utilização	Tipo de Dado	Descrição
<i>Executable</i>	Obrigatório	Boolean	Uma flag indica se o método pode ser invocado no momento.
<i>User Executable</i>	Obrigatório	Boolean	O mesmo que o atribulo “ <i>Executable</i> ” levando em consideração os direitos de acesso do usuário

Fonte: Elaborado pelo autor

4.4.2.6 Tipo de Referência (*ReferenceType*)

A semântica das referências são representadas pelo *ReferenceType*. Este é usado para definir o significado do relacionamento dos Nós. Ele é usado para representar o tipo de referências usadas pelo servidor OPC UA. Os tipos de referências são divididos em duas categorias: hierárquicas e não hierárquicas. As referências hierárquicas são usadas principalmente para organizar o espaço de endereçamento e refletir os equipamentos em campo dentro da arquitetura hierárquica.

Já as referências não hierárquicas são usadas para indicar os atributos dos equipamentos, como o valor de um sensor etc (HUIMING; ZHIFENG, 2010). A seguir, a Tabela 8 demonstra o nome do atributo, tipo de dado e descrição desse atributo, para o *ReferenceType*.

Tabela 8 - Atributo para a classe de Nó *ReferenceType*

Atributo	Utilização	Tipo de Dado	Descrição
<i>IsAbstract</i>	Obrigatório	Boolean	Especifica se o <i>ReferenceType</i> pode ser usado para Referências ou se é usado apenas para fins organizacionais na hierarquia.
<i>Symmetric</i>	Obrigatório	Boolean	Indica se a Referência é simétrica, isto é, se o significado é o mesmo nas direções direta e inversa.

Fonte: Elaborado pelo autor

4.4.2.7 Visualização (View)

Segundo Leitner e Mahnke (2006) visando simplificar o acesso do cliente ao espaço de endereço, o protocolo OPC UA define o conceito de visualização. Uma visualização permite a redução do número de nós e referências visíveis em um grande espaço de endereço. Usando essa classe de nó, os servidores podem organizar seu respectivo espaço de endereço e fornecer visualizações customizadas para tarefas específicas ou casos de uso. Os atributos citados pela Tabela 9 a seguir, são específicos para a classe de nó *View*.

Tabela 9 - Atributo para a classe de Nó *View*

Atributo	Utilização	Tipo de Dado	Descrição
<i>ContainsNoLops</i>	Obrigatório	Boolean	Este atributo indica se os nós contidos na visualização fazem uma hierarquia <i>nonlooping</i> ao seguir as referências hierárquicas.
<i>EventNotifier</i>	Obrigatório	Byte	Representa uma máscara de bits que identifica se a visualização pode ser usada para assinar eventos e se o histórico de eventos é acessível e modificável.

Fonte: Elaborado pelo autor

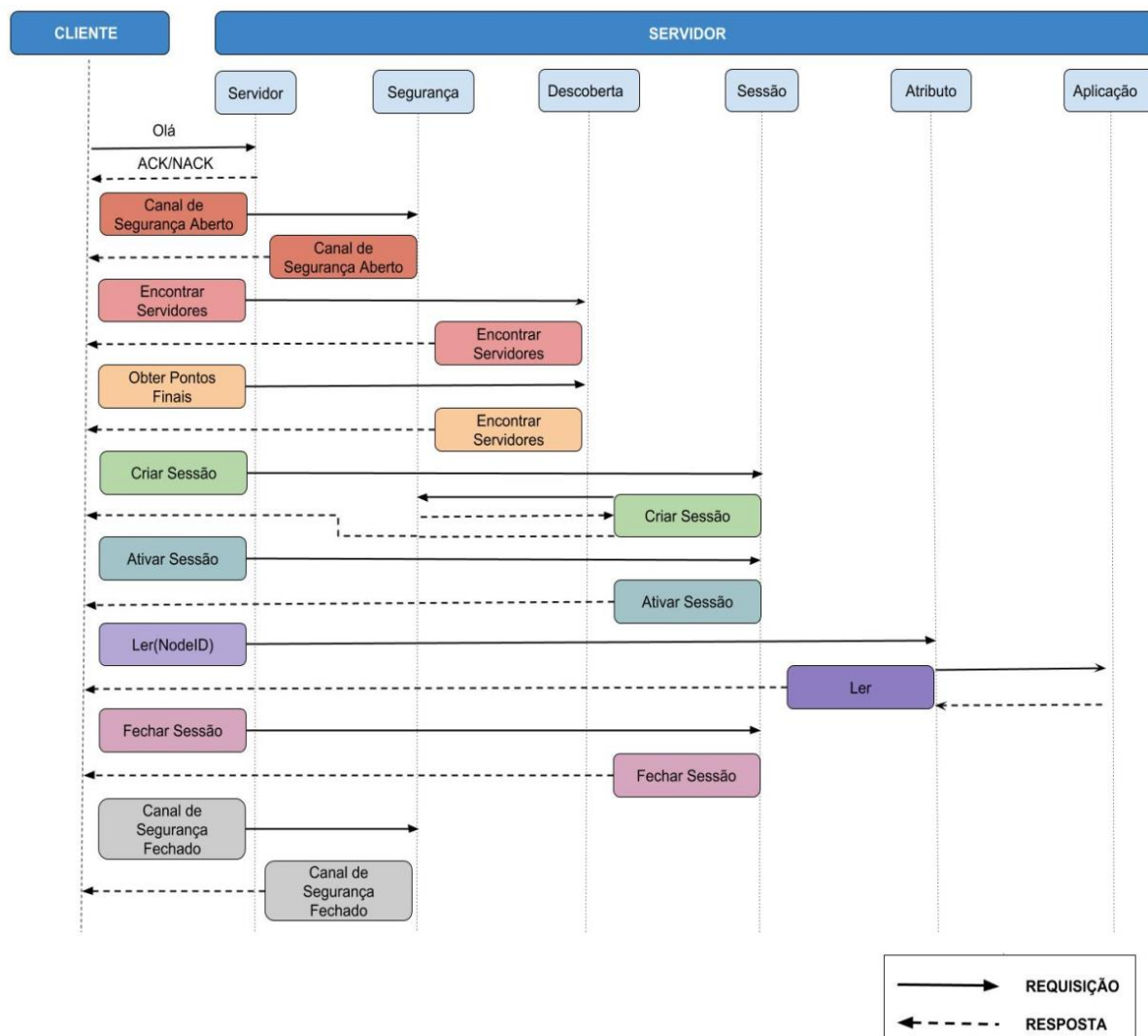
4.5 Conjunto de Serviços

Um dos conceitos mais importantes do protocolo OPC UA é com relação ao seu poderoso conjunto de serviços integrados (OPC FOUNDATION, 2018). Cada conjunto de serviços define um grupo lógico usado para acessar um aspecto particular do servidor OPC UA. A OPC Foundation fornece através da especificação de número 4 (*Services*), um total de dez conjuntos de serviços básicos. Esses conjuntos serão brevemente descritos a seguir.

A requisição de serviço e a resposta são transmitidas por meio da troca de mensagens entre clientes e servidores, conhecidos como Serviços da Web. Os serviços podem ser definidos como métodos usados por um cliente OPC UA para acessar os dados do Modelo de Informações fornecido por um servidor OPC UA (MANHKE; LEITNER; DAMM, 2009). A Figura 16 demonstra um exemplo de

comunicação entre Cliente UA e Servidor UA que utiliza alguns dos principais serviços disponibilizados dentro do protocolo.

Figura 16 - Exemplo de alguns serviços utilizados em uma comunicação entre cliente UA e servidor UA



Fonte: Elaborado pelo autor

4.5.1 Canal de Segurança (*SecureChannel*)

Um canal de segurança é uma longa conexão lógica entre um único cliente e um único servidor. Ele é utilizado para abrir um canal de comunicação que garante a integridade e confidencialidade de todas as mensagens trocadas entre cliente e servidor (OPC FOUNDATION, 2017; MAHNKE; LEITNER; DAMM, 2009). Tal canal também é designado para manter um conjunto de chaves necessárias pelas autenticações e criptografias das mensagens enviadas pela rede.

Diferente de outros serviços, os serviços do canal de segurança não são implementados diretamente no aplicativo OPC UA. Em vez disso, eles são fornecidos pela pilha de comunicação na qual o aplicativo OPC UA é construído. Citando por exemplo, um servidor OPC UA pode ser construído através de uma pilha que permite que os aplicativos estabeleçam um canal de segurança usando HTTPS. Neste cenário, o aplicativo OPC UA certamente deve verificar se a mensagem recebida está no contexto de uma conexão HTTPS (OPC FOUNDATION, 2018).

A conexão entre os aplicativos, isto é, entre cliente e servidor, se inicia através da criação de um canal de segurança. O cliente OPC UA estabelece uma conexão com o servidor OPC UA, enviando uma mensagem “*Hello Message*”. Em seguida, o servidor responde com uma mensagem de sucesso ou erro. Durante esse *handshake* inicial, algumas informações também são trocadas entre os aplicativos, como: número máximo de partes da mensagem, tamanho máximo da resposta e tamanho dos buffers de envio e recebimento (RELAY AUTOMATION, 2018).

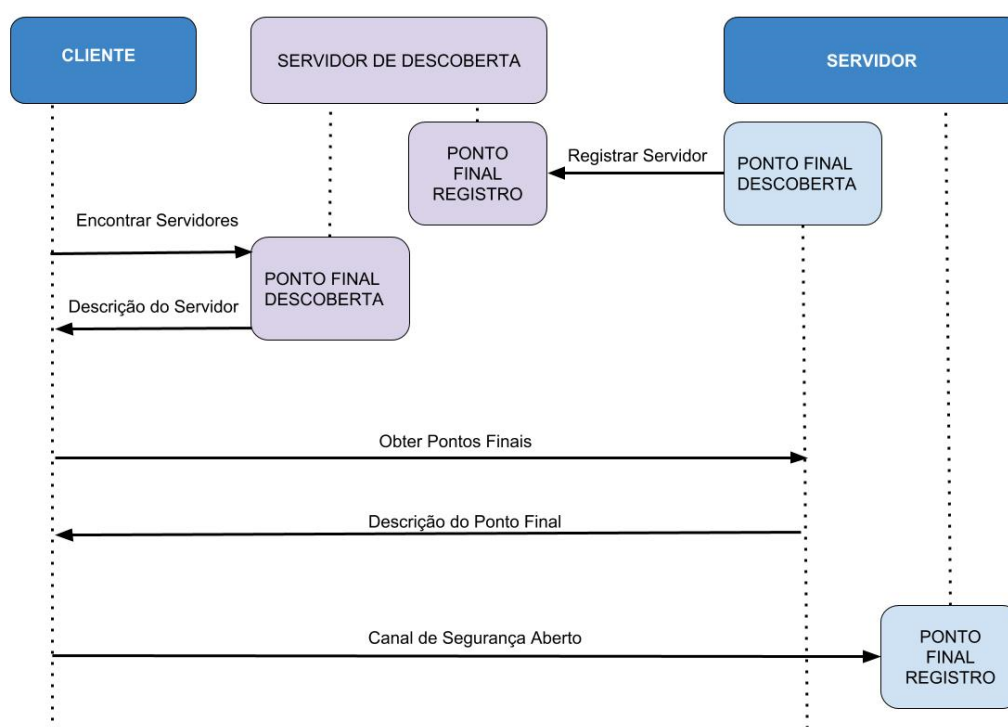
4.5.2 Descoberta (*Discovery*)

Após estabelecer um canal seguro, o cliente OPC UA normalmente solicitará a lista de servidores conhecidos do servidor de descoberta (RELAY AUTOMATION, 2018). Isso se torna possível através do conjunto de serviços de descoberta. O conjunto de serviços relacionado ao processo de descoberta é basicamente empregado para encontrar os pontos finais (*Endpoints*) implementados por um servidor e ler a configuração de segurança para esses pontos. Com isso, é possível que clientes descubram servidores OPC UA disponíveis em um sistema. Os *Endpoints* são endereços físicos disponíveis na rede que permitem que clientes acessem um ou mais serviços fornecidos por um servidor. Além disso, os *Endpoints* definem o protocolo de rede usado e as configurações de redes necessárias para conectar ao ponto final do servidor (MAHNKE; LEITNER; DAMM, 2009). Um *Endpoint* inclui as seguintes informações:

- URL do ponto final;
- Certificado do servidor associado ao ponto final;
- Modo de segurança do ponto final, informando se as mensagens devem ser assinadas, assinadas e criptografadas ou nenhuma;
- URI da política de segurança que define o perfil e o modo de segurança.

Cada servidor OPC UA deve ter um *DiscoveryEndPoint* (Ponto Final de Descoberta) que os clientes possam acessar sem estabelecer uma sessão. Os clientes leem as informações de segurança necessárias para estabelecer um canal de segurança chamando o serviço *GetEndpoints* (Obter Pontos Finais) no *DiscoveryEndPoint*. Os servidores podem se registrar com um servidor de descoberta bem conhecido usando o serviço *RegisterServer* (Registrar Servidor). Sendo assim, clientes podem descobrir posteriormente quaisquer servidores registrados chamando o serviço *FindServers* (Encontrar Servidores) no serviço de detecção. O processo de descoberta usando o *FindServers* é demonstrado conforme a Figura 17 a seguir (OPC FOUNDATION, 2018).

Figura 17 - Processo de Descoberta com Servidor de Descoberta



Fonte: Adaptado de OPC Foundation (2017)

Em sistemas de produção, os administradores podem desativar o serviço de descoberta por motivos de segurança. Para fornecer suporte a sistemas com clientes que possuem serviços de descobertas desabilitados, os administradores devem atualizar manualmente as *EndPointDescriptions* usadas para conectar-se a um servidor. Logo, os servidores devem permitir que os administradores desabilitem o *DiscoveryEndPoint*.

4.5.3 Sessão (Session)

Antes de usar os demais serviços citados adiante, o cliente deve criar uma sessão com o servidor. Nessa sessão, o cliente fornece o nome da sessão, uma cadeia aleatória de bytes e o certificado do cliente. Após isso, o servidor responde com uma sessão única denominada *NodeId*, um *token* de autenticação, uma cadeia aleatória de bytes, o certificado do servidor e a assinatura do servidor (RELAY AUTOMATION, 2018). O conjunto de serviços de sessão é usado apenas para estabelecer um canal de comunicação seguro e confiável e não para transferir informações de um sistema para outro (MAHNKE; LEITNER; DAMM, 2009).

4.5.4 Gerenciamento de Nó (NodeManagement)

Este conjunto de serviços fornece uma interface para a configuração de servidores OPC UA. Ele permite que os clientes adicionem, modifiquem e excluam os Nós e referências no espaço de endereço do servidor (OPC FOUNDATION, 2017; LEITNER; MAHNKE, 2006). Todos os Nós adicionados continuam a existir no Espaço de Endereço, mesmo se o cliente que os criou se desconectar do servidor.

4.5.5 Visualização (View)

O conjunto de serviços de visualização é utilizado para navegar no espaço de endereço do servidor (LEITNER; MAHNKE, 2006). A visualização é um subconjunto do espaço de endereço, permitindo que clientes descubram os Nós através da navegação. Esta por sua vez propicia aos clientes, o deslocamento para cima ou para baixo da hierarquia, ou até mesmo o seguimento de referências entre Nós contidos na visualização (OPC FOUNDATION, 2018).

4.5.6 Pergunta (Query)

Este conjunto de serviços é utilizado para emitir consultas a um servidor (OPC FOUNDATION, 2018) sendo compilado para consultar o espaço de endereço (LEITNER; MAHNKE, 2006). O OPC UA *Query* é caracterizado como genérico por fornecer um mecanismo de armazenamento independente, capaz de acessar uma enorme variedade de armazenamentos de dados. O *Query* permite que os usuários (clientes) acessem dados mantidos por um servidor (através do espaço de endereço) sem precisar navegar e ter conhecimento do esquema lógico usado para o

armazenamento interno dos dados, pois apenas o conhecimento do espaço de endereço é o suficiente.

4.5.7 Atributo (Attribute)

No que diz respeito ao conjunto de serviços de atributos, estes são utilizados para acessar os atributos que fazem parte dos Nós (OPC FOUNDATION, 2018). Para Leitner e Mahnke (2006), tal conjunto de serviços é destinado para ler e gravar valores de atributo. Os autores salientam que os atributos são características primitivas dos nós definidos pelo OPC UA e também os únicos elementos no espaço de endereço permitido a terem valores de dados.

4.5.8 Método (Method)

Os métodos representam as chamadas de função de objetos. O conjunto de serviços do método define os meios para invocar métodos. Os métodos são executados até a conclusão quando chamados. Eles podem ser chamados com parâmetros de entrada, sendo estes, específicos do método e podem também retornar parâmetros de saída, específicos do método (OPF FOUNDATION, 2018).

4.5.9 Item Monitorado (MonitoredItem)

Os clientes OPC UA utilizam esse conjunto de serviços para assinar dados e eventos. Cada Item *MonitoredItem* identifica o item a ser monitorado, como, por exemplo, qualquer atributo de Nó e também a subscrição a ser usada para enviar notificações. Além disso, os serviços em torno desse conjunto permitem especificar uma banda morta, taxas de atualizações para atributos e filtros para eventos (LEITNER; MAHNKE, 2006). Para indicar ao servidor como o item deve ser amostrado, avaliado e relatado, são definidos quatro parâmetros principais para *MonitoredItems*, sendo: modo de monitoramento, intervalo de amostragem, filtros e atributos de fila (OPC FOUNDATION, 2018).

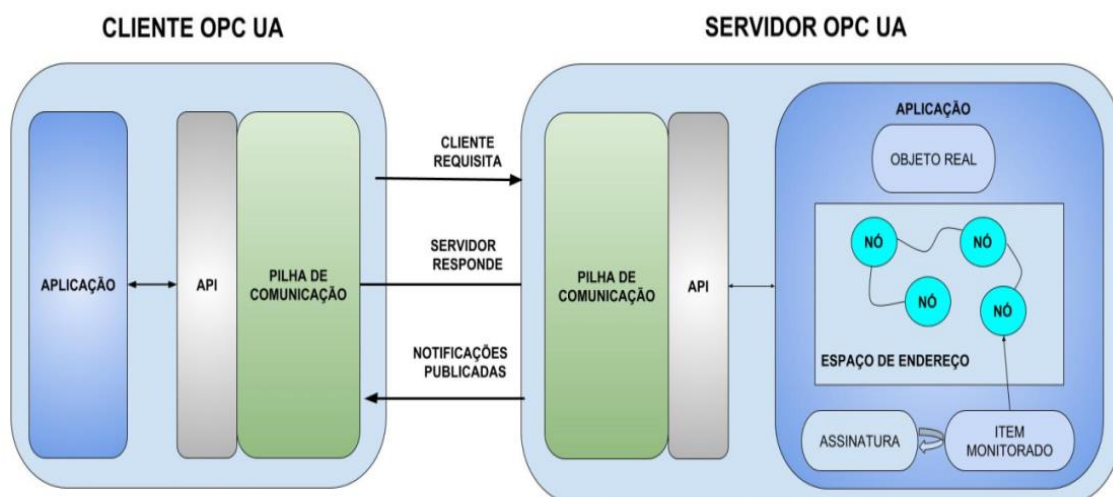
4.5.10 Assinatura (*Subscription*)

Este conjunto de serviços é utilizado pelo cliente OPC UA para criar e manter assinaturas, sendo que estas, podem ser definidas como entidades que publicam periodicamente mensagens de notificações (*NotificationMessages*) para itens monitorados (*MonitoredItems*) atribuídos a elas. Uma vez criada, a existência de uma assinatura se torna independente da sessão entre o cliente OPC UA com o servidor OPC UA. Dessa forma, um primeiro cliente pode criar uma assinatura e um segundo, possivelmente um cliente redundante, pode receber as mensagens de notificações dele (OPC FOUNDATION, 2018).

4.6 Arquitetura do OPC UA

Na parte 1 das suas especificações, classificada por “*Overview e Concepts*”, a OPC Foundation afirma que a arquitetura do protocolo OPC UA modela clientes e servidores para interagir como parceiros. Cada sistema pode conter diversos clientes e servidores, sendo que cada cliente pode relacionar-se com um ou mais servidores e o mesmo se aplica para o servidor, com relação à interação aos clientes (OPC FOUNDATION, 2018). Essa combinação de cliente e servidor formando então a arquitetura do OPC UA pode ser ilustrada conforme a Figura 18. Adiante, serão relatadas as principais características do cliente e servidor OPC UA.

Figura 18 - Arquitetura do sistema OPC UA



Fonte: Elaborado pelo autor

4.6.1 Cliente OPC UA

A arquitetura do cliente OPC UA modela o ponto final da integração entre cliente/servidor, resultando assim, na implementação de um cliente que é realizada por meio de uma pilha de comunicação. Para acessar essa pilha, o cliente utiliza uma API OPC UA. Essa API, voltada para o cliente, por sua vez, é utilizada para enviar e receber serviços, requisições e respostas do servidor. Outra característica importante da API é a de que ela isola o código da aplicação da pilha de comunicação. É indispensável frisar que essa API não é padronizada. Dessa forma, ela pode variar por diversas linguagens de programação, sistemas operacionais e potencialmente, por muitas pilhas de comunicação (OPC FOUNDATION, 2018).

4.6.2 Servidor OPC UA

Seguindo a base citada no parágrafo anterior, porém agora para o servidor, a arquitetura do servidor OPC UA modela o ponto de extremidade da integração entre cliente/servidor, resultando assim na implementação de um servidor. O aplicativo OPC UA é o código que executa o servidor, usando uma API apropriada para enviar e receber mensagens dos clientes. Uma característica similar ao caso do cliente é de que a API, agora do servidor, isola o código da aplicação da pilha de comunicação (OPC FOUNDATION, 2018). De acordo com Huiming e Zhifeng (2010) o servidor OPC UA pode ser desenvolvido com base em seis partes, sendo elas:

- Módulo de configuração e segurança: este é responsável pelas configurações de parâmetros quando o servidor é inicializado e também por criar um canal de comunicação de segurança especial entre servidor e cliente, autenticando o certificado do cliente antes de iniciar a troca de dados.
- Módulo de acesso de dados: trata da representação e uso dos dados de automação em servidores. Este módulo fornece serviços como leitura, escrita e monitoramento dos dados.
- Módulos de alarme e eventos: examina o estado dos equipamentos e produz alarmes/eventos.
- Módulo de acesso à história: fornece funcionalidades como examinar e salva dados históricos, ou até mesmo eventos históricos.

- Módulo de espaço de endereço e armazenamento de dados: constrói o espaço de endereço relacionado aos objetos reais. Ele cria as *Views* conforme diversas tarefas. Este módulo também coleta e armazena dados em tempo real e dados históricos do dispositivo em campo.

4.7 Mapeamentos de Tecnologias

Diferentes mapeamentos de tecnologias são combinados para a criação de perfis de pilha (*Stack Profiles*), sendo que todos os aplicativos OPC UA devem implementar pelo menos um perfil de pilha e só podem se comunicar com outras aplicações OPC UA que utilizam o mesmo perfil (NSHIAH; SCHAPPACHER; SIKORA, 2017). Sendo assim, a especificação de número 6 do OPC UA retrata que os mapeamentos podem ser definidos em *UA Web Services* ou *UA Native* (LEHNHOFF; ROHJANS; MAHNKE, 2011). O mapeamento *UA Web Services* utiliza *SOAP* (*Simple Object Access Protocol*) e algumas especificações de *Web Services*, enquanto o *UA Native* implementa uma simples rede binária e integra mecanismos de segurança (OPC FOUNDATION, 2018). Os principais elementos que compõem o mapeamento de tecnologias são: codificação dos dados, transporte e segurança (OPC FOUNDATION, 2018). Tais elementos são fundamentais para a criação da pilha de software do OPC UA e alguns aspectos das camadas que constituem o mapeamento de tecnologias são relatados:

4.7.1 Codificação dos Dados

Responsável pela codificação dos dados entre clientes e servidores. Essa camada pode ser feita em UA Binário ou em XML (*XML Extensible Markup Language*) (MAHNKE; LEITNER; DAMM, 2009).

A codificação feita em UA Binário apresenta um tamanho de mensagem menor em relação à codificação em XML, o que a torna mais rápida. Em contrapartida, a codificação em XML permite que clientes genéricos SOAP ⁵interpretem os dados da mensagem SOAP, enquanto eles só obteriam uma *string* binária caso utilizassem a

⁵ SOAP é um protocolo para troca de informações estruturadas em uma plataforma descentralizada e distribuída. Ele se baseia no XML para seu formato de mensagem, e normalmente baseia-se em outros protocolos da camada de aplicação.

codificação UA Binário. Complementarmente, há a codificação UA JSON ⁶(*JavaScript Object Notation*) que foi desenvolvida para interoperar com a Web e softwares empresariais que utilizam esse tipo de formatação (OPC FOUNDATION, 2018).

4.7.2 Segurança dos Dados

A camada de segurança é imprescindível para execução de uma comunicação segura, sendo responsável por assinar, codificar e decodificar mensagens trocadas entre clientes e servidores OPC UA. A especificação de número 2 propõe o modelo de segurança, incluindo diferentes mecanismos e parâmetros de segurança. Antes de abrir um canal seguro, o servidor e o cliente devem concordar com o mecanismo de segurança usado para a sessão de comunicação (HASKAMP et al., 2017). Para isso, são definidas quatro políticas de segurança para a encriptação dos dados:

- *None* (nenhuma): sem nenhuma configuração de segurança;
- *Basic128Rsa15*: configurações médias de segurança;
- *Basic256*: configurações médias para altas de seguranças;
- *Basic256Sha256*: configurações altas de segurança.

Em se tratando de segurança, outro ponto a ser considerado diz respeito à autenticação dos dados. Para a autenticação, a especificação de número 7 disponibiliza as seguintes formas para trocar *tokens* de identidade do usuário (NSHIAH; SCHAPPACHER; SIKORA, 2017):

- *Anonymous* (Anônimo): não é necessária nenhuma identidade de usuário;
- *User Name Password*: combinações de nomes e senhas para identificar usuário;
- *X.509 Certificate*: par de chaves público/privado para identidade do usuário.

4.7.3 Transporte dos Dados

Responsável pelo transporte dos dados entre clientes e servidores. Em referência a camada de transporte, o mapeamento *UA Web Services* emprega o

⁶ JSON é um modelo para armazenamento e transmissão de informações no formato texto. Esse modelo é muito difundido para aplicações Web diante da capacidade de estruturar informações de forma mais compacta que o XML, tornando mais rápido o parsing dessas informações.

SOAP/HTTPS (*Hyper Text Transfer Protocol Secure*), enquanto o *UA Native* executa diretamente no *UA TCP/IP* (*Transmission Control Protocol/Internet Protocol*) (STOPPER; KATALINIC, 2009). Vale destacar que para auxiliar nas questões de *firewalls*, o mapeamento *UA Native* também permite a integração de mensagens binárias codificadas em mensagens SOAP, utilizando HTTPS (LEITNER; MAHNKE, 2006).

4.8 Camadas de Software do OPC UA

Antes do desenvolvedor começar a projetar a arquitetura de sua aplicação OPC UA, ele deve pensar em especificar os requisitos e as funcionalidades, levando em consideração alguns objetivos de design, como, desempenho, segurança e portabilidade (MAHNKE; LEITNER; DAMM, 2009). A arquitetura de aplicativos OPC UA esclarece limites dentro e entre softwares. Essa arquitetura define os níveis de funcionalidades visíveis, como, aplicação, kit de desenvolvimento de software (SDK) e pilha de comunicação.

A camada de aplicação inclui basicamente dois tipos de aplicativos: clientes e servidores. Esses aplicativos usam a interface fornecida pelo SDK para se comunicarem entre si. Ambas as pilhas SDKs (do cliente e do servidor) podem ser implementadas em uma variedade de linguagens, sendo C/C++, Java e .NET as mais relevantes (PALONEN, 2010).

De acordo com Palomen (2010) “os SDKs são uma camada de software construída no topo das pilhas, destinadas a lidar com tarefas de baixo nível e ocultar a complexidade desnecessária do programador de aplicações”. A camada do SDK possui as APIs específicas tanto para servidores, como para clientes, aplicando em ambas, os conceitos do OPC UA. O SDK pode ser considerado como um facilitador durante o desenvolvimento de aplicações, pois constitui partes gerais da estrutura OPC UA, sendo estas, razoavelmente extensas. O SDK também cria uma API simplificada para usar a pilha. À vista disso, o desenvolvedor pode se concentrar nos recursos necessários para a aplicação (negócios, especificações de controle etc) (ASIKAINE, 2014). A pilha além de ser um aglomerado de bibliotecas de software que efetuam um ou mais perfis de pilha, é também usada para invocar serviços UA em todos os limites do processo da rede. Para a criação da pilha, o OPC UA utiliza as camadas citadas nos mapeamentos de tecnologias.

5 ANÁLISE COMPARATIVA ENTRE AS IMPLEMENTAÇÕES OPEN SOURCE DE SOFTWARE DO OPC UA

Com base no tópico anterior, para integrar o OPC UA dentro do Dispositivo de Controle, visto que este deve implementar um servidor no nível de aplicação, torna-se necessário uma análise comparativa entre as principais implementações de software open-source que implementam o SDK e a pilha, para definir qual é a melhor implementação para a realização da arquitetura proposta.

Inicialmente, é realizada uma análise descritiva dos principais projetos open-source atuais. Além de apresentar as particularidades de cada um desses projetos, são descritos quando possíveis, alguns exemplos de como e onde esses projetos estão sendo aplicados.

5.1 Node-OPCUA

O projeto node-opcua consiste de uma implementação do OPC UA escrita em *JavaScript* e *Node.js*. A escolha pelo *Node.js* se deu por esta linguagem ser uma excelente estrutura para projetar aplicativos assíncronos. O node-opcua pode ser executado em todas as plataformas que suportam *Node.js*, como *Linux*, *Mac* e *Windows* (ROSSIGNON, 2017). Em termos de licença, o *node-opcua* utiliza a licença MIT e atualmente encontra-se na versão 0.0.64.

A documentação do projeto node-opcua conta com inúmeros testes funcionais e um conjunto de exemplos práticos para ajudar o programador a estruturar suas aplicações OPC UA. Um exemplo disso são as possibilidades de criação de clientes e servidores OPC UA disponíveis em tutoriais que abordam desde a preparação até a execução e testes das aplicações. Outro ponto a ser considerado é utilização do *node-opcua* em sistemas embarcados. No site oficial está relatado que o projeto vem sendo otimizado para implementações em *Raspberry Pi*, permitindo assim, a integração do servidor OPC UA com sensores e dispositivos de entradas e saídas (ROSSIGNON, 2017). Derivando-se do projeto node-opcua, existem alguns repositórios que tratam diferentes tópicos, como o node-opcua-isa95, sendo uma extensão da ISA95 para um servidor OPC UA, e o opcua-commander que fornece um GUI para um Cliente OPC UA.

A respeito da utilização do *node-opcua*, uma aplicação do OPC UA que controla uma impressora 3D é apresentada (BAUMANN et al., 2017). No estudo realizado por Eruvankai (2017) é desenvolvido um conceito de uma IHM configurável e adaptável que monitora plantas discretas de produção, com base nas tecnologias OPC UA e *AutomationML*. Runarsson (2018) apresenta uma alternativa de PLC de hardware e software de código aberto usando o *node-opcua* para criação do servidor da aplicação.

5.2 Open62541

O projeto open62541 é uma implementação escrita em linguagem C/C++ e licenciado sobre o *Mozilla Public License v2.0*. A biblioteca do open62541 é utilizável com boa parte dos principais compiladores, além de fornecer as ferramentas necessárias para criação de aplicações de clientes e servidores OPC UA (PROFANTER, 2018). O open62541 é escalável, suportando arquitetura *Multi-Threaded*, em que cada conexão ou sessão pode ser operada por *threads* individuais (KOZAR, 2016). Destaca-se que o open62541 é portátil, pois suporta POSIX, o que torna o projeto habilitado em diferentes plataformas, como, *Windows, Linux, MacOS, Android* e Sistemas Embarcados.

O open62541 está na sua terceira versão (v0.3) e disponibiliza uma extensa e excelente documentação para suportar a criação de clientes e servidores OPC UA.

A respeito da utilização do open62541, Contrenas (2017) expõe as características essenciais que permitem que um sistema de fabricação seja adaptado para se tornar um aplicativo da I4.0. Neste trabalho a parte de comunicação é implementada usando o OPC UA. Johansson (2017) discute como uma simulação interativa de modelos de sistemas físicos para o *OpenModelica* pode ser projetada usando um cliente OPC UA. O estudo proposto por Muller (2017), apresenta um servidor OPC UA funcionando através de uma placa Arduino Yun.

5.3 Eclipse Milo

O Eclipse Milo foi apresentado durante a *EclipseCon Europe* em 2016 (Eclipse, 2016). O Eclipse Milo, parte do *Eclipse IoT Project*, é uma implementação de código aberto do OPC UA baseado na linguagem de programação Java. O projeto é desenvolvido pela Fundação Eclipse que fornece uma boa documentação e licenciado

sob o EPL -1.0. O Eclipse Milo oferece uma pilha e SDK em que há um núcleo de código comum compartilhado entre cliente e servidor (REIMANN, 2018).

A respeito da utilização do Eclipse Milo, a Eclipse (2018) apresenta o conceito de um adaptador de dispositivos que permite que o dispositivo seja facilmente conectado a sistemas de produção usando o OPC UA. O projeto apresentado por Profanter (2017), avalia algumas implementações de código aberto e certifica a Eclipse Milo como uma escolha perfeita para um sistema de produção multinível *Plug & Produce*.

5.4 ASNeg

O ASNeg (*Automation Service Next Generation* – Serviço de Automação de Próxima Geração) é uma implementação desenvolvida em C++ e licenciado sob o Apache 2.0. Além de fornecer soluções relacionadas a clientes e servidores OPC UA, o ASNeg conta com 9 repositórios para outras aplicações, incluindo OPCUADB (Base de dados do Servidor), OPCUaWebServer e OPCUARaspberry (HUEBL, 2017).

5.5 UA.NET

Desenvolvido diretamente por membros da *OPC Foundation*, o UA.NET é uma implementação desenvolvida em C# usando o *.NET Standard Library* (MOLDOVEN; BARNSTED; OURSEL, 2018). O UA.NET fornece uma pilha OPC UA contendo uma série de aplicativos de amostras. O UA.NET é licenciado sobre o GLP e RCL. O RCL permite que os membros da OPC Foundation implementem suas aplicações usando a pilha UA.NET sem que seja obrigatório divulgar o código do aplicativo. Enquanto isso, os não membros devem divulgar o código de suas aplicações ao usar a pilha UA.NET (OPC FOUNDATION, 2016). O *.Net Standard Library* permite que se desenvolva aplicativos que funcionem em todas as plataformas comuns disponíveis, incluindo *Linux*, *iOS*, *Android* (via *Xamarin*) e *Windows 7/8/8.1/10* (incluindo edições embarcadas/IoT), sem requerer modificações específicas da plataforma. Além disso, também são suportados aplicativos e serviços em nuvem (como *ASP.Net*, *DNX*, *Azure Web*, *Azure Webjobs*, *Azure Nano Server* e *Azure Service Fabric*) (MOLDOVEN; BARNSTED; OURSEL, 2018).

5.6 FreeOPCUA

O *FreeOPCUA* é um projeto composto por um conjunto de bibliotecas do OPC UA, construídas em C++ e *Python*. O *FreeOPCUA* conta com uma biblioteca LGPL C++ para desenvolvimento de aplicações de clientes e servidores OPC UA, utilizando a linguagem de programação C++ que é denominada “freeopcua”. Outra possibilidade é pela aplicação da biblioteca escrita em *Python*, fornecida através do repositório no GitHub, chamado “python-opcua” (RYKONANOV, 2018). Além disso, o *FreeOPCUA* fornece duas soluções de GUI (*Graphical User Interface*): *opcua-client-gui* e *opcua-modeler*. A primeira implementa as funcionalidades básicas, incluindo a alteração de dados e eventos, escrita de valores nas variáveis, listagem de atributos e referências. A segunda é usada para projetar espaços de endereços usando XML, o que permite que o XML produzido seja portátil para qualquer SDK (RYKONANOV, 2018).

A respeito da utilização do *FreeOPCUA*, Moldoven, Barnsted e Oursel (2018) usam o OPC UA num modelo proposto para implantar o conceito de metrologia para a Indústria 4.0. Akerman, Berglund e Ekered (2016) apresentam soluções que exemplificam a integração vertical e horizontal de sistemas usando OPC UA. Dentro dessas soluções é implementado um sistema RFID combinado com a implementação *FreeOPCUA*.

5.7 Análise comparativa com base nos requisitos estabelecidos

Para os projetos citados anteriormente, foram avaliados (com base no capítulo anterior) os respectivos requisitos: conjuntos de serviços implementados, mapeamentos de tecnologias fornecidos, políticas de segurança e versões de projetos. Sendo assim, a Tabela 10 demonstra a comparação dos requisitos estabelecidos em função das implementações de código aberto do OPC UA.

Tabela 10 - Análise Comparativa entre Implementações de Software de Código Aberto do OPCUA

Parâmetros		node-opcua	open62541	Eclipse Milo	ASNeg	UA.NET	FreeOPCUA (Python)	FreeOPCUA (C++)
Versão do código		v0.2.2	v0.3	v0.2.1	v1.6.2	-	v0.90.6	v0.2.2
Mapeamentos de Tecnologias	UA-TCP UA-SC UA Binary	✓	✓	✓	✓	✓	✓	✓
	HTTP WS-SC UA Binary			✓		✓		
	HTTP WS-SC UA XML			✓		✓		
	Nenhuma	✓	✓	✓	✓	✓	✓	✓

Políticas de Segurança	Basic128Rsa15	✓	✓	✓	✓	✓	✓	
	Basic256	✓		✓	✓	✓	✓	
	Basic256Sha256	✓		✓	✓	✓		
Autenticação	Anonymous	✓	✓	✓	✓	✓	✓	✓
	User Name Password	✓	✓	✓		✓	✓	
	X509 Certificate			✓		✓	✓	
Serviços de Descoberta	FindServers	✓	✓	✓	✓	✓	✓	✓
	GetEndpoints	✓	✓	✓	✓	✓	✓	✓
	RegisterServer	✓	✓	✓	✓	✓	✓	✓
Serviços de Canal de Segurança	OpenSecureChannel	✓	✓	✓		✓	✓	✓
	CloseSecureChannel	✓	✓	✓		✓	✓	✓
Serviços de Sessão	CreateSession	✓	✓	✓	✓	✓	✓	✓
	CloseSession	✓	✓	✓		✓	✓	✓
	ActivateSession	✓	✓	✓	✓	✓	✓	✓
	Cancel			✓	✓	✓	✓	✓
Serviços de Visualização	Browse	✓	✓	✓	✓	✓	✓	✓
	BrowseNext	✓	✓	✓	✓	✓	✓	✓
	RegisterNodes	✓	✓	✓	✓	✓	✓	✓
	UnregisterNodes	✓	✓	✓	✓	✓	✓	✓
Serviços de Atributos	Read	✓	✓	✓	✓	✓	✓	✓
	Write	✓	✓	✓	✓	✓	✓	✓
	HistoryRead			✓	✓	✓	✓	
	HistoryUpdate			✓	✓	✓	✓	✓
Serviços de Itens Monitorados	CreateMonitoredItems	✓	✓	✓	✓	✓	✓	✓
	ModifyMonitoredItems	✓	✓	✓	✓	✓	✓	
	SetMonitoringMode	✓	✓	✓	✓	✓	✓	
	SetTriggering			✓	✓	✓	✓	
	DeleteMonitoredItems	✓		✓	✓	✓	✓	✓
Serviços de Assinatura	CreateSubscription	✓	✓	✓	✓	✓	✓	✓
	ModifySubscription	✓	✓	✓	✓	✓	✓	✓
	DeleteSubscriptions	✓	✓	✓	✓	✓	✓	✓
	Publish	✓	✓	✓	✓	✓	✓	✓
	Republish	✓	✓	✓	✓	✓	✓	✓
Serviços de Gerenciamento de Nós	AddNodes		✓	✓	✓	✓	✓	✓
	AddReferences		✓	✓	✓	✓	✓	✓
	DeleteNodes		✓	✓	✓	✓	✓	✓
	DeleteReferences		✓	✓	✓	✓	✓	
Serviços de Perguntas	QueryFirst			✓	✓	✓	✓	
	QueryNext			✓	✓	✓	✓	

Fonte: Elaborado pelo autor

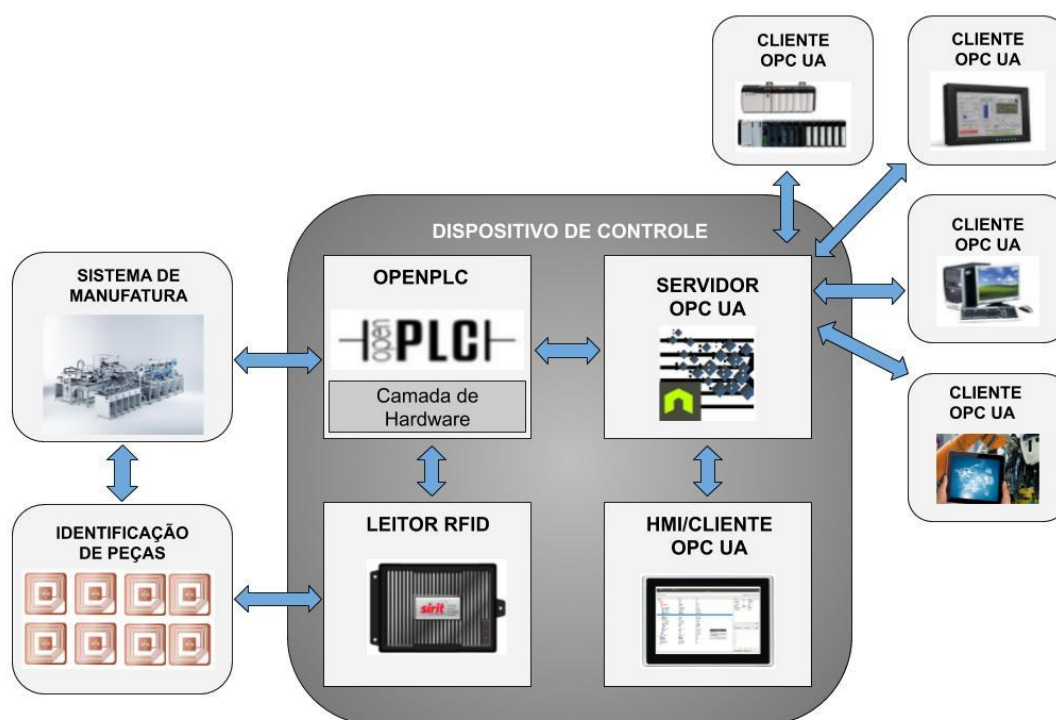
6 MATERIAIS E METÓDOS

6.1 Proposta do Trabalho

Este estudo propõe o desenvolvimento em código aberto de um Dispositivo de Controle para aplicações da Indústria 4.0, utilizando o RAMI 4.0 como modelo de referência, juntamente com o protocolo OPC UA para a realização da comunicação padronizada e interoperável entre dispositivos e sistemas. Além disso, o Dispositivo de Controle visa a integração das respectivas funcionalidades: programação de processos, identificação de peças por rádio frequência, comunicação em rede e supervisão de equipamentos.

Para esse Dispositivo de Controle, todas as informações relativas ao processo controlado serão disponibilizadas em tempo real para qualquer tipo de plataforma conectada à rede e/ou Internet que seja capaz de interagir através dos protocolos de comunicação adotados para esta proposta, num ambiente da Indústria 4.0. Diante disso, a Figura 19 demonstra a integração das tecnologias que compõem a arquitetura do sistema do Dispositivo de Controle para a Indústria 4.0.

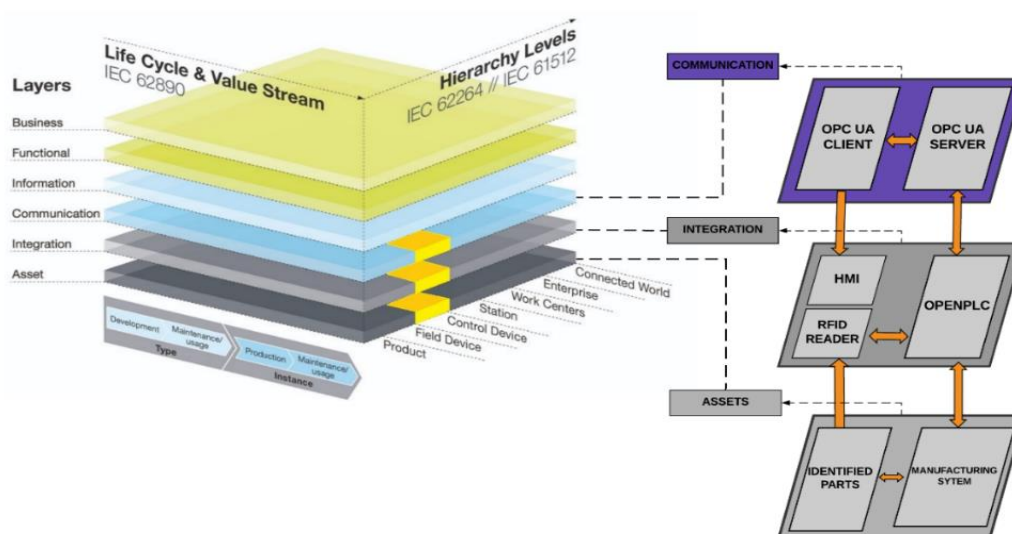
Figura 19 - Arquitetura proposta para o desenvolvimento de um Dispositivo de Controle para Industria 4.0



Fonte: Elaborado pelo autor

Com base nas pesquisas realizadas durante este estudo, observou-se que ao integrar as tecnologias envolvidas através da Figura 19, utilizando um Modelo de Referência, torna-se possível fornecer uma maneira comum de descrever as interações complexas e os requisitos fundamentais entre os elementos associados na arquitetura proposta. Sendo assim, o foco dessa proposta é relacionar a arquitetura do Dispositivo de Controle proposto, com as camadas do RAMI 4.0. Para isso, foram selecionadas as seguintes camadas: **Ativos Técnicos**, **Integração** e **Comunicação**. A Figura 20 demonstra a relação entre a Arquitetura do Dispositivo de Controle e os elementos do RAMI 4.0.

Figura 20 - Relação entre as tecnologias do Dispositivo de Controle e o RAMI4.0



Fonte: Elaborado pelo autor

6.1.1 A camada de Ativos Técnicos para o Dispositivo de Controle

Uma das principais funcionalidades do Dispositivo de Controle proposto é a identificação de peças em relação aos requisitos de processamento realizados por máquinas e estações de trabalho. Essa funcionalidade é encontrada na camada de **Ativos Técnicos**, uma vez que esta representa toda realidade física, contendo objetos como, processos, máquinas, sensores, atuadores e toda documentação técnica necessária para a produção.

Dependendo do tipo de ativo, este pode executar um **processo controlado** através de um dispositivo de controle, como um PLC, implementado diante dos

requisitos da norma IEC 61131-3 ⁷(LUDER et al, 2017). De maneira geral, os ativos podem ser localizados em unidades de trabalho, estações, grupos de funções e componentes de hierarquia do sistema de produção (HELL et al, 2016). Sendo assim, para relacionar a camada de ativos com a arquitetura do sistema proposto, serão implementadas, além do processo controlado, **etiquetas RFID para identificações individuais de peças**. Com isso, é possível obter informações sobre quais processos e estações de trabalho uma peça ou produto deve ser submetido, automatizando a realização de tarefas de acordo com diferentes produtos. Além disso, o dispositivo proposto deverá ser capaz de identificar qual estação de trabalho está executando o respectivo processo controlado, envolvido na aplicação da rede da I4.0.

6.1.2 A camada de Integração para o Dispositivo de Controle

Em se tratando de uma arquitetura em que as informações dos ativos devem estar disponíveis para as camadas superiores (Informação, Funcional e Regra de Negócios), o RAMI 4.0 disponibiliza a camada de **Integração**. Nesta camada estão contidos todos os elementos associados a TI (como softwares para a programação de PLCs) incluindo até mesmo, a possibilidade de se empregar uma IHM, gerando eventos com base nas informações adquiridas e executando o controle final dos processos (WANG; TOWARE; ANDERL, 2017). Com isso, para a implementação da camada de integração neste projeto, será utilizado o projeto **OpenPLC** (ALVES, 2017). O OpenPLC é um projeto de código aberto de integração de software e hardware para aplicações de automação e supervisão de processos (ALVES et al., 2014). Complementando o desenvolvimento e a integração de tecnologias, nesta camada, também será implementado como um objeto físico, um *display touch screen*, pensando na adição de uma **IHM** para manipulações dos dados do Dispositivo de Controle por parte dos operadores que possam estar envolvidos numa aplicação real da I4.0. Além disso, outro objeto físico que estará contido ao Dispositivo de Controle, estando presente na camada de integração, é o **leitor RFID**, tendo este por propósito o processamento das informações provenientes das etiquetas que são fixadas nas peças a serem submetidas para determinados processos em estações de trabalho. Uma vez extraídas as informações das etiquetas RFID, o leitor deverá disponibilizar

⁷ IEC 61131 é um padrão internacional para controladores programáveis industriais, contemplando o projeto completo desde hardware, instalação, testes, documentação, programação e comunicação.

essas informações para que elas possam ser armazenadas em variáveis internas do OpenPLC. Dessa forma, essas informações deverão estar disponíveis durante o desenvolvimento da programação de processos para específicas peças/produtos. Como atualmente o projeto OpenPLC não fornece suporte direto a nenhum tipo de leitor RFID, torna-se imprescindível o desenvolvimento de uma API, capaz de realizar o devido compartilhamento de informações das etiquetas, bem como as respectivas configurações do leitor RFID a ser escolhido.

6.1.3 A camada de Comunicação para o Dispositivo de Controle

Para controlar as interações que ocorrem na camada de integração, o RAMI 4.0 fornece a camada de **Comunicação**. Esta camada é responsável por permitir a comunicação entre diferentes elementos da rede com base em protocolos de comunicação uniformes, como o OPC UA. Tendo em vista a arquitetura proposta, será criado um servidor OPC UA, responsável por disponibilizar para clientes OPC UA todas as informações referentes a identificação de peças, estados do processo controlado e o desempenho do Dispositivo de Controle durante a execução de tal processo num sistema real. Pensando em possíveis limitações físicas que possam vir a ocorrer durante os processos de instalação e manuseio do Dispositivo de Controle em um sistema produtivo, será empregado um cliente OPC UA interno para validar o pleno funcionamento do servidor OPC UA e exibir localmente, através de uma IHM composta por um *display touch screen*, todas as informações organizadas para uma determinada aplicação.

O servidor OPC UA de aplicação a ser desenvolvido será baseado no projeto *node-opcua* (<https://github.com/node-opcua> - projeto que implementa a pilha e o sdk do padrão OPC UA). Este servidor será responsável pela disponibilização dos estados lógicos das entradas e saídas referentes aos sensores e atuadores controlados pelo processo programado (desenvolvido sob o OpenPLC). Além disso, este servidor fornecerá através do seu espaço de endereço um objeto contendo quais peças foram identificadas através do sistema RFID (Informações: *TagID* e *UserData*) e as devidas antenas (*AntennaID*) com as quais essas peças estão se comunicando. A partir disso, qualquer cliente OPC UA poderá saber em tempo real, qual é o processo que está sendo executado para uma determinada peça e em qual estação. É importante destacar que para que tudo isso ocorra, será necessário que o servidor

OPC UA a ser desenvolvido seja capaz de se comunicar com o projeto OpenPLC, uma vez que este concentra todos os valores referentes aos pinos de entradas e saídas, juntamente com os dados que dizem respeito a identificação de peças RFID. Para isso, será necessária a implementação de uma API, capaz de realizar o compartilhamento das informações do processo controlado pelo OpenPLC com o servidor OPC UA, conectando assim, a camada de Comunicação com a camada de Integração.

Ainda, não menos importante, no espaço de endereço do servidor OPC UA, será disponibilizado também um objeto referente a alguns elementos pertencentes ao hardware do Dispositivo de Controle para integrar toda a arquitetura da proposta. Dentre estes elementos, destacam-se: percentual de memória livre, percentual do uso da CPU, temperatura da CPU, arquitetura do processador e o sistema operacional.

A IHM como cliente OPC UA, citada anteriormente, será empregada através do projeto `opcua-client-gui` (<https://github.com/FreeOpcUa/opcua-client-gui> - projeto de código aberto baseado no `freeopcua`), sendo que este fornece uma GUI de fácil usabilidade, visualização das informações e manipulação dos dados. Esse cliente OPC UA disponibilizará ao operador do Dispositivo de Controle as respectivas interações: conexão com o servidor OPC UA, navegação com ícones por tipos de nós, visualização de atributos e referências do espaço de endereço e subscrição de variáveis do OpenPLC.

Em suma, com a integração do cliente OPC UA como IHM e do desenvolvimento do servidor OPC UA, torna-se possível a disponibilização de uma vasta gama de informações para as camadas superiores, como a de Informação, Funcional e Regra de Negócios. Além disso, a proposta foca no uso do elemento Dispositivo de Controle, localizado nos níveis hierárquicos do RAMI 4.0, mas isso não se restringe a este elemento, uma vez que a camada de comunicação implementada na arquitetura proposta, fornecerá um modelo de informação padronizado e usável em possíveis cenários da Indústria 4.0 que envolvam o uso de outros elementos e eixos do RAMI 4.0.

6.2 Tecnologias para o Dispositivo de Controle

Para que haja a integração entre os elementos associados pela arquitetura proposta, faz-se necessário o envolvimento de tecnologias compatíveis com as normas da Indústria 4.0 e que sejam capazes de se comunicar através dos protocolos Industriais utilizados. Além disso, será necessária uma linguagem de programação e um sistema de identificação, sendo estes responsáveis por modularizar a relação entre as camadas de Ativos e Comunicação, com a camada de Integração do RAMI 4.0. Dentro dessas circunstâncias, este tópico justifica os elementos escolhidos estarão representados na camada de Integração.

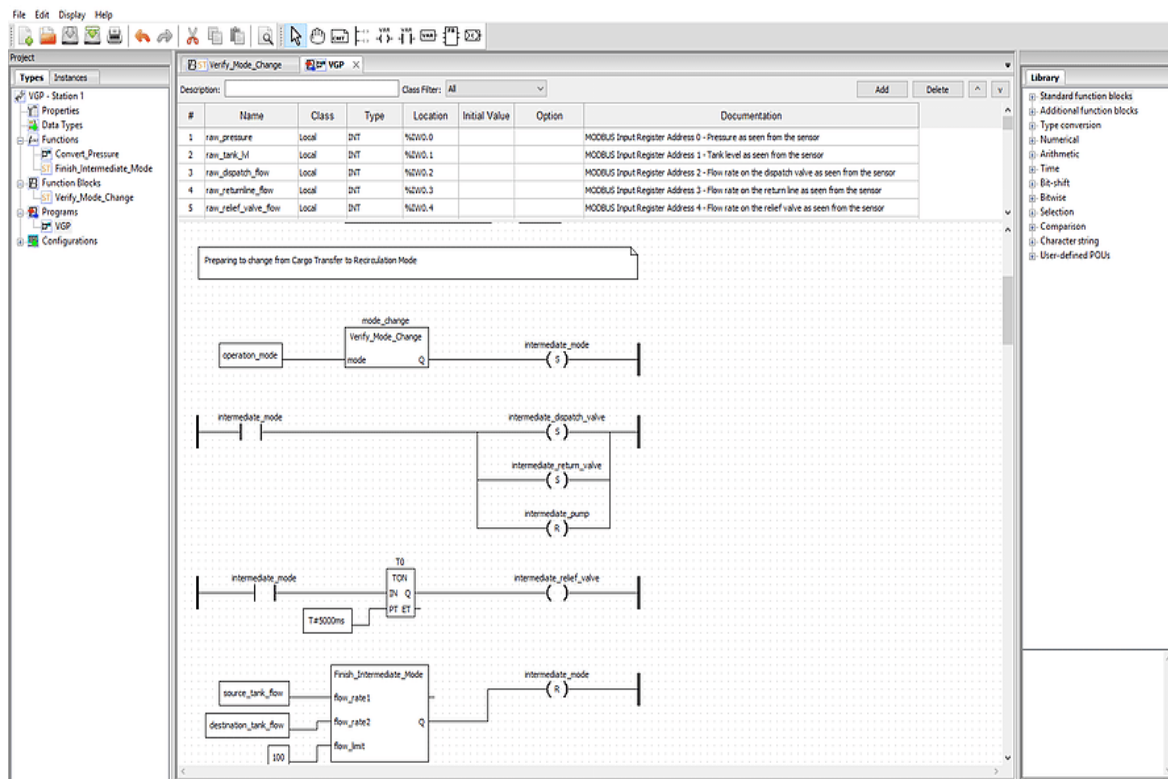
6.2.1 OpenPLC

Em países menos desenvolvidos, há uma necessidade em relação à adoção de tecnologias de baixo custo que ajudem a acelerar o desenvolvimento e a produção industrial. Não se restringindo somente aos países menos desenvolvidos, as empresas de maneira geral, estão sempre procurando alternativas para aumentar a produção, objetivando o menor prazo de entrega possível (ALVES et al., 2014). Diante deste contexto, a automação surge como uma opção para sanar tais necessidades. Apesar do PLC ser um componente de suma importância dentro do contexto industrial Alves et al (2014) afirma que esse tipo de tecnologia apresenta custos elevados, sendo inacessível por questões financeiras em diversos lugares no mundo. Ainda, outra limitação acerca do PLC tradicional é de que as empresas que fornecem esses equipamentos, não disponibilizam informações detalhadas sobre como esses controladores trabalham internamente, pois esses equipamentos são todos de código fechado (ALVES et al., 2014).

Para quebrar essas barreiras, o projeto OpenPLC apresenta um PLC de software e hardware livres, que pode ser programado nas cinco principais linguagens de programação para PLCs, definidas conforme a norma IEC 61131-3, que estabelece a arquitetura básica de software e as linguagens de programação para PLCs. Dentre as linguagens suportadas pelo OpenPLC, estão: Diagrama Ladder (*LD – Ladder Diagram*), Diagrama de Blocos Funcionais (*FDB – Function Block Diagram*), Texto Estruturado (*ST - Structured Text*), Lista de Instruções (*IL – Instruction List*) e Sequenciamento Gráfico de Funções (*SFC – Sequential Function Chart*). Outro ponto interessante a se destacar refere-se à compatibilidade do OpenPLC com o PLCopen

Editor, sendo esse um software que permite escrever programas para CLP de acordo com a IEC 61131-3, estando em conformidade com o PLCopen XML⁸. Com isso, a Figura 21 ilustra um programa desenvolvido com a linguagem Ladder através do PLCopen Editor.

Figura 21 - Programação em Ladder feita através do PLCopen Editor



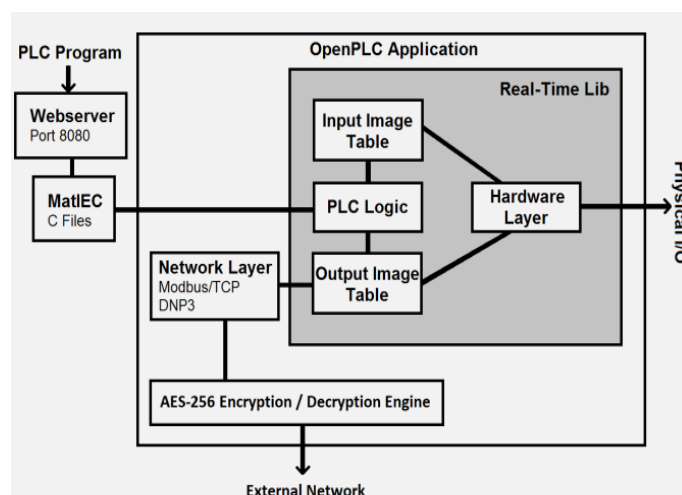
Fonte: Elaborado pelo autor

A programação do hardware é realizada por meio do PLCopen Editor, onde são gerados arquivos no formato ST (*Structured Text*). O projeto OpenPLC apresenta atualmente duas possibilidades de servidores Webs, sendo a primeira (OpenPLC_v2) baseada em Node.js e a segunda (OpenPLC_v3) em Python. Esses servidores são capazes de controlar se o OpenPLC está de fato sendo executado ou não, e permite que o usuário faça upload do arquivo ST. Durante a execução do servidor, basta abrir o navegador, que haverá uma interface Web, possibilitando o envio de novos programas ao OpenPLC.

⁸ 2 PLCopen XML é um padrão aberto e não proprietário para especificação em formato XML de programas de controladores programáveis padronizados pela IEC 61131-3, para integração com outras ferramentas.

Além do mais, o OpenPLC utiliza o projeto MATIEC que tem por objetivo o fornecimento de um compilador *open-source* capaz de suportar as cinco linguagens de programação para PLCs, relacionadas anteriormente. Em referência a camada de rede, o OpenPLC possibilita a aplicação do Modbus/TCP-IP ou DNP3. No caso da criptografia, o OpenPLC utiliza o AES-256. A Figura 22 demonstra a arquitetura do projeto OpenPLC.

Figura 22 - Arquitetura do projeto OpenPLC



Fonte: Alves (2017)

De acordo com Alves (2019), o OpenPLC relaciona o protocolo Modbus TCP/IP para comunicação SCADA. O autor ainda destaca que o Modbus é um dos protocolos mais utilizados na indústria, sendo *open-source* e isento de *royalties*. O Modbus possui quatro funções principais, que são responsáveis pela leitura ou escrita em uma E/S (Entrada/Saída) específica. O espaço de endereço de E/S no OpenPLC é mapeado para o espaço de endereço Modbus conforme a Tabela 11.

Tabela 11- Espaço de endereço de E/S no OpenPLC mapeado para o espaço de endereço Modbus.

Tipo de registrador	Uso	Endereço CLP	Endereço Modbus	Tamanho do registrador	Faixa de Valor	Acesso
<i>Coil</i>	Saída Digital	%QX0.0- %QX99.7	0 – 799	1 bit	0 ou 1	Escrita/ Leitura
<i>Discrete Input</i>	Entrada Digital	%IX0.0- %IX99.7	0 – 799	1 bit	0 ou 1	Leitura

<i>Input</i>	Entrada Analógica	%IW0- %IW99	0 – 1023	16 bits	0 – 65.535	Leitura
<i>Holding</i>	Saída Analógica	%QW0 - %QW99	0 – 1023	16 bits	0 – 65.535	Escrita/ Leitura

Fonte: OpenPLC (2019)

O projeto OpenPLC também é compatível com algumas plataformas de hardware livres, como Arduino, Raspberry Pi e ESP-8266. Adicionalmente, o projeto dá suporte à UniPI e PiXtend, sendo estas plataformas de hardwares baseadas na Raspberry Pi. Além do mais, o OpenPLC fornece os esquemas elétricos para que o usuário crie seu próprio hardware, caso não deseje utilizar nenhuma dessas plataformas para suas aplicações. Vale destacar que para a plataforma Arduino, o OpenPLC não funciona como um aplicativo autônomo, isto é, ele depende de um sistema host para execução da lógica no núcleo. O sistema host pode ser dispositivo contendo Windows, Linux; ou uma placa do tipo Raspberry Pi (ALVES, 2019).

6.2.2 Node.js

Por volta de 2005, ocorreu a revolução do *Ajax* (o uso metodológico de tecnologias como *JavaScript* e *XML*, fornecidas por navegadores para tornar páginas Web mais interativas). Nesse período, o *JavaScript* deixou de ser uma linguagem “brinquedo” para se tornar algo com que as pessoas pudessem escrever programas reais e com mais significados. Dentre esses programas, alguns se tornaram notáveis, como *Google Maps* e o *Gmail*. Mais adiante, o *JavaScript* ganhou ainda mais robustez, pois em 2008 a *Google* lançou o seu navegador, *Chrome*, e neste mesmo período, havia uma intensa concorrência entre os principais navegadores (Mozilla, Microsoft, Apple e Opera) (CANTELON et al., 2014).

Nos dias de hoje, *JavaScript* pode ser considerada a linguagem de programação mais utilizada e popular do planeta (CANTELON et al., 2014; TEIXEIRA, 2012). Grande parte dos programadores Web estão acostumados a escrever *JavaScript* no navegador, sendo o servidor, uma extensão natural disto (TEIXEIRA, 2012). Em outras palavras a linguagem *JavaScript* funciona diretamente no navegador do cliente, sendo necessário ter um interpretador embutido no navegador. A título de exemplo, no navegador Google Chrome o nome do interpretador JavaScript é o *V8 Engine*.

No ano seguinte, isto é, em 2009, durante uma apresentação realizada na “*The European JavaScript Conference*” em Berlim, Ryan Dahl apresentou uma plataforma capaz de utilizar a máquina virtual V8 para rodar JavaScript fora do navegador (especialmente na criação de servidores web) (TEIXEIRA, 2012; HUGHES-CROUCHER; WILSON, 2012). Essa plataforma ficou conhecida como Node.js. De acordo com KOTA e PRABHU (2014), o Node.js é uma plataforma de software criada com base no tempo de execução do V8 *JavaScript* para criar aplicativos de redes dimensionáveis sem esforços. Além disso, o Node.js usa um modelo de Entradas e Saídas sem bloqueio, orientado a eventos, o que o torna leve e extremamente eficiente, sendo assim, perfeito para aplicativos em tempo real que usam muitos dados e que são executados em dispositivos distribuídos.

Conforme Kota e Prabhu (2014), a plataforma Node.js utiliza como base JavaScript por uma série de fatores. Entre eles, destacam-se:

- **Assíncrono:** o JavaScript é naturalmente assíncrono com o modelo de evento adequado para a criação de aplicativos Web, altamente escalonáveis por meio de retornos de chamada;
- **Menor curva de aprendizado:** uma enorme gama de desenvolvedores estão familiarizados com JavaScript e programação assíncrona de anos;
- **Praticidade:** enormes avanços na velocidade de execução tornaram prático escrever software do lado do servidor inteiramente em JavaScript;
- **Compartilhamento de Código:** os desenvolvedores podem escrever aplicativos Web em uma linguagem, o que ajuda a reduzir a alternância de contexto entre o desenvolvimento de cliente e servidor, permitindo o compartilhamento de código entre cliente e servidor;
- **Suporte para NoSQL:** JavaScript é a linguagem usada em vários bancos de dados, como, por exemplo, CouchDB, MongoDB.
- **JSON:** é um formato de dados muito popular hoje nos dias de hoje e é JavaScript nativo.

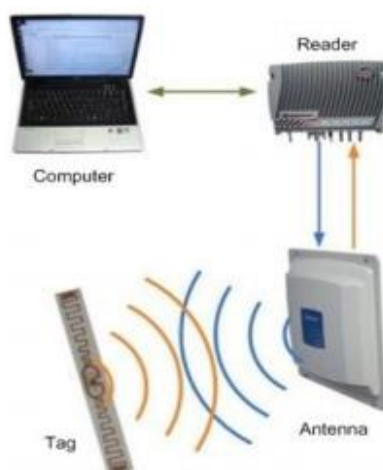
6.2.3 Sistema RFID

A produção de maneira geral é caracterizada por uma alta variedade de produtos que demandam curtos períodos de tempo para entrega, levando em consideração que

as informações geradas no chão de fábrica são indispensáveis para atingir os objetivos, sendo que estas muitas vezes não se encontram disponíveis (ENGELHARDT; REINHART, 2012). Os autores apontam que uma maneira de solucionar tal necessidade é através da utilização da tecnologia RFID, sendo esta uma tecnologia de identificação consolidada e muito eficiente (WANT, 2006) que permite monitoração e supervisão mais precisas e adequadas das informações do chão de fábrica, possibilitando também a aplicação de uma metodologia de configuração para análise do ambiente fabril. Além disso, o RFID viabiliza o rastreamento de objetos pelas linhas de produção e posteriormente o rastreamento do seu ciclo de vida. Essas informações agregadas com outros dados provenientes da IoT permitem a tomada de decisão em diferentes níveis hierárquicos da indústria, proporcionando benefícios operacionais e estratégicos.

Nos sistemas de manufatura discreta, um sistema RFID pode ser aplicado no controle da produção na busca da melhoria da qualidade e do gerenciamento do nível de eficiência da produção. O RFID detecta os objetos durante os estágios do processo de produção e a informação pode ser atualizada imediatamente num banco de dados. Portanto, o sistema de gerenciamento pode checar o status dos produtos ou da produção em tempo real. (AKBARI et al., 2015). Tornando tudo isso possível, de forma simples, porém extremamente eficiente, a tecnologia RFID é composta por um sistema típico que emprega a utilização de etiquetas (transmissoras/responsivas), leitores (transmissores/receptores) e antenas, conforme ilustra a Figura 23.

Figura 23 - Componentes de um sistema RFID



Fonte: Elaborado pelo autor

O leitor normalmente está conectado a um computador central ou outro equipamento que possua a inteligência necessária para processar os dados da etiqueta. O leitor de RFID se comunica com a etiqueta de RFID usando onda de rádio, cuja frequência de operação é determinada pelas exigências da aplicação, tais como velocidade, distância de identificação e condições ambientais, sendo atualmente usadas as frequências de 125KHz, 13,56Mhz e 915Mhz (SANCHES, 2018). O autor aponta que o funcionamento do sistema ocorre da seguinte forma: um aparelho com função de leitura, envia por meio de uma antena, sinais de radiofrequência em busca de objetos a serem identificados. No momento em que um dos objetos é atingido pela radiação, ocorre um acoplamento entre ele e a antena, o que possibilita que os dados armazenados no objeto sejam recebidos pelo leitor. Este trata a informação recebida (identificação) e a envia ao computador. O elemento que permite a comunicação entre a etiqueta e o leitor é a antena.

7 DESENVOLVIMENTO

A realização deste projeto contou com a utilização da infraestrutura do Grupo de Automação e Sistemas Integráveis (GASI) da Unesp Sorocaba.

O Dispositivo de Controle é composto por meio de um computador de placa única (SBC – *Single Board Computer*), especificamente, a Raspberry Pi (Rpi) – Modelo 3B+, atuando este como dispositivo host do sistema. A escolha pela plataforma de SBC Raspberry Pi se deve pela grande difusão e disponibilidade no mercado brasileiro, além de ser uma opção de código aberto, com adequada capacidade de processamento e disponibilidade de memória. Suplementarmente, essa SBC possui conexão Wi-Fi de forma nativa, além da conexão Ethernet cabeada, que são utilizadas para a comunicação do Dispositivo de Controle com outros dispositivos e sistemas.

Visando a eventual validação do sistema desenvolvido em um ambiente real de manufatura, tornou-se necessária a utilização de uma placa de expansão para a Raspberry-Pi, com a função de possibilitar com que a Rpi funcione como um Controlador Lógico Programável. Diante disso, optou-se por utilizar a placa UniPi1.1, pois esta além de ser totalmente compatível com a SBC escolhida, também possui uma implementação dedicada para funcionamento do projeto OpenPLC. Além disso, a UniPi1.1 emprega outros recursos de hardware, conforme demonstra a Tabela 12, possibilitando então futuras expansões de funcionalidades do Dispositivo de Controle.

Tabela 12- Recursos de Hardware da placa de expansão Unipi1.1.

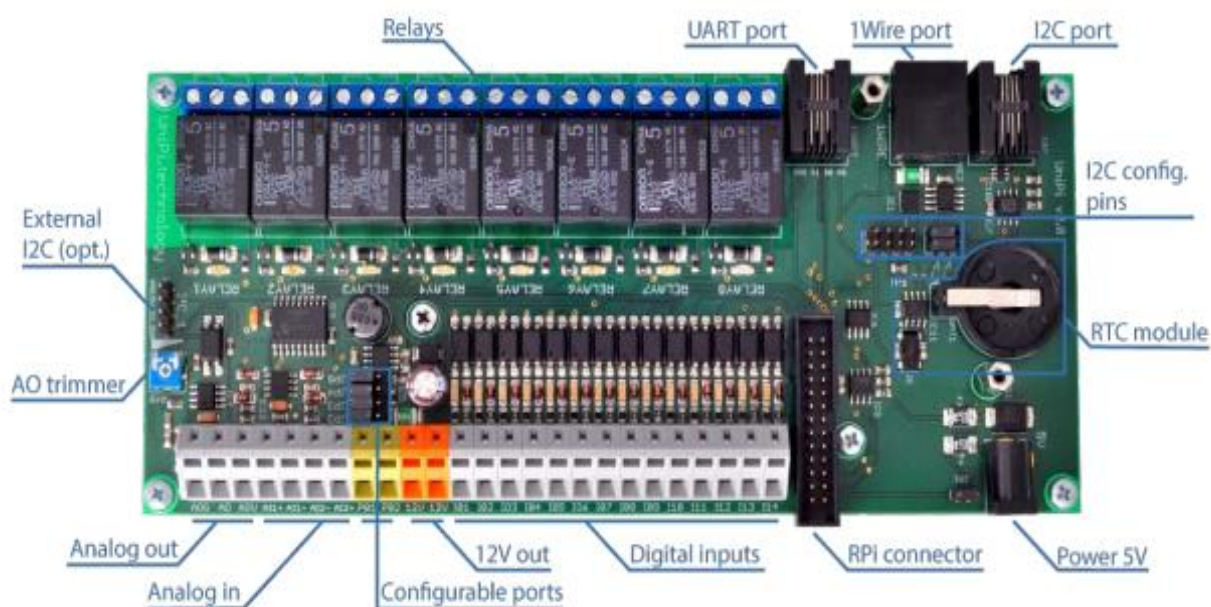
Recurso	Qtd	Descrição
Relés	8	250VAC/5A ou 24VDC – Elementos de comutação de controle.
Porta UART	1	Porta serial padrão para conectar dispositivos, como leitores, NFC etc.
Porta 1-Wire	1	Fornece interface de barramento 1-Wire para conectar dispositivos 1-Wire, como sensores de temperatura e umidade.
Porta I2C	1	Para conectar outros módulos de extensão que comuniquem a partir do padrão I2C, como relés ou módulos de saídas analógico.
Pinos de Configuração I2C	1	Para conectar outros dispositivos I2C, utilizando a interface I2C_0 da Raspberry Pi.
Módulo RTC	1	Fornece um sistema de tempo real em caso de falta de internet ou energia.

Alimentação de 5V	1	Dispõe de um conector de 2,1mm para fonte de alimentação
Conector para Rpi	1	Conector de 26 pinos para interligar a Raspberry Pi à placa de expansão
Entradas Digitais	14	Entradas digitais isoladas galvanicamente para leitura de sinais de dispositivos externos
Saída de 12V	1	Fonte de alimentação de 12V/200mA – uso somente com entradas digitais do UniPi
Portas configuráveis	1	Para configurar entradas digitais para uso com fonte de energia externa
Entradas Analógicas	2	Entradas analógicas de 0-10V para leitura de sinais analógicos de dispositivos externos.
Saídas Analógicas	1	Saída analógica 0-10V para controle proporcional de dispositivos externos com tal requisito.
Trimmer	1	Usado para ajuste preciso da saída analógica

Fonte: (UNIFI, 2019).

Com base nos recursos destacados pela Tabela, a Figura 24 a seguir demonstra a distribuição de tais recursos pela placa de expansão UniPi1.1.

Figura 24 – Distribuição dos componentes na Unipi1.1.



Fonte: (UNIFI, 2019).

O sistema operacional escolhido para o dispositivo host foi o *Raspbian*. Este é uma distribuição Linux baseada no Debian e desenvolvido justamente para Raspberry Pi. A justificativa pela escolha de tal sistema operacional se fez pelo fato deste suportar a aplicação principal, juntamente com todos os conjuntos de códigos do OpenPLC, as pilhas que implementam o protocolo OPC UA através do node-opcua e o suporte à instalação de todas as dependências. Além do mais, o Raspbian apresenta simplicidade de uso, facilidade de instalação de possíveis novos pacotes e um framework pronto e funcional.

Após relacionados os dispositivos de hardwares necessários para embarcar todos os componentes de software do projeto, é fundamental definir a interconexão entre os elementos relacionados para compor o Dispositivo de Controle com os itens presentes no RAMI 4.0.

7.1 Conexão entre as camadas de Integração e Comunicação

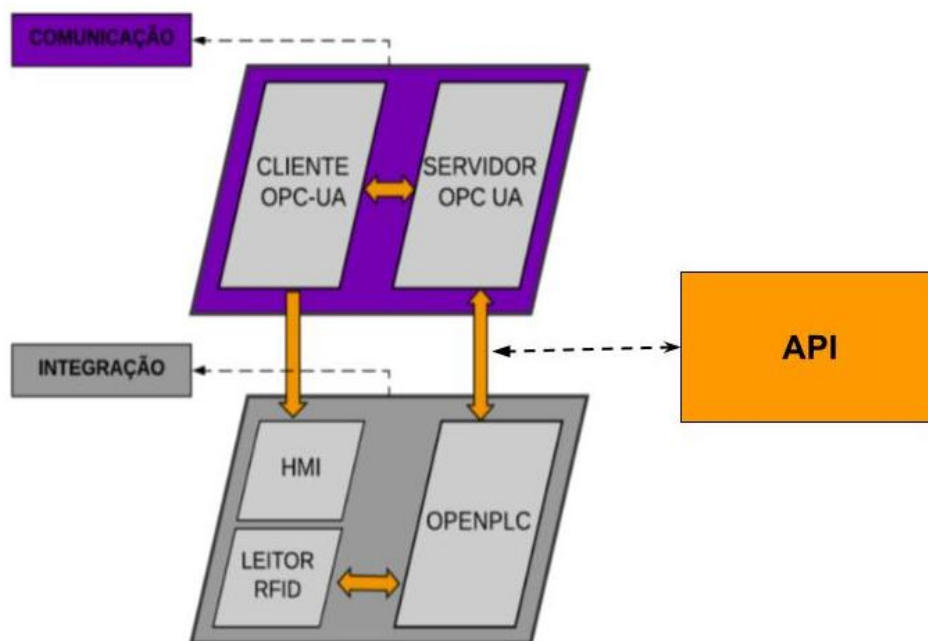
Para a conexão entre a camada de Integração e Comunicação do RAMI 4.0 diante do Dispositivo de Controle desenvolvido, fizeram-se necessárias as comunicações dos respectivos elementos:

A- OpenPLC (Integração) x Servidor OPC UA (Comunicação);

B- IHM (Integração) x opcua-gui-client (Comunicação).

Além do desenvolvimento do servidor OPC UA para o Dispositivo de Controle, foi necessário também implementar uma API, capaz de fornecer para tal servidor todos os dados do processo controlado e do sistema RFID, sendo estas informações gerenciadas pelo projeto OpenPLC. Uma vez que o servidor OPC UA está ativo, o próprio Dispositivo de Controle expõe as informações disponibilizadas pelo espaço de endereço do servidor OPC UA para a camada de Integração, através de uma IHM que se integra à um cliente OPC UA interno. Sendo assim, a Figura 25 destaca a conexão entre a camada de Integração e Comunicação para o Dispositivo de Controle.

Figura 25 - Tecnologias do Dispositivo de Controle contidas nas camadas de Integração e Comunicação do RAMI4.0



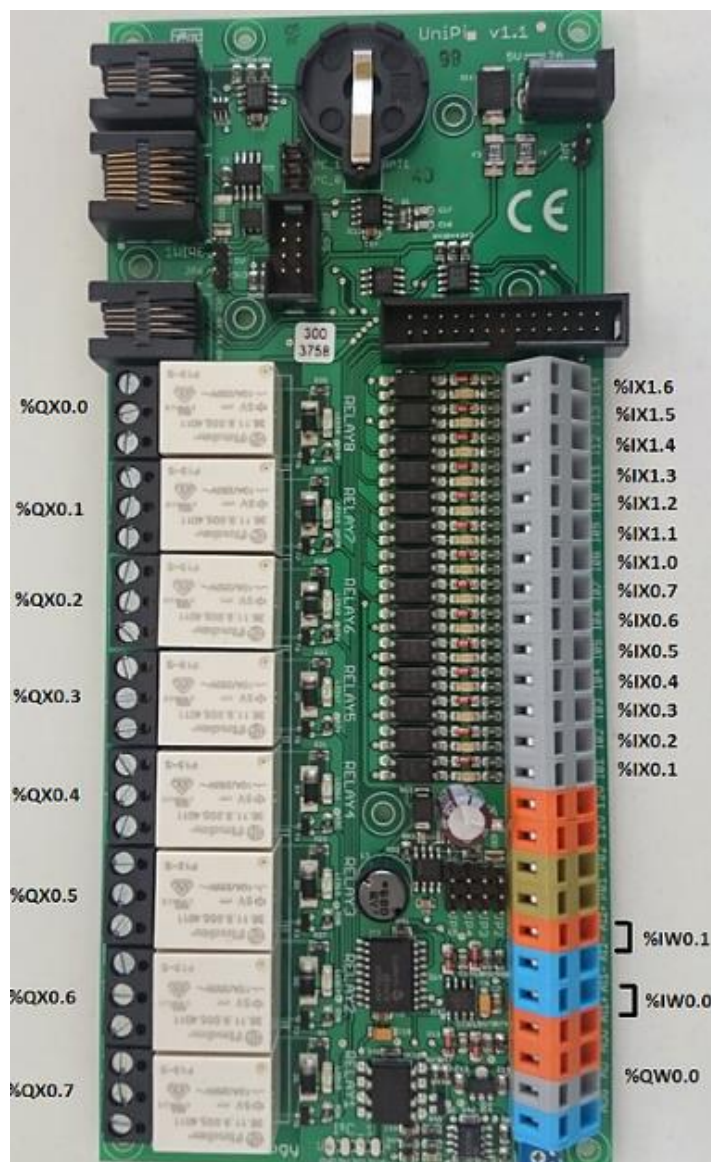
Fonte: Elaborado pelo autor

7.1.1 API entre OpenPLC e Servidor OPC UA

A API desenvolvida para a conexão entre o OpenPLC e o servidor OPC UA foi baseada no protocolo Industrial Modbus TCP/IP. Isso ocorreu pois o projeto OpenPLC incorpora um servidor Modbus que disponibiliza os dados das variáveis internas do OpenPLC para outros dispositivos e sistemas que implementem clientes Modbus. Esta API desenvolvida possui um recurso de mapeamento de endereço integrado, que permite inserir blocos de endereço comuns e, em seguida, a API cria os endereços individuais, exibindo-os através do navegador OPC UA. Além do mais, a API também poderá fornecer uma maneira de conectar o Dispositivo de Controle à qualquer dispositivo que suporte o protocolo Modbus TCP-IP.

Em tal desenvolvimento, é importante frisar que o projeto OpenPLC compatibiliza seus pinos de I/Os para a UniPI1.1, de maneira individual, sendo que os estados lógicos desses pinos são disponibilizados através de um servidor Modbus do OpenPLC. A Figura 26 aponta o mapeamento de pinos do OpenPLC para UniPI1.1, seguindo o padrão Modbus TCP-IP.

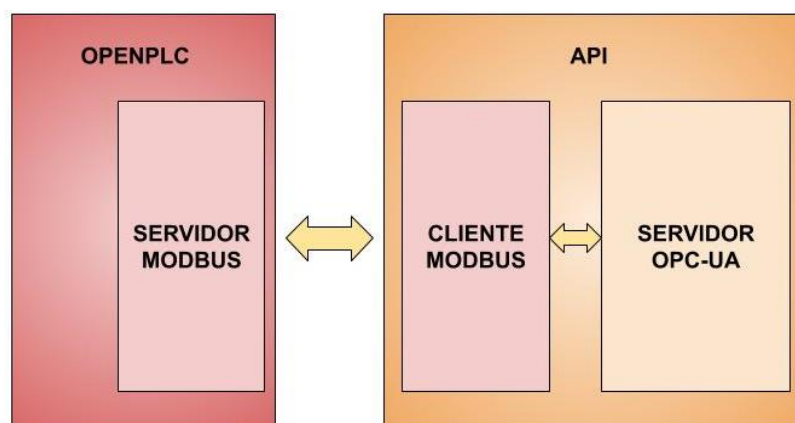
Figura 26 - Mapeamentos de pinos Modbus para UniPi1.1 no OpenPLC



Fonte: (OpenPLC, 2019)

Uma vez que os dados dos I/Os da placa UniPi 1.1 são disponibilizados via Modbus, foi implementado através da API desenvolvida um cliente Modbus que armazenasse em algumas variáveis internas, os valores dos estados dos pinos de I/O, e posteriormente compartilhasse estes valores com o servidor OPC UA, sendo que toda essa integração acontecesse na mesma aplicação. Para compatibilizar toda estrutura dentro de um sistema padrão, visando futuras expansões da arquitetura, bem como a facilitação para manutenções de código fonte, escolheu-se o Node.js (sob Java Script) para a criação do cliente Modbus (baseado no projeto de código aberto, denominado jsmodbus) e do servidor OPC UA (node-opcua). A Figura 27 demonstra a estrutura da API para integração entre o OpenPLC e o servidor OPC UA.

Figura 27 - API de comunicação entre OpenPLC e Servidor OPC UA



Fonte: Elaborado pelo autor

O cliente Modbus desenvolvido, além de possuir funções de conexão e desconexão (do servidor Modbus contido no OpenPLC), fornece também uma classe que possibilita a leitura de bobinas, leitura de entradas discretas, leitura de registrador de retenção de saída analógica, escrita em uma única bobina e escrita em um único registrador. Este cliente Modbus foi adaptado a partir do projeto jsmodbus, como citado anteriormente. O jsmodbus implementa um cliente e servidor através de uma simples API de fácil instalação, contendo maneiras para depuração da aplicação e diversos exemplos práticos de implementações. Dessa forma, a Tabela 13 demonstra o mapeamento Modbus para o UniPi 1.1 no OpenPLC, sendo este mapeamento de endereços necessário para um posterior compartilhamento de variáveis entre o cliente Modbus e o servidor OPC UA.

Tabela 13- Mapeamento Modbus para UniPi1.1

Tipo de registrador	Uso	Endereço Unipi1.1	Tamanho do Registrador	Faixa de Valor	Acesso
<i>Coil</i>	Saída Digital	%QX0.0- %QX0.7	1 bit	0 ou 1	Escrita/ Leitura
<i>Discrete Input</i>	Entrada Digital	%IX0.1- %IX1.6	1 bit	0 ou 1	Leitura
<i>Holding Register</i>	Analógica	%QW0 (PWM), %IW0- %IW1	16 bits	0 – 65.535	Escrita/ Leitura

Fonte: OpenPLC (2019).

7.1.2 Desenvolvimento do Servidor OPC UA

Para expor os dados do processo controlado pelo OpenPLC, de forma padronizável e interoperável através do uso do protocolo OPC UA, tanto para um cliente interno da arquitetura proposta quanto para outros clientes OPC UA externos, foi necessário desenvolver um servidor OPC UA em nível da aplicação. É importante salientar que este desenvolvimento é fundamentado a partir do uso da pilha e do SDK, ambos fornecidos pelo projeto node-opcua.

Em se tratando da construtividade do servidor de aplicação, inicialmente foi criada uma instância do servidor, visando à customização e comportamento do mesmo. Vale ressaltar que para a criação de servidor de aplicação, o desenvolvedor deve especificar uma porta TCP para que clientes OPC UA consigam estabelecer futuras conexões com este servidor. Essa instância criada, por sua vez, pode conter o caminho do recurso, sendo este usado para construir um identificador de recursos uniforme (URI) do ponto final do servidor. Com isso criado, o cliente precisará então, passar esse URI para se conectar ao servidor. Ainda, sobre o processo de instanciação, outras informações foram adicionadas sendo estas geradas através de um objeto criado no espaço de endereço que contém particularidades do servidor OPC UA (*ServerStatus*). Esse objeto contém as informações de construção, nome do produto, data de construção, versão de software, data atual, data de início do servidor, estado atual do servidor entre outros.

Uma vez instanciado, isto é, criado, o servidor deve ser inicializado. No processo de inicialização, o servidor carregará o seu Nó principal e preparará a ligação de todas as variáveis padrão do OPC UA. Após a inicialização, é criado o *namespace* com todas as variáveis contidas no espaço de endereço do servidor. Para isso, é fundamental criar uma função que estenderá o espaço de endereço padrão do servidor com as variáveis necessárias para atender aos requisitos da arquitetura proposta. O espaço de endereço, sendo o mais importante item do padrão OPC UA é usado para personalizar o modelo de objeto que o servidor irá expor ao mundo externo. Dessa maneira, dentro do espaço de endereço da aplicação, foram criados cinco objetos principais, visando a representação dos elementos físicos controlados pelo OpenPLC. Os três primeiros objetos estão relacionados aos registradores Modbus implementados para o Dispositivo de Controle, sendo eles: Coils, Discrete Inputs e Holding Registers. O quarto objeto por sua vez, representa as informações

referentes ao sistema de identificação de peças, usando RFID. Para essa proposta, foram criadas variáveis que dizem respeito às peças identificadas por etiquetas RFID (Informações – TagID e UserData) e também uma variável que expõe a identificação da Antena (AntennaID) usada no sistema. O quinto e último objeto criado expõe elementos referentes ao dispositivo host do sistema (Rpi), expondo em tempo real para clientes OPC UA, dados como, os percentuais de uso da CPU e memória livre e a temperatura da CPU (°C); que podem ser monitorados durante a execução de uma determinada rotina referente ao processo controlado, sendo este submetido ao sistema de manufatura. É importante destacar que todos esses objetos criados no espaço de endereço do servidor utilizaram como referência o *FolderType*. Essa estrutura permite que a informação seja visualizada de forma hierarquizada, pensando em melhorar a interoperabilidade entre os aplicativos (servidor do Dispositivo de Controle e diversos tipos de clientes OPC UA que se conectem a tal servidor).

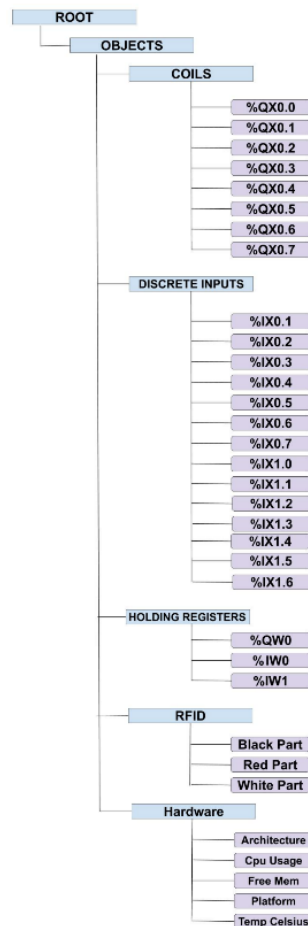
Para cada um dos objetos criados, foram definidas classes de Nós do tipo *Variable*, representando dentro da proposta, atributos de dados desses objetos. No caso dos *Coils* e *Discrete Inputs*, o *DataType* definido através do software que implementa o servidor foi o *Boolean*. Já para o *Holding Register* e *RFID*, o tipo definido foi o *Double*. Além disso, foram modificados especialmente os respectivos atributos para as variáveis criadas, em referência aos diferentes objetos: *componentOf*, *DisplayName*, *browseName*, *description* e *dataType*. Vale ressaltar que no caso do *NodeId*, se ele não for especificado para a variável criada, o servidor atribuirá automaticamente um novo *NodeId* para os respectivos Nós criados no espaço de endereço.

Uma vez criado o espaço de endereço, a próxima etapa a nível de desenvolvimento consiste em atualizar os valores referentes às variáveis controladas pelo OpenPLC (entradas e saídas digitais, entradas e saídas analógicas e identificação de peças através do sistema RFID). Todas essas informações são requisitadas pelo *driver* criado (discutido no tópico anterior) através do cliente Modbus e compartilhadas dentro de uma aplicação (desenvolvida com o Node.js) com o servidor OPC UA. Uma vez que essas informações já estão compatibilizadas com o padrão OPC UA, elas são atualizadas num período de 100ms.

Por fim, após finalizado o desenvolvimento do servidor OPC UA para o Dispositivo de Controle, torna-se possível monitorar todos os dados do processo

controlado pelo OpenPLC usando um cliente OPC UA. A Figura 28 apresenta de forma sistematizada as variáveis criadas para os diferentes tipos de objetos, dentro do espaço de endereço do servidor OPC UA.

Figura 28- Espaço de endereço para a proposta



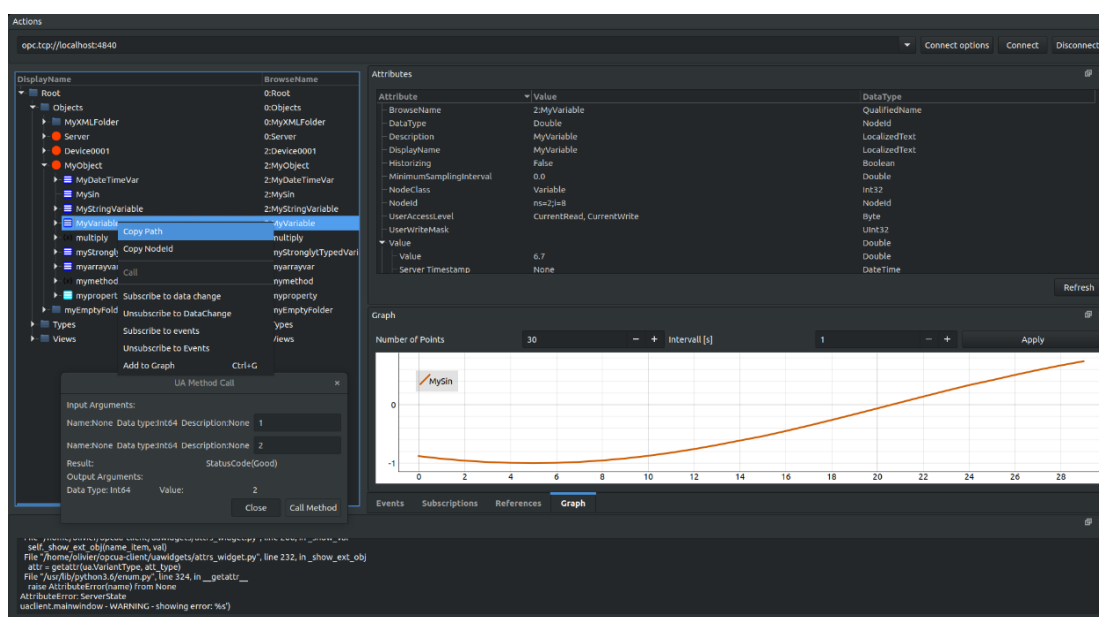
Fonte: Elaborado pelo autor

7.1.3 Comunicação entre IHM (Cliente OPC UA) e Servidor OPC UA

Visando a validação da solução proposta no item anterior, é necessária a comunicação entre um cliente OPC UA (IHM) e o servidor OPC UA, ambos contidos na arquitetura do Dispositivo de Controle. Para o uso de tal cliente, sendo este encontrado na camada de comunicação do RAMI 4.0, foi embarcado ao Dispositivo de Controle um cliente local, representado pelo projeto “opcua-client-gui” (baseado no projeto FreeOPCUA), tendo este por finalidade implementar uma IHM que possibilita ao usuário a visualização das informações do processo controlado, fornecidas através do servidor OPC UA.

A escolha pelo “opcua-client-gui se deu por este ser open-source e portátil para Raspberry Pi. O projeto “opcua-client-gui” é baseado no PyQt 5. Atualmente, este projeto atende aos requisitos do Dispositivo de Controle, disponibilizando então as funções de conexão e desconexão, navegação com ícones por tipos de Nós, visualização de atributos e referências, subscrição de uma variável, gravação de valores do Nó e inscrição em eventos. A Figura 29 ilustra a interface gráfica gerada pelo opcua-client-gui”, após a conexão com um servidor genérico.

Figura 29 - Interface Gráfica gerada pelo opcua-client-gui”



Fonte: opcua-client-gui

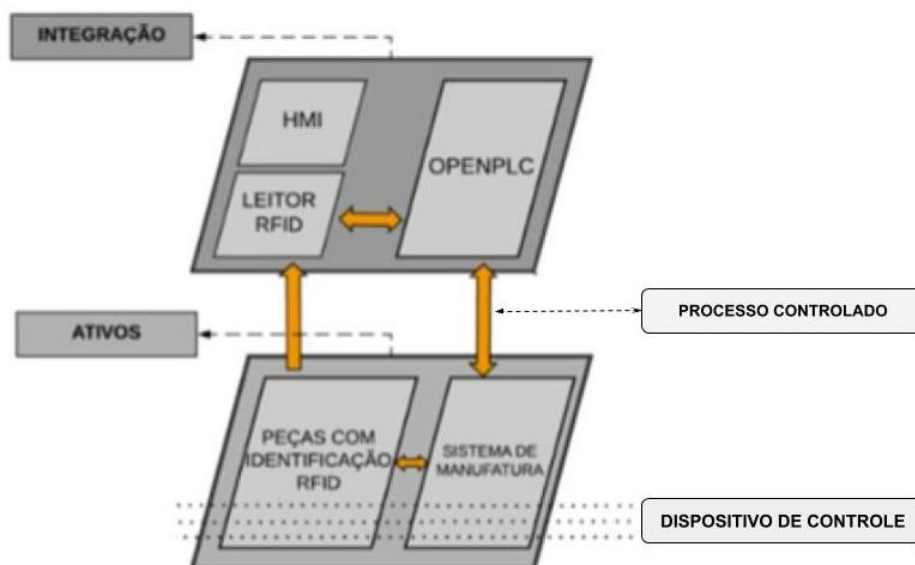
7.2 Conexão entre as camadas de Ativos Técnicos e Integração

Em se tratando do Dispositivo de Controle, a conexão entre as camadas de Ativos Técnicos e Integração ocorre de duas formas:

- A- Peças Identificadas (Ativos Técnicos) x Leitor RFID (Integração);
- B- Sistema de Manufatura (Ativos Técnicos) x OpenPLC (Integração).

As peças identificadas pelo sistema RFID além de se comunicarem com o leitor RFID, também são submetidas a determinados blocos de instruções de um sistema de manufatura. Esse sistema de manufatura, executa as ações pré estabelecidas de acordo com um processo previamente programado, através do projeto OpenPLC. A Figura 30 ilustra a conexão entre as camadas de Integração e Ativos Técnicos para o Dispositivo de Controle.

Figura 30 - Relação entre as camadas de Integração e Ativos Técnicos



Fonte: Elaborado pelo autor

7.2.1 Identificação de Peças

A identificação de peças é realizada por meio de etiquetas de RFID fixadas nas peças, conforme demonstra a Figura 31. As etiquetas são formadas internamente por um circuito eletrônico integrado e uma antena, que quando estimuladas pela rádio frequência emitida pelo leitor devolvem uma informação que as identificam.

Figura 31 - Etiquetas RFID fixadas nas peças



Fonte: Elaborado pelo autor

Com o uso das etiquetas acopladas às peças, é possível realizar o monitoramento da produção de três diferentes tipos de peças dentro da arquitetura proposta. É importante destacar que para cada peça, haverá um processo de produção diferente realizado através do sistema de manufatura.

A aquisição das informações geradas pelas etiquetas acopladas nas peças é realizada pelo leitor RFID. Este leitor, além de processar essas informações, deve

também ser capaz de compartilhá-las com o projeto OpenPLC. Para isso, foi primordial o desenvolvimento de uma API integrada e compilada junto ao código fonte do OpenPLC, que fosse capaz de realizar todas as configurações do leitor RFID escolhido (Sirit -510).

Esta API de configuração do leitor RFID compilada junto ao core do OpenPLC, tem por objetivo enviar os comandos de configurações de leitura para o leitor RFID via comunicação Ethernet TCP/IP, e receber as respectivas informações do leitor, quando as peças são identificadas pelo sistema. A API desenvolvida configura o leitor RFID no modo ativo de leitura, baseado em eventos assíncronos (INFINITY510, 2015). Com a configuração do leitor RFID no modo ativo, dois eventos relacionados à leitura de tags podem ser registrados para o Dispositivo de Controle, sendo eles:

- **event.tag.arrive:** um evento é gerado na porta de eventos do leitor toda vez que uma tag é lida pela primeira vez em uma antena. Este evento só ocorre novamente para o mesmo conjunto tag-antena quando um evento de “depart” é gerado para este par.

- **event.tag.depart:** um evento é gerado quando uma determinada tag para de ser lida por uma determinada antena, indicando que esta saiu do campo de leitura.

O conteúdo apresentado por estes eventos pode conter as seguintes informações:

- *tag_id*: Identificação da tag lida.
- *tid*: Identificação única de uma tag, proveniente do fabricante.
- *user_data*: Espaço de memória que pode ser utilizado para armazenar algum dado pertinente.
- *type*: Tipo de tag.
- *time*: Data e hora em que o evento ocorreu. No caso do evento de “arrive” este campo apresenta a data e hora em que determinada tag foi percebida por uma determinada antena. No caso do evento “depart” este campo apresenta a data e hora em que determinada tag deixou o campo de leitura uma determinada antena.
- *antenna*: Em qual antena o evento ocorreu.
- *rssi*: Indicador de força do sinal recebido.
- *tx_power*: Indicador da potência de transmissão da tag.

Os campos que serão apresentados nos eventos são determinados pelas três seguintes variáveis: *tag.reporting.arrive_field* e *tag_reporting.depart_field*, sendo que para o evento “*depart*” há ainda uma configuração adicional que define o tempo que uma tag deve ficar sem ser percebida por uma determinada antena para que o evento “*depart*” ocorra, funcionando como uma tolerância para evitar que ruídos de leitura acusem que uma tag saiu do campo de leitura de uma antena erroneamente.

Com base nestes eventos, o programa realiza a configuração e registro dos eventos de “*arrive*” e “*depart*” no leitor, de tal forma que, toda vez que uma tag entra no campo de leitura de uma antena um evento *arrive* é gerado indicando a identificação da tag lida, a antena que realizou a leitura e a data/hora em que o evento ocorreu.

O tempo de tolerância do evento *depart* configurado foi de 1 segundo, ou seja, quando uma tag que se encontra em frente a uma antena, sai do campo de leitura desta por 1 segundo um evento de *depart*, acusando que a tag não mais está sendo lida é gerado. Vale ressaltar que o campo “*time*” gerado no evento de *depart* não inclui este tempo de tolerância e, portanto, apresenta o momento exato em que a tag saiu do campo de leitura da antena.

Diante da ocorrência dos eventos de *arrive* e *depart*, as informações de *TagId* e *Antenna* são processadas pelo leitor e compartilhadas em variáveis que representam endereços Modbus no OpenPLC, sendo mais especificamente destinadas para os tipos de registradores “Holding Register”. O mapeamento das informações geradas foi realizado sequencialmente da antena 1 à 4 com a identificação da peça na primeira variável e os dados da peça na segunda. Por exemplo, o ID e os dados de uma peça identificada são mapeados, por exemplo nas variáveis %IW2 e %IW3. Vale destacar que pensando na validação do sistema, foram escolhidas três peças e fixados os respectivos endereços: Peça Branca (%IW2 e %IW3), Peça Vermelha (%IW4 e %IW5), Peça Preta (%IW6 e %IW7) e Antena (%IW8). A Tabela 14 levanta o mapeamento Modbus para o sistema RFID dentro do OpenPLC.

Tabela 14- Mapeamento Modbus para sistema RFID dentro do OpenPLC

Tipo de registrador	Uso	Endereço Unipi 1.1	Tamanho do registrador	Faixa de Valor	Acesso
<i>Holding Register</i>	Tags RFID	%IW2- %IW8	16 bits	0 a 65535	Leitura

Fonte: Elaborado pelo autor

Para a identificação de cada peça, foi enviada da peça ao leitor uma string que contém 4 caracteres correspondentes ao ID da respectiva etiqueta detectada pelo leitor. Considerando o uso limitado de peças para a validação dos resultados obtidos, precisou-se armazenar partes desses caracteres em registradores Modbus, pois só assim o OpenPLC pode ser capaz de armazenar essas informações das etiquetas em variáveis para serem utilizadas durante o desenvolvimento do processo controlado. Como o registrador em questão consegue armazenar apenas 2 bytes, foram considerados os dois primeiros caracteres enviados ao leitor. Logo, quando uma peça branca é identificada, o leitor recebe 4 caracteres, sendo eles A111 e considera para a conversão apenas A1. Diante disso, a Tabela 15 traz os valores para cada peça utilizada, nos formatos gerados pelo leitor e transformados para o OpenPLC.

Tabela 15 - Peças e seus respectivos valores de Tag IDs

Cor da Peça	Tag ID (4 bytes)	Conversão para OpenPLC (2 Bytes do Tag ID)	Endereço Utilizado	Acesso
Branças	A111	16689	%IW2	Leitura
Vermelhas	B222	16946	%IW4	Leitura
Pretas	C333	17203	%IW6	Leitura

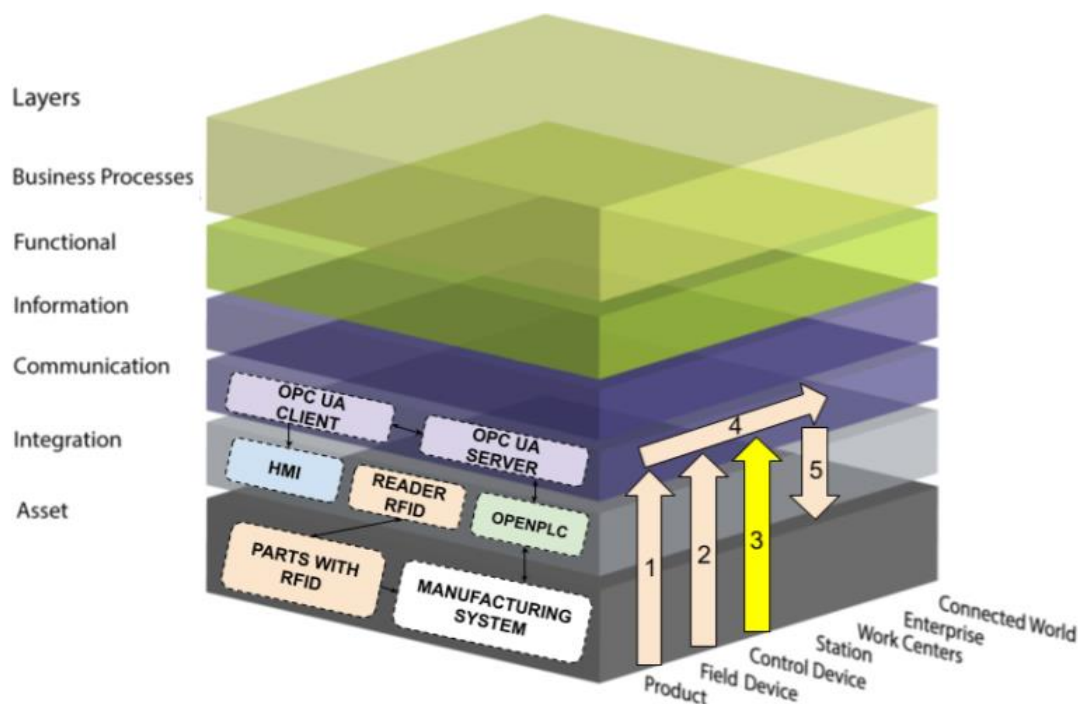
Fonte: Elaborado pelo autor

8 RESULTADOS

Um experimento de laboratório em um cenário I4.0 foi realizado como prova de conceito para validar o desenvolvimento do Dispositivo de Controle de código aberto baseado no RAMI 4.0. Equipamentos industriais relacionados a um sistema modular de produção, como o FESTO MPS foram utilizados em uma aplicação de identificação de peças (produtos) em relação aos requisitos de processamento para máquinas e estações de trabalho. A ideia é processar e classificar partes diferentes (cores diferentes) na estação de classificação usando as informações das etiquetas RFID incluídas em cada parte, na qual o Dispositivo de Controle é responsável por gerenciar todas as tarefas de lógica e controle.

Embora o experimento tenha sido focado na validação da operação do Dispositivo de Controle, ele incluiu interações entre elementos dos Níveis de Hierarquia e os eixos Camadas do RAMI 4.0, como mostra a Figura 32. A compatibilidade do Dispositivo de Controle com o RAMI 4.0 foi descrita detalhando as interações usando setas numeradas.

Figura 32 – Interações entre elementos dos eixos RAMI 4.0 no experimento de validação de dispositivos de controle.



Fonte: Elaborado pelo autor

A seta 1-Produto se refere às peças com identificação RFID, gerando informações sobre o produto (TagID e TagData) para a camada de Comunicação RAMI 4.0. A seta 2-Dispositivo de Campo representa os dados do sistema de fabricação do sistema RFID. Nesse caso, é possível identificar qual estação de trabalho está sendo usada no processo, pois as informações da antena (Antennald) também estão disponíveis para a camada Comunicação. A interação que ocorre na seta 3-Dispositivo de Controle diz respeito à operação do OpenPLC, podendo receber as informações da estação e controlar o processo de fabricação (programa Ladder de acordo com a norma IEC 61131-3), além de atribuir essas informações para a camada de comunicação. Na estação controlada, sempre que um produto registrado é identificado pelo sistema RFID, a camada de Integração obtém essas informações e executa uma lógica específica (instruções) para cada peça identificada. A seta 4-Estrutura de Comunicação representa a exibição de informações organizadas relacionadas à identificação de peças do produto, identificação do sistema de fabricação, informações de desempenho do hardware e estados lógicos dos sensores /atuadores, através do Espaço de Endereço do servidor OPC UA. Finalmente, a seta 5-Estação lista o uso por usuários/operadores (clientes OPC UA) de todas as informações disponíveis (servidor), que podem ser fornecidas através de IHMs e usadas para tarefas de supervisão.

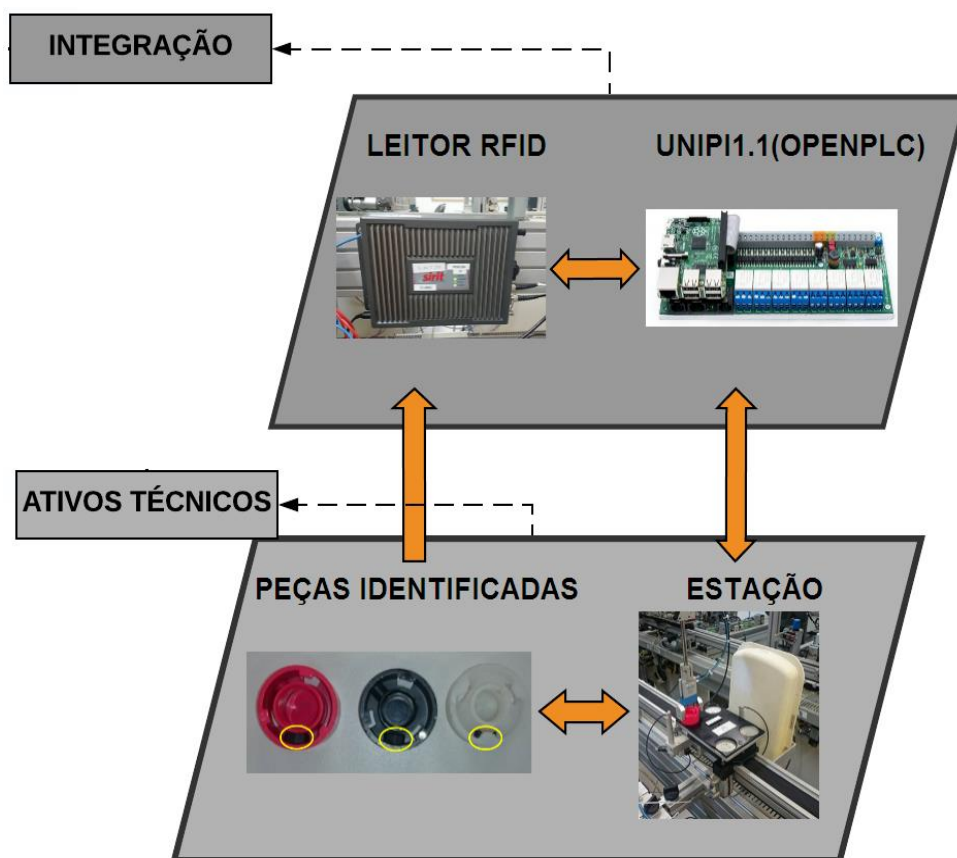
Essa relação destaca pelo paragrafo anterior, foi validada em duas etapas:

- A- Validação da conexão entre as camadas de Ativos Técnicos e Integração;
- B- Validação da conexão entre as camadas de Integração e Comunicação.

8.1 Validação da conexão entre os Ativos Técnicos e a Integração

A validação da conexão entre as camadas de Ativos Técnicos e Integração consiste a partir do funcionamento da API desenvolvida para que o OpenPLC fosse capaz de identificar as peças submetidas na estação. Além de preparar a estação para a realização dos testes, foi necessário implementar um processo controlado, sendo esta uma etapa a ser executada por parte do usuário, durante a inicialização do sistema. Sendo assim, a Figura 33 representa os principais elementos físicos, distribuídos das respectivas camadas do RAMI 4.0 para a validação do desenvolvimento realizado.

Figura 33 - Elementos contidos nas camadas de integração e ativos técnicos



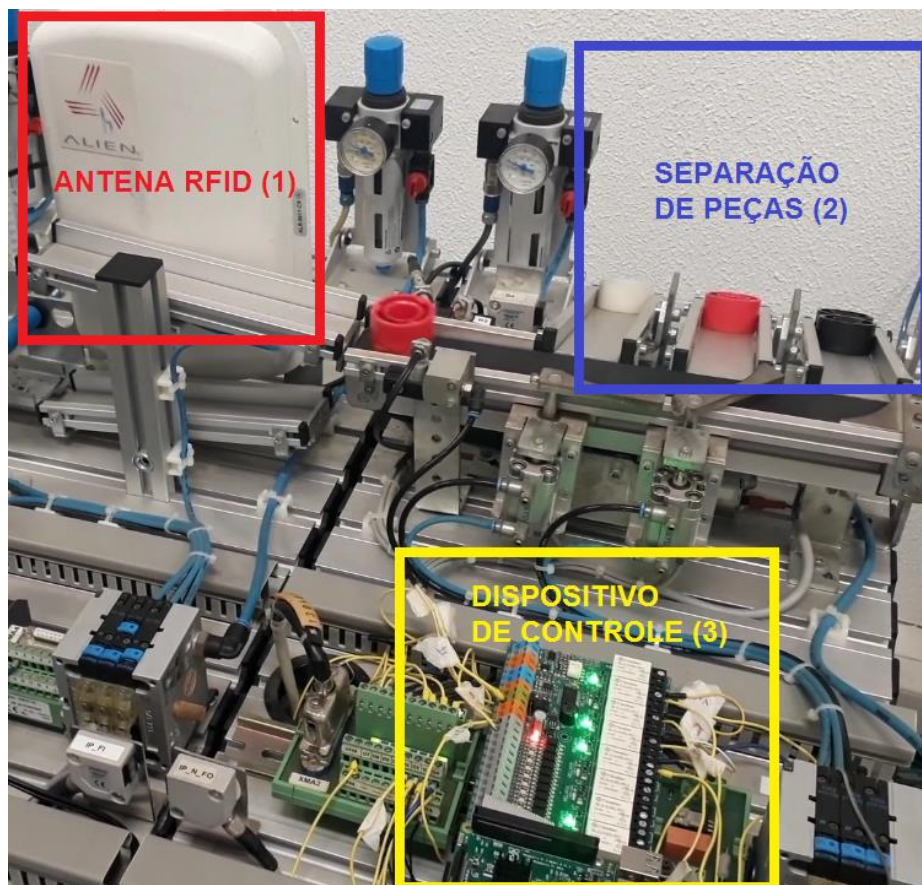
Fonte: Elaborado pelo autor

8.1.1 Estação Sorting

Sendo parte do conjunto Festo MPS 200 e encontrada na camada de Ativos Técnicos do RAMI 4.0, a Estação Sorting (Figura 34) é responsável por realizar a separação das peças por tipos em três bandejas distintas. O Dispositivo de Controle controla a operação da estação de acordo com os requisitos da aplicação. Dessa forma, as peças (cores branca, vermelha e preta) naquela estação são identificadas usando o sistema RFID que possui a antena presa ao cinto da estação (1). De acordo com a peça identificada (cor), o Dispositivo de Controle controla a estação para classificar as peças em diferentes posições (2). É importante notar que, em sua configuração original, a estação de separação possui sensores específicos e um CLP responsável pela interface de E/S e programação da lógica que controla a operação da estação. Entretanto, este equipamento não foi utilizado neste estudo, pois o Dispositivo de Controle (3) foi responsável por gerenciar todas as funções dentro da operação da estação e pela classificação da peça usando o sistema RFID. Nesta

experiência, os respectivos endereços para as informações de TagID, TagData, Antena e Alavancas Eletropneumáticas de cada peça, obtidas no servidor OPC UA foram: Parte branca (% IW2,% IW3,% QX0.0 e% QX0.6), Parte vermelha (% IW4,% IW5,% QX0.1 e% QX0.5), peça preta (% IW6,% IW7 e% QX0.2), antena (% IW8).

Figura 34 - Estação Sorting com o ambiente de testes para o Dispositivo de Controle.



Fonte: Elaborado pelo autor

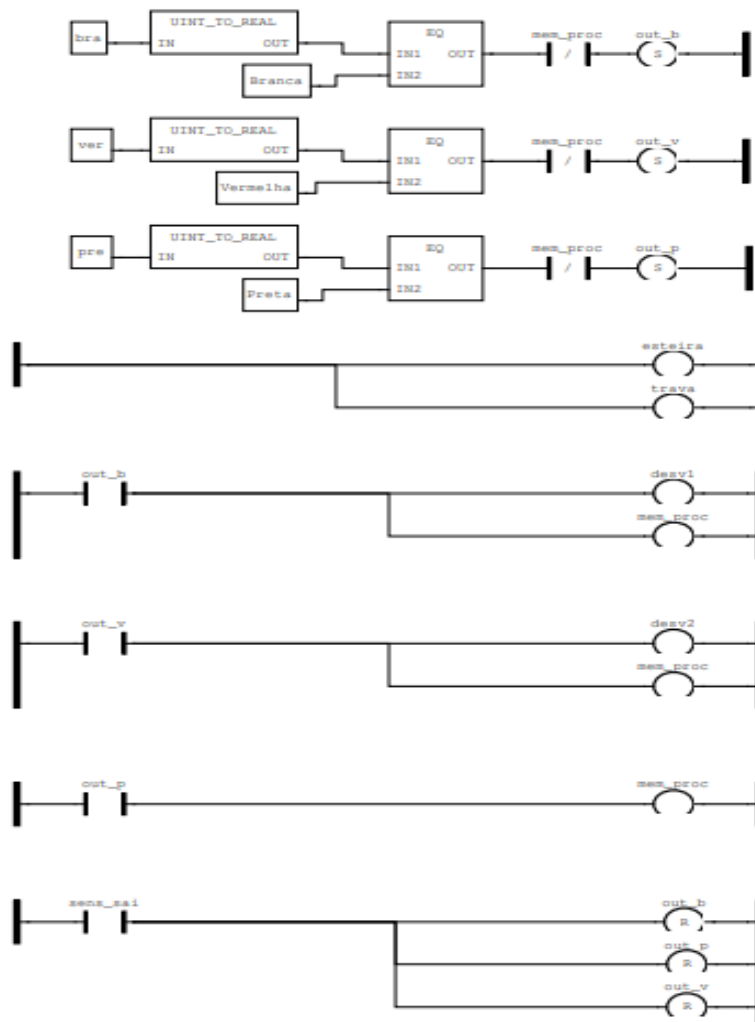
8.1.2 Processo de Manufatura

Para realizar a programação em ladder de um processo que valida o Dispositivo de Controle desenvolvido, utilizou-se o software PLCOpen Editor (ferramenta para criar programas compatíveis com o OpenPLC).

Em relação ao programa desenvolvido, toda vez que uma peça cadastrada é identificada pelo sistema RFID, o OpenPLC é capaz de obter tal informação e executar um determinado bloco de instruções para a peça identificada. Para a estação Sorting, no instante em que uma peça, como por exemplo, a branca, passa pelo leitor RFID,

uma das alavancas eletropneumáticas é acionada para que a peça seja direcionada para o local correto. O processo desenvolvido replica essa funcionalidade para diferentes peças, usando diferentes alavancas da estação. A Figura 35 a seguir ilustra o diagrama em ladder para validar o sistema RFID funcionando a partir do OpenPLC.

Figura 35 - Programa ladder desenvolvido para validação do Dispositivo de Controle.

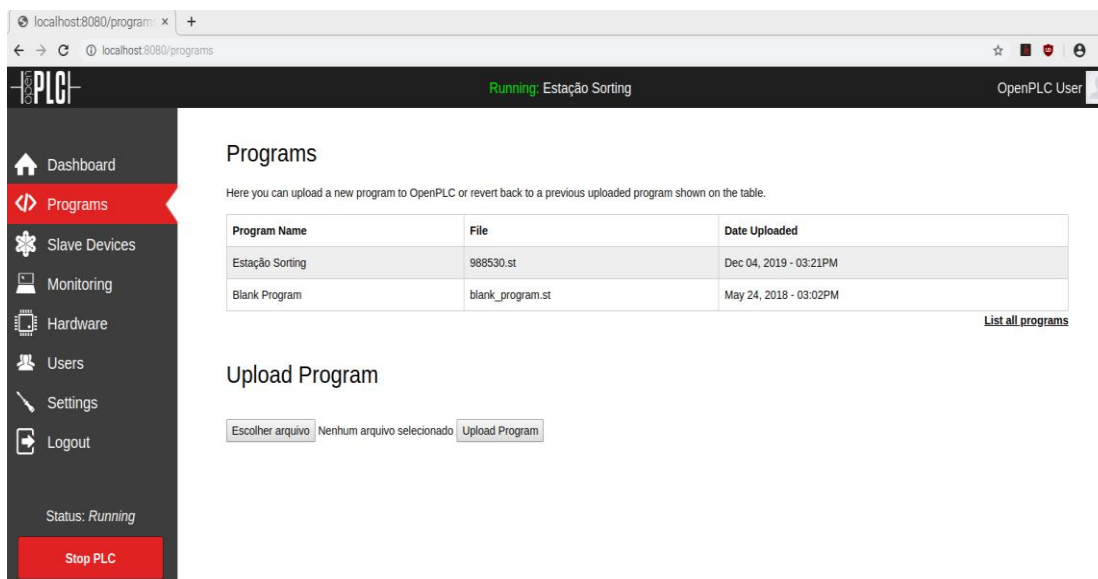


Fonte: Elaborado pelo autor

8.1.3 Controle do processo pelo OpenPLC

Depois de elaborada a lógica em ladder, tornou-se necessário fazer o upload do arquivo gerado (formato .st) após a compilação da lógica (1). Para isso, o servidor do OpenPLC deverá estar em execução e atuando sobre a porta http 8080. Dessa forma, o usuário será capaz de navegar até o item “*Programs*” e então realizar o upload do arquivo (2), conforme demonstra a Figura 36.

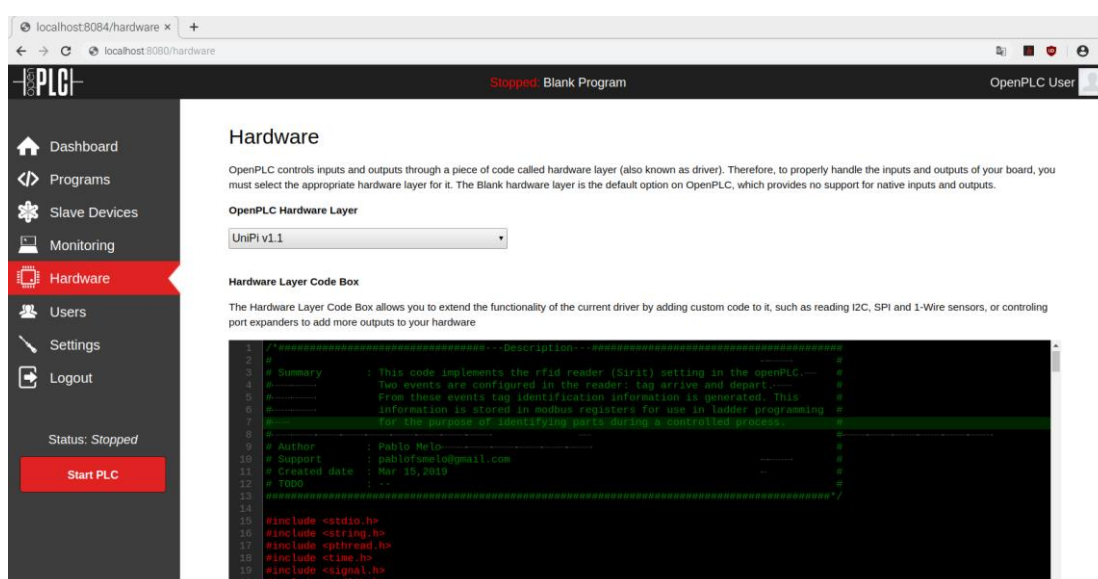
Figura 36 - Upload do programa no servidor do OpenPLC



Fonte: Elaborado pelo autor

A próxima etapa consiste em compilar o driver do sistema RFID junto ao core do OpenPLC. Para isso, o projeto OpenPLC disponibiliza através do seu servidor web uma opção para criar camadas de hardwares customizadas. Sendo assim, através do ícone de “*Hardware*” apresentado pela Figura 37, foi possível inserir o código da API RFID (1) e compilá-la ao *core* do OpenPLC. É importante ressaltar que para o Dispositivo de Controle é necessário selecionar em “*Open Hardware Layer*” a opção “UniPi v1.1” (2).

Figura 37 - Seleção da UniPi1.1 e compilação do driver referente ao sistema RFID



Fonte: Elaborado pelo autor

Assim que o código é compilado, torna-se necessário executar o programa desenvolvido. Com o programa em execução, foram submetidas peças brancas, vermelhas e pretas e criados logs para identificação dessas peças no driver desenvolvido. A Figura 38 relata os respectivos logs para a identificação e remoção dessas peças durante a execução do OpenPLC.

Figura 38 - Validação da Identificação de Peças pelo sistema RFID no OpenPLC

```
OpenPLC Runtime starting...
Interactive Server: Listening on port 43628
Interactive Server: waiting for new client...
str = event.tag.arrive tag_id=0xA111
Registrando nova peça...
Peça branca adicionada com sucesso !

str = event.tag.arrive tag_id=0xB222
Registrando nova peça...
Peça vermelha adicionada com sucesso !

str = event.tag.arrive tag_id=0xC333
Registrando nova peça...
Peça preta adicionada com sucesso !

str = event.tag.depart tag_id=0xA111
Removendo nova peça...
Peça branca removida com sucesso !

str = event.tag.depart tag_id=0xB222
Removendo nova peça...
Peça vermelha removida com sucesso !

str = event.tag.depart tag_id=0xC333
Removendo nova peça...
Peça preta removida com sucesso !
```

Fonte: Elaborado pelo autor

8.2 Validação da conexão entre a Integração e a Comunicação

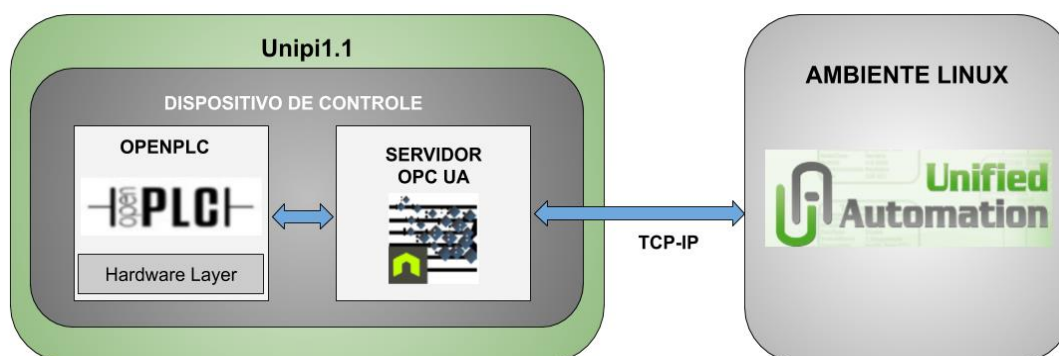
A validação da relação entre as camadas de Integração e Comunicação do RAMI 4.0 para a arquitetura proposta é fundamentada a partir do funcionamento do driver desenvolvido com o propósito de integrar o OpenPLC ao Servidor OPC UA da

aplicação. Além disso, nos testes realizados com base nessa integração, avaliaram-se particularidades do servidor OPC UA, conectando um ou mais clientes, sendo estes de diferentes tipos e plataformas. Foram utilizados nessas conexões os níveis e políticas de seguranças suportados pela pilha e sdk do node-opcua.

8.2.1 Execução do Servidor OPC UA e conexão com o UAExpert

Com o servidor do OpenPLC em execução, foi possível conectá-lo ao servidor OPC UA desenvolvido (a partir do driver relacionado no tópico 7.1.1). Diante dessa conexão estabelecida, o servidor OPC UA estava apto a fornecer os dados referentes ao processo para os clientes OPC UA. Para validar esse conjunto de dados do OpenPLC, foi utilizado num primeiro momento apenas um cliente, sendo o UaExpert. Este cliente por sua vez, implementa todas as funcionalidades, sendo também multiplataforma e desenvolvido em C++. O UaExpert está disponível para instalação em ambiente Windows e Linux. A Figura 39 aponta a estrutura a fim de avaliar a migração dos dados do OpenPLC para o OPC UA.

Figura 39 - Estrutura para validar Integração entre OpenPLC e OPC UA

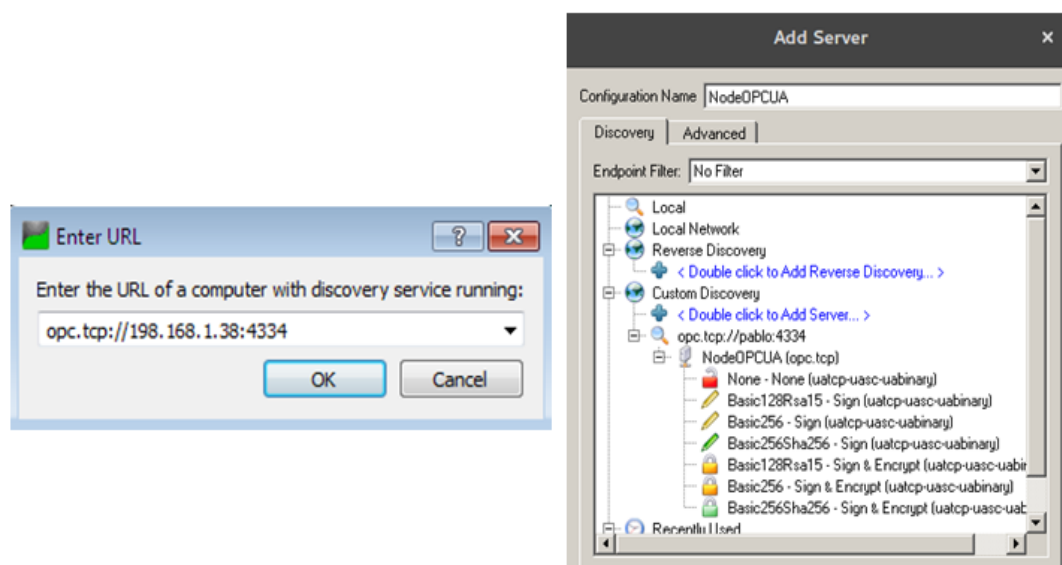


Fonte: Elaborado pelo autor

Para se conectar ao servidor OPC UA da aplicação, o cliente OPC UA UaExpert possui um ícone para adicionar um servidor. Após isso, é solicitada a URL para a descoberta do servidor. Neste caso é necessário inserir o URL do servidor contido na Unipi1.1, já que cada servidor precisa fornecer um ponto de extremidade de descoberta. A Figura demonstra a inserção da URL para conexão com o servidor. Posteriormente, a URL é adicionada à lista e, ao expandir a árvore, o servidor e todos os pontos os tipos de configurações de seguranças são mostrados. Com isso o usuário pode escolher o tipo de segurança desejado para a conexão entre cliente e

servidor. A Figura 40 a seguir ilustra as configurações iniciais para a conexão entre cliente e servidor.

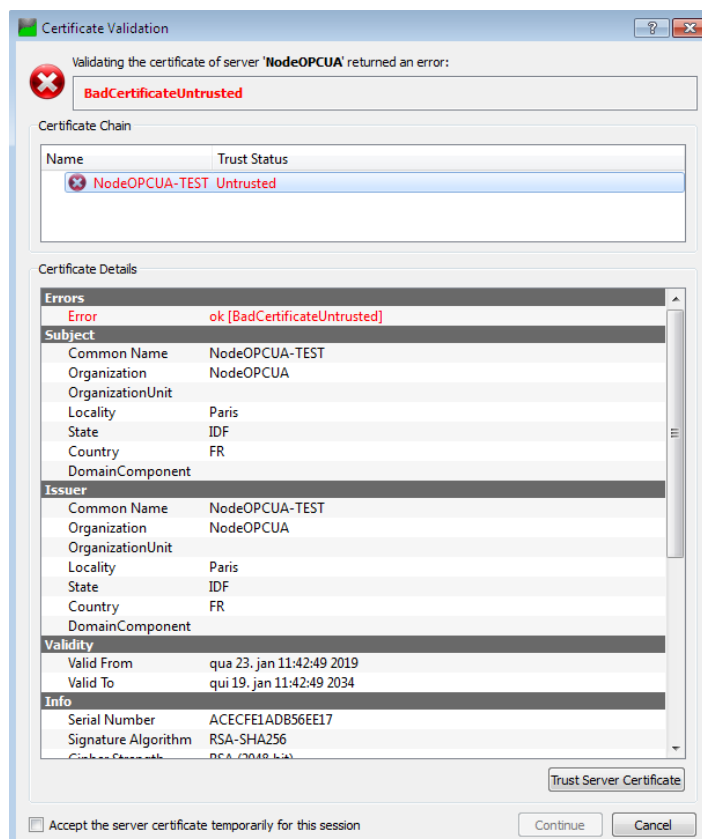
Figura 40 - Etapas de configurações para a comunicação com o servidor



Fonte: Elaborado pelo autor

Após selecionado o tipo de segurança desejado, o servidor estará listado na janela *Project*. Essa janela exibe todas as conexões e visualizações do servidor. Nessa janela podem ser adicionados e removidos servidores, conectar-se ou desconectar-se de um servidor, editar propriedades de conexão e alterar as configurações de autenticações. Antes de ocorrer efetivamente à conexão entre cliente e servidor, uma nova janela de diálogo para validar o certificado do servidor foi exposta. Depois de examinar o certificado, o usuário precisa escolher a opção “*Trust Server Certificate*” para adicionar permanentemente o certificado à lista de confiança do cliente UaExpert. Vale apontar que também é possível selecionar a opção de aceitar o certificado do servidor temporariamente para a sessão em questão e escolher continuar para não salvar o certificado na lista de confiança. A Figura 41 demonstra a tela de validação do certificado.

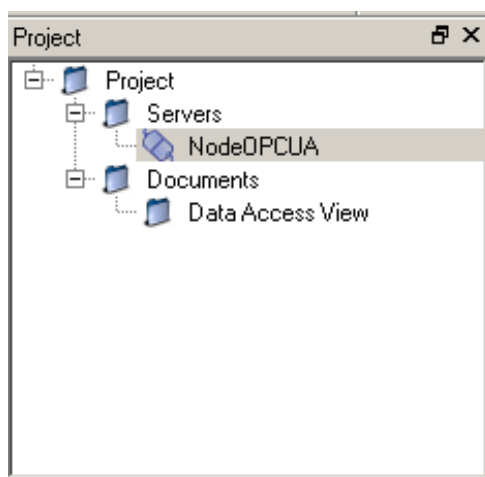
Figura 41 - Tela de validação do certificado



Fonte: Elaborado pelo autor

Uma vez que o certificado foi validado, ocorreu então a conexão do cliente UaExpert ao Servidor OPC UA da aplicação, conforme demonstra a Figura 42.

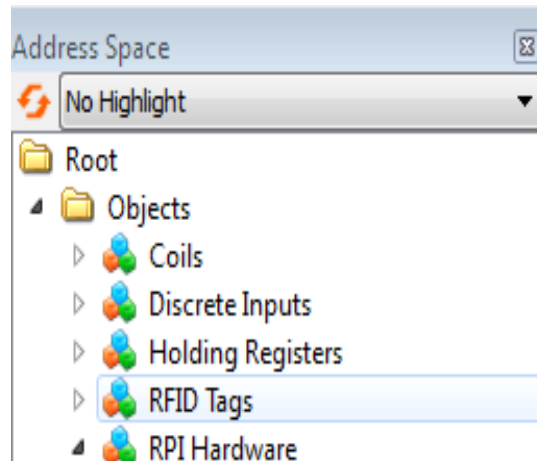
Figura 42 - Tela representando conexão estabelecida.



Fonte: Elaborado pelo autor

Após isso, os objetos criados estavam disponíveis na janela que representa o espaço de endereço. Segundo relatado na proposta, o espaço de endereço contém cinco objetos principais que são: *Coils*, *Discrete Inputs* e *Holding Registers*, *RFID Tags* e *RPI Hardware*. A Figura 43 apresenta esses objetos e suas variáveis criadas no espaço de endereço da aplicação.

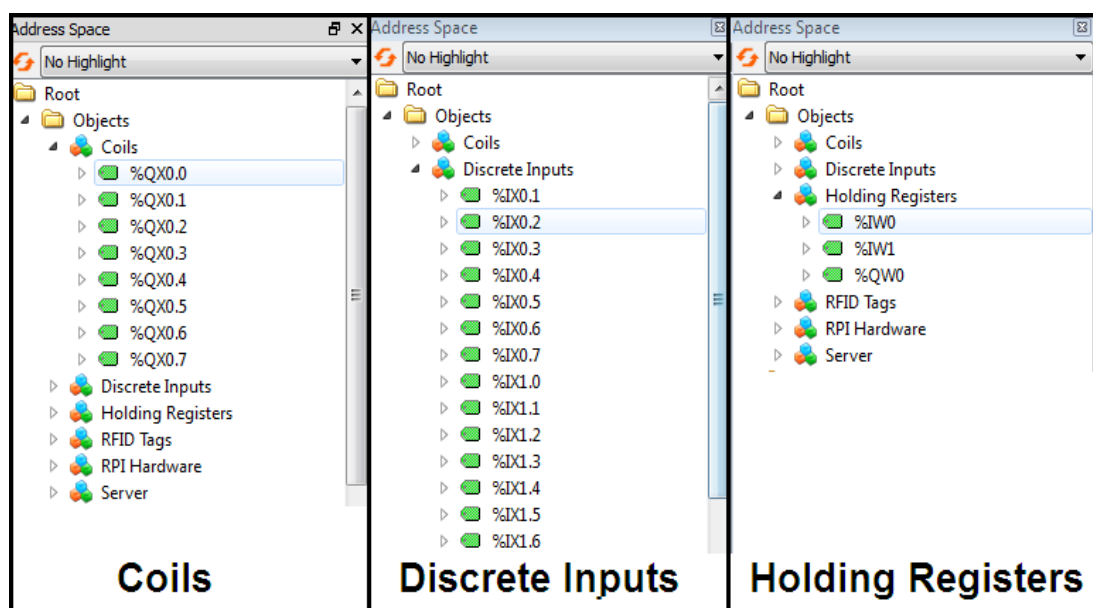
Figura 43 - Validação dos Objetos no Espaço de Endereço



Fonte: Elaborado pelo autor

É possível ainda visualizar as respectivas variáveis criadas a partir de cada objeto criado. A Figura 44 traz os registradores utilizados para representar as entradas e saídas (digitais e analógicas) do OpenPLC.

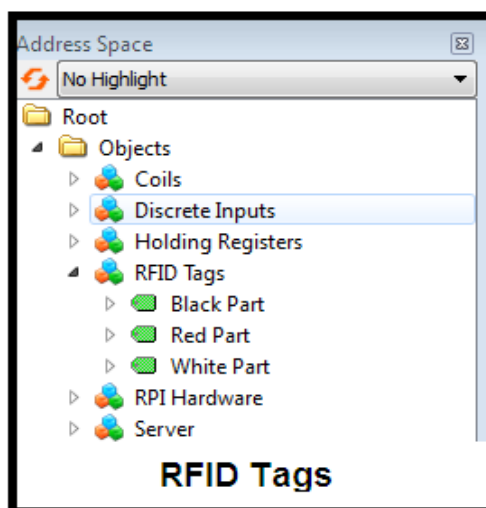
Figura 44 – Validação do Espaço de Endereço para variáveis do OpenPLC.



Fonte: Elaborado pelo autor

Mais um objeto foi adicionado ao servidor com a finalidade de exibir quais peças estavam sendo identificadas pelo sistema RFID durante a execução do processo. A Figura 45 retrata as variáveis criadas para retratar as peças identificadas.

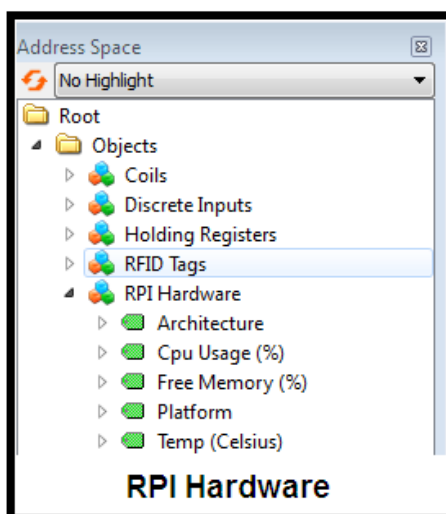
Figura 45 – Validação das variáveis referentes a identificação de peças pelo sistema RFID.



Fonte: Elaborado pelo autor

O quinto e último objeto a ser validado no espaço de endereço diz respeito a alguns elementos de hardware a serem avaliados durante a execução do processo. A Figura 46 ilustra as variáveis criadas referentes às informações disponibilizadas pelo dispositivo host.

Figura 46 – Validação das variáveis referentes ao hardware durante a execução do processo controlado.



Fonte: Elaborado pelo autor

Depois de ocorrer a navegação por todos os elementos criados no espaço de endereço, o próximo passo foi validar o conteúdo dos Atributos criados. Como citado no tópico que descreve essa proposta, foram editados alguns atributos como o *BrowseName*, *DisplayName* e *Description*. O atributo referente ao *NodeId*, não foi adicionado e pôde-se certificar que o servidor ficou encarregado de atribuir um valor de identificação para este atributo. Para representar os entradas e saídas digitais do OpenPLC e as identificação de peças, os *dataTypes* definidos foram do tipo *Boolean*. Com relação as variáveis criadas no *folder* do *Holding Registers*, todos os *dataTypes* utilizados foram o *Double*. Já para o objeto referente o hardware, utilizou-se o *Double* e *String*. Com relação aos valores, podem ser visualizados o *SourceTimeStamp*, *SourcePicoseconds*, *ServerTimeStamp*, *StatusCode* e *Value*. O *sourceTimeStamp* por exemplo é usado para refletir o registro de data e hora que foi aplicado a um valor de variável pela origem de dados. Enquanto o *ServerTimeStamp* representa a hora em que o servidor recebeu um valor de variável. O *Value* representa o valor a ser aplicado no registrador. A Figura 47 demonstra o comportamento dos Atributos para um determinado *Coil*.

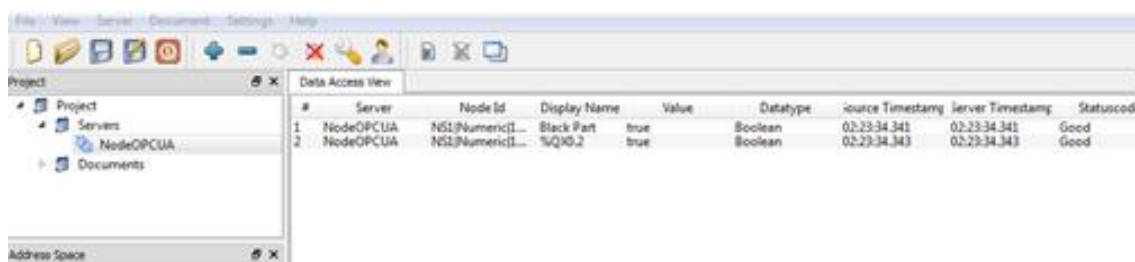
Figura 47 - Validação dos Atributos no Espaço de Endereço

Attribute	Value
NodeId	ns=1, i=1017
NamespaceIndex	1
IdentifierType	Numeric
Identifier	1017
NodeClass	Variable
BrowseName	1, "%QX0.0"
DisplayName	"", "%QX0.0"
Description	"", "Value Read Coil"
WriteMask	0
UserWriteMask	0
RolePermissions	BadAttributeIdInvalid (0x80350000)
UserRolePermissions	BadAttributeIdInvalid (0x80350000)
AccessRestrictions	BadAttributeIdInvalid (0x80350000)
Value	
SourceTimeStamp	26/1/2019 14:11:12.561
SourcePicoseconds	4300
ServerTimeStamp	26/1/2019 14:11:12.561
ServerPicoseconds	4300
StatusCode	Good (0x00000000)
Value	false
DataType	Boolean
NamespaceIndex	0
IdentifierType	Numeric
Identifier	1 [Boolean]
ValueRank	-1
ArrayDimensions	UInt32 Array[0]
AccessLevel	CurrentRead, CurrentWrite
UserAccessLevel	CurrentRead, CurrentWrite
AccessLevelEx	BadAttributeIdInvalid (0x80350000)
MinimumSamplingInterval	0
Historizing	false

Fonte: Elaborado pelo autor

Com esse Cliente foi possível também deslocar as variáveis (criadas dentro desses objetos) para uma janela de visualização de acesso a dados que pôde ser usada para monitorar mudanças do atributo de valor das variáveis, conforme demonstrado na Figura 48. A título de exemplo foi selecionada a variável referente à peça preta e à respectiva saída digital acionada quando tal peça é identificada no sistema.

Figura 48 – Assinatura criada para a identificação da peça preta e sua respectiva saída digital.

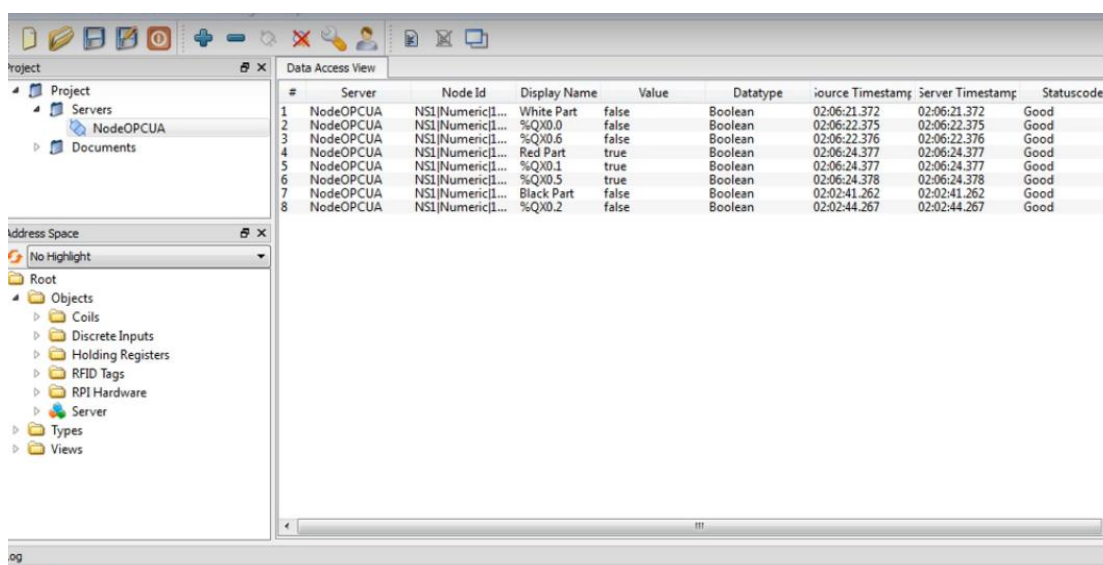


#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	StatusCode
1	NodeOPCUA	NS1/NumericI...	Black Part	true	Boolean	02:23:34.341	02:23:34.341	Good
2	NodeOPCUA	NS1/NumericI...	%QX0.2	true	Boolean	02:23:34.343	02:23:34.343	Good

Fonte: Elaborado pelo autor

Durante a validação, criaram-se também as assinaturas para todas as variáveis referentes ao processo controlado. Dessa maneira, o usuário poderá ter uma visão geral de todos os estados usando apenas uma única tela de visualização, que emprega os principais atributos de um determinado Nó do espaço de endereço. A Figura 49 constata o monitoramento de todas as variáveis do processo controlado durante sua execução.

Figura 49 – Monitoramento de todas as variáveis do processo controlado.



#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	StatusCode
1	NodeOPCUA	NS1/NumericI...	White Part	false	Boolean	02:06:21.372	02:06:21.372	Good
2	NodeOPCUA	NS1/NumericI...	%QX0.0	false	Boolean	02:06:22.375	02:06:22.375	Good
3	NodeOPCUA	NS1/NumericI...	%QX0.6	false	Boolean	02:06:22.376	02:06:22.376	Good
4	NodeOPCUA	NS1/NumericI...	Red Part	true	Boolean	02:06:24.377	02:06:24.377	Good
5	NodeOPCUA	NS1/NumericI...	%QX0.1	true	Boolean	02:06:24.377	02:06:24.377	Good
6	NodeOPCUA	NS1/NumericI...	%QX0.5	true	Boolean	02:06:24.378	02:06:24.378	Good
7	NodeOPCUA	NS1/NumericI...	Black Part	false	Boolean	02:02:41.262	02:02:41.262	Good
8	NodeOPCUA	NS1/NumericI...	%QX0.2	false	Boolean	02:02:44.267	02:02:44.267	Good

Fonte: Elaborado pelo autor

8.2.2 Servidor OPC UA conectando com diferente tipos de clientes

Uma vez estabelecida a conexão de um cliente OPC UA com o servidor OPC UA do Dispositivo de Controle, foram realizados novos testes conectando diferentes tipos de clientes durante a execução de um processo controlado. Os dados operacionais adquiridos durante o experimento foram coletados para avaliar o Dispositivo de Controle durante o experimento. Considerando a natureza da operação do Dispositivo de Controle nos cenários I4.0, várias conexões dos clientes OPC UA foram estabelecidas. Essas conexões com o servidor OPC UA do Dispositivo de Controle foram feitas durante sua operação para o controle da estação de classificação, conforme descrito. Até cinco clientes OPC UA foram conectados simultaneamente ao Dispositivo de Controle usando a conexão Wi-Fi. A Figura 50 mostra detalhes do teste, com a conexão de cinco clientes (numerados de 1 a 5) com o servidor OPC UA do Dispositivo de Controle (A).

Figura 50 – Clientes OPC UA (númerados de 1 a 5) conectados com o servidor OPC UA (A)



Fonte: Elaborado pelo autor

Cada conexão do cliente com o servidor OPC UA do dispositivo de controle foi estabelecida a cada dois minutos. Cada conexão usou uma política e um tipo de política de segurança OPC UA diferentes para verificar a funcionalidade do Dispositivo de Controle e avaliar o impacto em sua operação. A Tabela 16 detalha a sequência de conexão, o tipo de cliente OPC UA, o modo de segurança, a política de segurança e o tempo de conexão com o servidor Control Device.

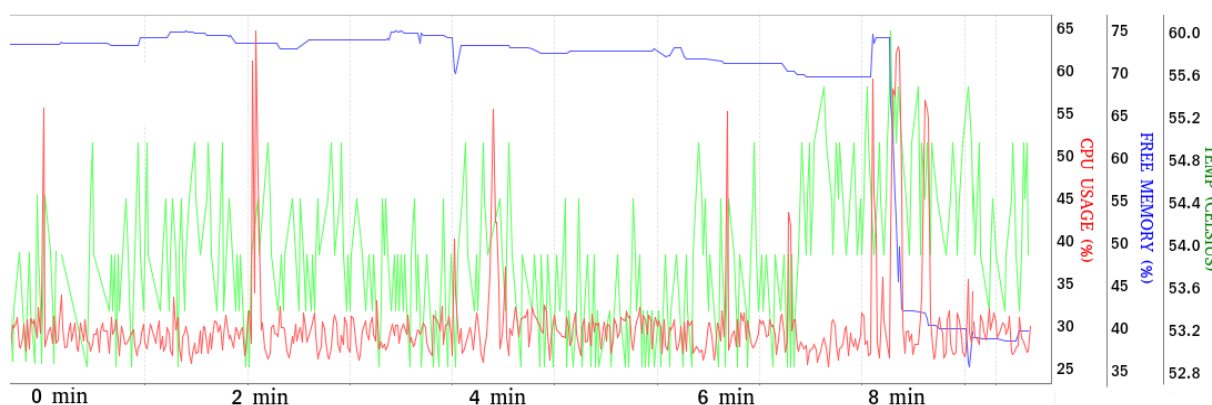
Tabela 16 - Sequência de conexão dos clientes com o servidor OPC UA.

ID	Cliente OPC UA	Modo de Segurança	Política de Segurança	Instante da Conexão (min)
1	Proslys Software	None	None	0
2	Proslys Software	Sign	Basic128Rsa15	2
3	UA Exepert Software	Sign	Basic256	4
4	OPCUA Client Mobile app	Sign & Encrypt	Basic256Sha256	6
5	IHM (cliente opcua interno)	None	None	8

Fonte: Elaborado pelo autor

Vale ressaltar que os cinco clientes foram conectados ao Dispositivo de Controle para receber os valores de três variáveis do objeto “RPI Hardware”: uso da CPU (%), memória livre (%) e temperatura (°C). O gráfico na Figura 51 mostra alterações nesses dados adquiridos do Dispositivo de Controle durante o experimento e a conexão dos usuários (clientes).

Figura 51 - Dados operacionais durante o experimento com várias conexões de usuário (cliente) para o Dispositivo de Controle (servidor)



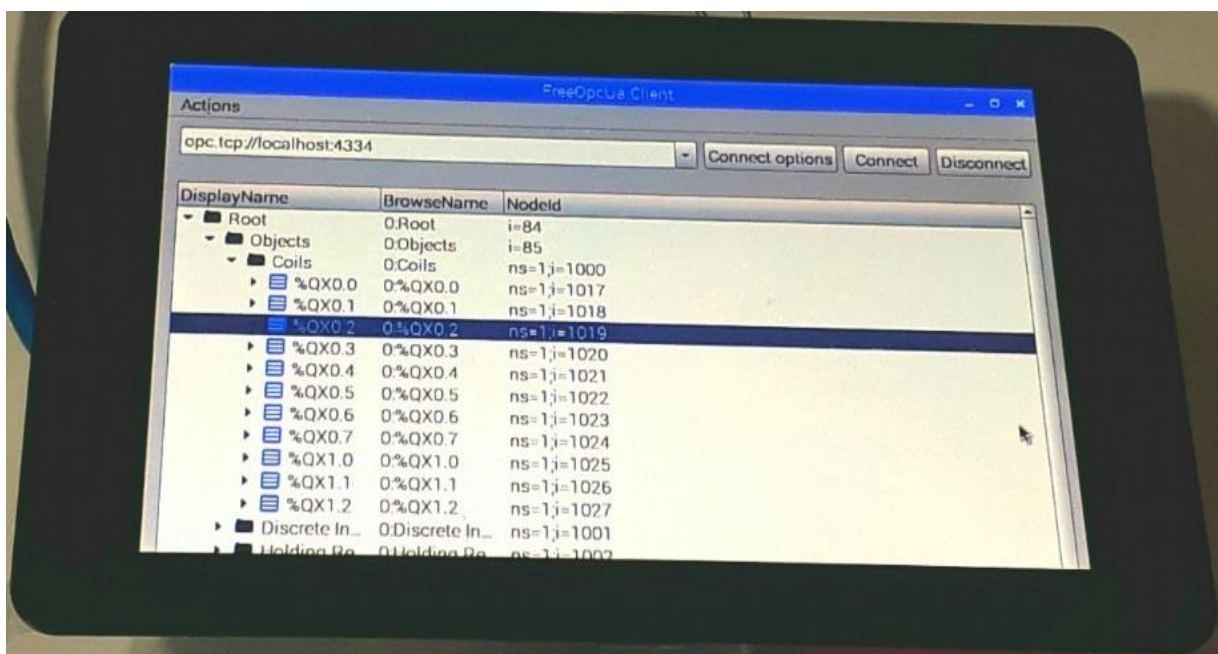
Fonte: Elaborado pelo autor

A operação do Dispositivo de Controle foi robusta, sem interrupção ou falha, nem problemas de comunicação ou desconexão do OPC UA durante os experimentos. Os resultados na Figura 51 mostram o uso normal de hardware do Dispositivo de Controle

durante o experimento com o controle da estação Festo e com as conexões de vários clientes. O uso da CPU foi de aproximadamente 30 a 35%. Alguns picos no uso da CPU ocorreram quando cada cliente se conectou ao servidor do Dispositivo de Controle. Não houve mudanças significativas de temperatura durante a operação do Dispositivo de Controle. A porcentagem de memória livre do sistema foi aceitável, com diminuição relevante com a execução e conexão do cliente IHM interno (ID -5). Era esperado, pois a execução do cliente IHM interno consome recursos do hardware do Dispositivo de Controle. Além disso, os resultados também mostram que o uso de diferentes tipos e políticas de segurança suportadas pelo Dispositivo de Controle não afeta significativamente seu desempenho.

É importante salientar que para o quinto cliente foi utilizado um monitor de computador via conexão HDMI com a Unipi1.1, conforme relatado na figura mas também avaliou-se sua usabilidade a partir de um display touch screen para *Raspberry Pi*. A solução via *display touch screen* se torna perfeita para ambientes onde seja difícil a instalação de uma tela através da conexão HDMI, garantindo a instalação do *display touch* próximo a UniPI1.1. A Figura 52 retrata o uso do quinto cliente (opcua-client-gui) sendo utilizado num *display touch screen*.

Figura 52 - Cliente OPC UA implementando uma Interface Gráfica com o Usuário.



Fonte: Elaborado pelo autor

8.3 Discussões e Observações Finais

O RAMI 4.0 fornece uma maneira comum de descrever as interações complexas e os requisitos fundamentais dos sistemas compatíveis com a I4.0. O grande desafio no desenvolvimento de uma solução compatível com RAMI 4.0 é a compreensão dos relacionamentos necessários entre tecnologias, equipamentos e sistemas relacionados aos eixos RAMI 4.0 para cada tipo de aplicativo I4.0. Como o RAMI 4.0 é abstrato, diferentes perspectivas e interpretações podem ser possíveis de acordo com a solução pretendida. O Dispositivo de Controle desenvolvido foi concebido como uma prova de conceito, com foco na integração de soluções de código aberto e abrangendo os relacionamentos entre três camadas do RAMI 4.0: Ativos, Integração e Comunicação. Embora o Dispositivo de Controle ainda não preencha os requisitos para ser um componente I4.0 (I4.0C), ele é compatível com aplicativos I4.0 baseados no RAMI 4.0.

Fuchs et.al (2019) descrevem que um I4.0C é a combinação de objetos do mundo físico e do mundo digital. O mundo físico é composto por elementos das camadas de Ativos e Integrações do RAMI 4.0 (Fig.2). O mundo digital é composto pelas camadas superiores, desde a Comunicação até o Negócio, dependendo do elemento. Considerando o elemento Dispositivo de Controle do RAMI 4.0, o mundo digital incluiria interações até a camada Informações. A maioria das funcionalidades necessárias relacionadas às camadas de Ativos e Integração (mundo físico) e à Camada de Comunicação (mundo digital) foram incluídas no Dispositivo de Controle desenvolvido. No entanto, a camada de informação, que é uma parte importante do mundo digital, ainda não estava completamente incluída. Essa camada é responsável por gerenciar e estruturar todos os dados das camadas inferiores em um modelo de informações padronizado comum a todos os componentes (I4.0C) de um aplicativo I4.0.

No entanto, analogias entre a estrutura de informações do Dispositivo de Controle desenvolvido com o Shell de Administração (AS) do I4.0C podem ser citadas (FUCHS et.al, 2019). O espaço de endereço do OPC UA do Dispositivo de Controle contém a definição de objetos físicos que são acessados por diferentes tipos de clientes. Esses objetos são basicamente modelados em termos de visualizações, atributos, variáveis, métodos e relacionamentos com outros objetos. Portanto, ambas as metodologias têm significado estrutural na especificação de conceitos, como Ativos

x Nós; Propriedade de Ativos x Atributos e Variáveis de Objetos, Submodelos x Objeto; Gerenciador de Recursos x Espaço de Endereço.

Portanto, seria possível criar um modelo de informações padrão para o Dispositivo de Controle no futuro, atendendo aos requisitos de um I4.0C. Como um I4.0C, o Dispositivo de Controle teria submodelos aplicáveis a cada ativo ou tipo de ativo (YE; HONG, 2019). Os autores ainda propõem um Shell de Administração de Ativos com um submodelo genérico composto por um item de dados indexados, declaração de valor de propriedade, documentação e comunicação. Uma proposta futura para o Dispositivo de Controle seria um modelo padrão com essas informações adicionais dos submodelos de ativos, como manuais, parâmetros de configuração e comunicação, histórico de manutenção, perfil funcional, etc. Como resultado, todas as informações relacionadas ao Controle, o ambiente do dispositivo estaria disponível para outro I4.0C conectado ao mesmo aplicativo I4.0.

9 CONCLUSÕES

Este trabalho apresentou o desenvolvimento de um Dispositivo de Controle de código aberto para I4.0 baseado no RAMI 4.0 e no OPC UA. O Dispositivo de Controle compreendia elementos das camadas Comunicação, Integração e Ativos do RAMI 4.0. Esse desenvolvimento serve como prova de conceito de um Dispositivo de Controle compatível com RAMI 4.0 e pode ser adaptado para um sistema de fabricação que visa atender aos requisitos das aplicações I4.0.

A arquitetura do Dispositivo de Controle é modular e integra diferentes componentes para atender aos requisitos de desenvolvimento das camadas incluídas do RAMI 4.0. O projeto OpenPLC foi usado na camada Integração para controle lógico compatível com os processos de fabricação da IEC 61131-3. Um suporte adicional para identificação de peças usando RFID necessário na camada de ativos foi incluído usando a camada de abstração de hardware do projeto. O node-opcua foi utilizado para a implementação do protocolo OPC UA, que é o protocolo recomendado pelo RAMI 4.0 para a camada Comunicação. O Espaço de Endereço desenvolvido do Dispositivo de Controle se relaciona com o Shell de Administração do RAMI 4.0 e fornece informações relacionadas ao processo controlado de maneira padronizada e totalmente interoperável.

Para atender aos itens propostos, este projeto enfatizou o uso dos principais elementos presentes no protocolo OPC UA pois este é um padrão bastante importante dentro do contexto industrial atual, pois ele integra as camadas que vão desde o chão de fábrica até os sistemas corporativos, e também é o único padrão definido pelo RAMI 4.0 para implementar a camada de comunicação. É pertinente acentuar que grande parte da evolução do OPC UA ocorre devido à grande influência do OPC Clássico dentro da esfera industrial, o que vem acarretando na utilização deste protocolo pelos principais fornecedores de soluções para automação, durante a construção de suas aplicações. Com isso, este trabalho apresentou uma revisão do protocolo OPC UA e seu histórico de desenvolvimento, bem como um detalhamento das funcionalidades básicas, mapeamento de tecnologias e serviços. Um levantamento de características, linguagens de programação, tecnologias de comunicação, políticas de segurança e conjuntos de serviços das principais implementações de código aberto do OPC UA, que contemplam aplicativos de clientes e servidores, foi realizado. Esse levantamento permitiu a realização de uma análise

comparativa das implementações de software open-source do OPC UA, através da qual se concluiu que todas as implementações viabilizam a comunicação OPC UA e se encontram em estágio de desenvolvimento. No entanto, dentre as implementações estudadas, apenas a Eclipse Milo e UA.NET apresentam todos os requisitos estabelecidos e analisados. Porém para o desenvolvimento deste projeto, foi utilizado o projeto node-opcua, desenvolvido através do Node.Js, sendo que este é utilizado em outras tecnologias contidas no desenvolvimento desta proposta, como o jsmodbus. Além disso, o node-opcua apresenta uma comunidade ativa de desenvolvedores, oferecendo suporte para SBCs do tipo Raspberry Pi 3, atualizações de software, documentações, e exemplos de demonstração. A sistematização das informações neste trabalho permitiu criar uma documentação de referência para orientar trabalhos de pesquisa e desenvolvimento sob o protocolo OPC UA e sua aplicação em IIoT e I4.0.

Todos os componentes de software foram implantados e incorporados ao SBC (Raspberry Pi 3B +). Os recursos de hardware do Dispositivo de Controle foram avaliados em um cenário I4.0 real. Além da operação robusta e do controle lógico confiável da estação de fabricação, os resultados experimentais com várias conexões simultâneas do usuário (cliente) ao servidor do Dispositivo de Controle demonstraram pouco uso dos recursos de hardware.

Trabalhos futuros incidirão na atualização do Dispositivo de Controle em um Componente I4.0 da RAMI 4.0. Conforme discutido nas considerações finais do documento, mesmo com as atuais semelhanças entre o Espaço de Endereço do Dispositivo de Controle e a Administração da RAMI 4.0, é necessário um modelo padronizado para a camada Informações para a atualização.

REFERÊNCIAS

A VOZ DA INDÚSTRIA, **Ebook: Manufatura Avançada**, 2016.

ADOLPH, L.; ANLAHR, T.; BEDENBENDER, H.; BENTKUS, A.; BRUMBY, L.; DIEDRICH, C.; DIRZUS, D.; ELAMAS, F.; EPPLE, U.; FRIEDRICH, J.; FRITZ, J.; GEBHARDT, H.; GEILEN, J.; HECKER, C.; HEIDEL, R.; HEMBERGER, K.; HIENSCH, S.; HILGENDORF, E.; HÖRCHER, G.; KLEMM, E.; MEHRFELD, J. **German Standardization Roadmap Industry 4.0 DIN/DKE Roadmap**, 2016. Disponível em: <<http://www.din.de/en/din-and-our-partners/press/pressreleases/updated-german-standardization-roadmap-on-industry-4-0-110576>>. Acesso em: novembro, 2017.

ADOLPHS, P.; BEDENBENDER, H.; DIRZUS, D.; EHLICH, M.; EPPLE, U.; HANKEL, M.; HEIDEL, R.; HOFFMEISTER, M.; HUHLE, H.; KÄRCHER, B.; KOZIOLEK, H.; PICHLER, R.; POLLMEIER, S.; SCHEWE, F.; WALTER, A.; WASER, B.; WOLLSCHLAEGER, M. **Reference Architecture Model Industrie 4.0 (RAMI 4.0) VDI/VDE and ZVEI**, 2015. Disponível em: <<https://www.zvei.org/en/subjects/industry-4-0/the-reference-architectural-model-rami-4-0-and-the-industry-4-0-component/>>. Acesso em: novembro, 2017.

AKERMAN, M.; BERGLUND, A.F.; EKERED, S. **Interoperability for a Dynamic Assembly System**. In 6th CIRP Conference on Assembly Technologies and Systems (CATS), 2016.

ALVES, T.R. **What is OpenPLC?** OpenPLC Project, 2017. Disponível em: <<http://www.openplcproject.com>>. Acesso em novembro de 2017.

ALVES, T.R.; BURATTO, M.; SOUZA, F.M.; RODRIGUES, T, V. **OpenPLC: An Open Source Alternative to Automation**. IEEE 2014 Global Humanitarian Technology Conference (GHTC 2014), San Jose, CA, 2014, pp. 585-589.

ALVES 2019. **OpenPLC Project**. Disponível em <<http://www.openplcproject.com>>. Acesso em novembro 2018.

AKBARI, A.; MIRSHAHI, S.; HASHEMIPOUR, M. **Application of RFID system for the process control of distributed manufacturing system**. In Electrical and Computer Engineering (CCECE), 2015 IEEE 28th Canadian Conference on, vol., no., pp.497-501, 3-6 May.

ASHTON, A. **That Internet of Things**. In Thing. RFID Journal, 03 jul.2010, p.1.

ASIKAINEN, J. **OPC UA Java History Gateway with Inherent Database Integration**. 2014.69 f. Dissertação. Aalto University-School of Electrical Engineering. Espoo, 2014.

BANGEMANN, T.; BAUER, C.; BEDENBENDER, H.; DIESNER, M.; EPPLE, U.; ELMAS, F.; FRIEDRICH, J.; GOLDSCHIMDT, T.; GÖBE, F.; GRÜNER, S.; HANKEL, M.; HESSELMANN, K.; HÜTTEMANN, G.; KEHL, H.; LÖWEN, U.; PFROMMER, J.; SCHLEIPEN, M.; SCHLICH, B.; USLÄNDER, T.; WERTERKAMP, C.; WINTER, A.; WOLLSCHLAEGER, M. **Industrie 4.0 - Technical Assets: Basic terminology concepts, life cycles and administration models**, 2016. Disponível em

<<http://publica.fraunhofer.de/documents/N-432116.html>>. Acesso em: dezembro, 2017.

BAUER, W.; SCHLUND, S.; MARRENBACH, D.; GANSCHAR, O. **Industry 4.0: Volkswirtschaftliches Potenzial für Deutschland**. Tech. Rep. TR 2014-1, Fraunhofer Institute of Technology, Hansastrasse 27c, Munich (MU), 2014.

BAUERNHANSL, T.; HOMPEL, M. T.; VOGEL-HEUSER, B. **Industrie 4.0 in Produktion, Automatisierung und Logistik: Anwendung, Technologien, Migration**. Springer-Verlag, Abraham-Lincoln-Strasse 46, Germany, Wiesbaden (WIE), 2014.

BAUMANN, F. W.; FALKENTHAL, M.; HUDERTM S.; ODEFEY, U.; ZIMMERMANN, M. **Cyber-Physical System Control via Industrial Protocol OPC UA**. In Institute of Architecture of Application Systems, Universität Stuttgart, Germany, 2017.

BHATT, P.; BALDEVIA, R. P. **Integrate IEDs With OPC Technology**. In 7th Annual Western Power Delivery Automation Conference. 2005. p.1-18.

BLAZEK, P.; KREJCAR, D.; JUN.; KUKA, K. **Device Security Implementation Model based on Internet of Things for a Laboratory Environment**. IFAC-PapersOnLine, 49(25), 2016, pp. 419-424.

BURKE, T.J. **OPC Unified Architecture: Interoperability for Industrie 4.0 and the Internet of Things**. OPC Foundation, 2017.

CANTELON, M.; HARTER, M.; HOLOWAYCHUK, T. J.; RAJLICH, N. **NodeJs**. Action Manning, 2014.

CARDOSO, J.; VOIGT, K.; WINKER, M. **Service Engineering for The Internet of Services, to appear in Enterprise Information Systems**, Lecture Notes in Business Information Processing (LNBIP), 2008.

CHANTANAYINGYONG, F. M. **What is the theme of Davos 2016?** 2016. Disponível em: <<https://www.weforum.org/agenda/2015/11/what-is-the-theme-of-davos-2016/>>. Acesso em: dezembro, 2017.

CHUANG, A.C.C. **Discuss the standard of Industry 4.0**. Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Taiwan, 2016. Disponível em: http://www.ebc.nthu.edu.tw/StudentProject/eifinal/2015_eiProject/word/07.pdf. Acesso em: dezembro, 2017.

CHUANYING, Y.; HE, L.; ZHIHONG, L. **Implementation of Migrations from Class OPC to OPC UA for Data Acquisition System**. In International Conference on System Science and Engineering. 2012 IEEE International Conference on, IEEE, 2012.p.588-592.

CONTRENAS, J.D.; GARCIA, J.I.; PASTRANA, J.D. **Developing of Industry 4.0 Applications**. International Journal of Online Engineering, iJOE – Vol. 13, No. 10, 2017.

DERHAMY, H.; RONNHOLM, J.; DELSING, J.; ELIASSON, J.; DEVENTER, J.V. **Protocol interoperability of OPC UA in Service Oriented Architectures**. IEEE 15th International Conference on Industrial Informatics (INDIN), Emden, 2017.

DORST, W.; GLOHR, C.; HAHN, T.; KNAFLA, F.; LOEWEN, U.; ROSEN, R.; SCHIEMANN, T.; VOLLMAR, F.; WINTERHALTER, C.; DIEGNER, B.; DÜMMLER, M.; ERKER, S.; HERFS, W.; HILGER, C.; JÄNICKE, L.; JASPERNEITE, J.; KALHOFF, J.; KUBACH, U.; LÖWEN, U.; MATTIS, G.; MENGES, G.; MILDNER, F.; QUETSCHLICH, M.; STEFFENS, E.-J.; STIEDL, T.; ADOLPHS, P.; BEDENBENDER, H.; EHILCH, M.; EPPLE, U.; HANKEL, M.; HEIDEL, R.; HOFFMEISTER, M.; HUHLE, H.; KÄRCHER, B.; KOZIOLEK, H.; PICHLER, R.; POLLMEIER, S.; SCHEWE, F.; SCHULZ, T.; SCHWEICHHART, K.; WALTER, A.; WASER, B.; WOLLSCHLAEGER, M.; JÄNICKE, L.; JOCHEM, M.; KAISER, H.; KISCH, M.; KLASSEN, W.; LEHMANN, J.; LINKE, L.; MEHRFELD, J.; SANDNER, M. **Implementation Strategy Industrie 4.0: Report on the results of the Industrie 4.0 Platform, 2016**. Disponível em: <https://www.bitkom.org/Bitkom/Publikationen/Implementation-Strategy-Industrie-40Report-on-the-results-of-the-Industrie-40-Platform.html>. Acesso em: dezembro, 2017.

DURKOP, L.; IMTIAZ, J.; TRSEK, H. **Service-Oriented Architecture for the Autoconfiguration of Real-Time Ethernet Systems**, IEEE 11th International Conference on Industrial Informatics (INDIN), 2013.

ECLIPSE. **Eclipse Milo 0.1.0 Release Review**. Available in: <https://projects.eclipse.org/projects/iot.milo/reviews/0.1.0-release-review> Acesso em Janeiro de 2018.

ENGELHARDT, P.; REINHART, G. **Approach for an RFID-based situational shop floor control**. IEEE International Conference on Industrial Engineering and Engineering Management, p. 444–448, 2012.

ERUVANKAI, S.; MUTHUKRISHNAN, M.; MYSORE, A.K. **Accelerating IIOT Adoption with OPC UA**. In: Internetworking Indonesia Journal, v.9, n1, p.3-8, 2017.

FLEISCHMANN, H.; BROSSOG, M.; BECK, M.; FRANKE, J. **Automated Generation of Human-Machine Interfaces in Electric Drives Manufacturing**. In: Proceedings of the 7th International Electric Drives Production Conference and Exhibition 2017, Wuerzburg, Germany, IEEE, 2017.

FUCHS, J.; SCHIMDT, J.; FRANKE, J.; REHMAN, K.; SAUER, M.; KARNOUSKOS, S. **I4.0-compliant integration of assets utilizing the Asset Administration Shell**. In: 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Zaragoza, Spain, 2019, pp. 1243-1247.

GHAZIVAKILI M.; DEMARTINI, C.; ZUZINO, C. **Industrial Data-Collector by enabling OPC UA standard for Industry 4.0**. IEEE 14th International Workshop on Factory Communication Systems (WFCS), Imperia, 2018.

GOTZE, J. **Reference Architecture for Industry 4.0**. Qualiware- Center of Excellence, 2016. Disponível em: <<https://coe.qualiware.com/reference-architectures-for-industry-4-0/>>. Acesso em dezembro de 2017.

GROSSMANN, D.; BENDER, K.; DANZER, B. **OPC UA based Field Device Integration**. In SICE Annual Conference. 2008 IEEE International Conference on. p.933-938, 2008.

GRUNER, S.; PFROMMER, J.; PALM, F. **RESTful Industrial Communication with OPC UA**. Industrial Informatics, IEEE Transactions on, v.12, n.5, p.1832-1841, 2016.

GUBBI, J.; BUYYA, R.; MARUSIC, S.; PALANISWAMI, M. **Internet of Things (IoT): A vision, architectural elements, and future directions**. In Future Generation Computer Systems (FGCS), 2013 Elsevier p.1645- 1660, 2013.

HANKEL, M.; REXROTH, B.; Industrie 4.0: **The Reference Architectural Model Industrie 4.0 (RAMI 4.0)**. ZVEI - German Electrical and Electronic Manufacturers' Association, 2016. Disponível em: <https://www.zvei.org/fileadmin/user_upload/Themen/Industrie_4.0/Das_Referenzarchitekturmo-dell_RAMI_4.0_und_die_Industrie_4.0-Komponente/pdf/ZVEI-Industrie-40-RAMI-40-English.pdf>. Acesso em: dezembro de 2017.

HANNELIUS, T. **Roadmap to adopting OPC UA**. In the IEEE International Conference on Industrial Informatics, 2008 IEEE International Conference on IEEE, 2008.p 756-761.

HASKAMP, M.; MEYER, R.; MOLLMANN, F.; ORTH, F.; COLOMBO, A.W. **Benchmarking of existing OPC UA implementations for Industrie 4.0-compliant digitalization solutions**. In 2017 IEEE 15th International Conference on Industrial Informatics, Emden, Germany, pp. 589-594, 2017.

HASSANZADEH, A.; MODI, S.; MULCHANDANI, S. **Towards Effective Security Control Assignment in the Industrial Internet of Things**. 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), Milan, p. 795-800, 2015.

HELL, K.; KRISTOFER, R.; HILLMANN, A.; LUDER, H.; ROPKE, J.; ZAWISZA, J.; SCIHMDT, A. **Demands on virtual representation of physical Industrie 4.0 components**, Proceedings of the 2nd INCOSE Italia Conference on Systems Engineering, Turin, Italy, November 2016. Disponível em <<http://ceur-ws.org/Vol-1728/paper8.pdf>>. Acesso em: Novembro de 2018.

HOFFMEISTER, M. Industrie 4.0: **The Industrie 4.0 Component**. ZVEI- German Electrical and Electronic Manufactures Association, 2015.

HOPPE, S. **There is no Industrie 4.0 without OPC UA**. Disponível em: [http://opcconnect.opcfoundation.org/2017/06/there-is-no-industrie-4-0-without-OPC UA/](http://opcconnect.opcfoundation.org/2017/06/there-is-no-industrie-4-0-without-OPC-UA/). Acesso em: outubro, 2017.

HUEBL, K. Willkommen beim **ASNeG Git Repository**. Available in: <<https://81.169.197.52:8443/repositories/>>>. Acesso em: January, 2018,2017.

HUGHES-CROUCHER, T; WILSON, M. **Node: Up and running: Scalable server-side code with JavaScript**. "O'Reilly Media, Inc.", 2012.

HUIMING, L.; ZHIFENG, Y. **Research on Key Technology of the Address Space for OPC UA Server.** In 2nd International Conference on Advanced Computer Control. 2010 IEEE International Conference on. IEEE, 2010.p.278-281.

IMTIAZ, J.; JASPERNEITE, J. **Scalability of OPC UA Down to the chip level enables Internet of Things.** 11th IEEE International Conference on Industrial Informatics, Bochum, 2013.

IWANITZ, F. **OPC Fundamentals, Implementation and Application.** In 3rd ed., Huthig Verlag Heidelberg, 2006.

JAZDI, N. **Cyber physical systems in the context of Industry 4.0.** In: Automation, Quality and Testing, Robotics, 2014 IEEE International Conference on. IEEE, 2014. p. 1-4.

JOHANSSON, C.F. **Open Modelica Interactive Simulation using an OPC UA client.** Bachelor thesis, Linköping University, Department of Computer and Information Science, Sweden, 2017.

KAGERMANN, H.; WAHLSTER, W.; HELBIG, J. **Recommendations for Implementing the Strategic Initiative. Industrie 4.0 Final Report of the Industrie 4.0 Working Group.** Disponível em: <
<http://www.acatech.de/de/publikationen/stellungnahmen/kooperationen/detail/artikel/recommendations-for-implementing-the-strategic-initiative-industrie-40-final-report-of-the-industr.html> >. Acesso em: novembro, 2017.

KHAKIFIROOZ, M.; CAYARD, D.; CHIEN, C.F.; FATHI, M. **A system dynamic model for implementation of Industry 4.0.** International Conference on System Science and Engineering (ICSSE), 2018.

KOTA, A.K; PRABHU, D.M. **Node.js: Lightweight, Event driven I/O web development.** In Informatics NIC, Government of India, 2014.

KOZAR, S. **Integration of IEC 61499 with OPC UA.** In 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), Berlin, Germany, pp. 1-7, 2016.

KUMAR, R.; BOSE, A.K. **Internet of Things and OPC UA.** In the International Symposium on Advances in Content- oriented Networks and Systems, 2015 The Eleventh International Conference on Networking and Services (ICNS), 2015.p.38-44.

LANGMANN, R.; ROJAS-PEÑA, L. PLCs as Industry 4.0 Components in Laboratory Applications. International Journal of Online Engineering, iJOE, v. 12, n. 7, p. 37–44, 2016.

LEE, E.A; SESHIA, S.A. **Introduction to embedded systems: A cyber-physical systems approach,** MIT Press, 2017.

LEHNHOFF, S.; ROHJANS, M.; MAHNKE, W. **OPC Unifield Architecture- A Service Oriented Architecture for Smart Grids.** In First International Workshop on Software Engineering Challenges for the Smart Grid (SE-SmartGrids). 2012 IEEE International Conference on, IEEE, 2012.p.1-7.

LEHNHOFF, S.; MAHNKE, W.; ROHJANS, M.; USLAR, M. **IEC 61850 based OPC UA Communication - The Future of Smart Grid**. In 17th Power Systems Computation Conference, Stockholm Sweden, p.1-9, 2011.

LEITNER, S.H.; MAHNKE, W. **OPC UA – Service-oriented Architecture for Industrial Applications**. In ABB Corporate Research Center, 2006.

LUDER, A.; SCHLEIPEN, M.; SCIHMDT, N.; PFROMMER, J. **One step towards an Industry 4.0 component**. 13th Conference on Automation Science and Engineering, China, 2017.

MAHNKE, W.; LEITNER, S, H.; DAMM, M. **OPC Unified Architecture**. In Springer, 2009.

MARCON, P.; ZEZULKA, F.; BRADAC, Z.; ARM, J.; BENESL, T.; VESELY, I. **Communication Technology for Industry 4.0**. International Federation of Automatic Control (IFAC), 2018.

MARSEU, E. **Exemplary transfer of the RAMI 4.0 Administration Shell to the SmartFactory System Architecture for Industrie 4.0 Production Systems**. Smart Factory, 2017.

MOLDOVEN, A; BARNSTED, E; OURSEL, E. **OPC UA.Net Standard Stack and Samples**. Available in: <https://github.com/OPCFoundation/UA-.NETStandard>. Access Date, January, 2018, 2018.

MOSH, C. **RAMI 4.0 and Industrie 4.0 components**. Disponível em: <<https://industrie40.vdma.org/en/viewer/-/v2article/render/15557415>>. Acesso em Dezembro de 2018.

MULLER, M; WINGS, E; BERGMANN, L. **Developing open source cyber-physical systems for service-oriented architectures using OPC UA**. 2017 In 2017 IEEE 15th International Conference on Industrial Informatics (INDIN), Emden, Germany, pp. 83-88, 2017.

NSIAH, A.K; SCHAPPACHER, M; SIKORA, A. **Dynamic mapping of EDDL device descriptions to OPC UA**. Journal of Physics: Conference Series, vol. 870, no. 1, pp. 1-8, 2017.

OPC FOUNDATION. **Unified Architecture**, 2017. Disponível em: <<https://opcfoundation.org/developer-tools/specifications-unified-architecture/>>. Acesso em Agosto de 2017.

PANDA, S.K.; SCRODER, T.; WISNIEWSKI, L.; DIEDRICH, C. **Plug&Produce Integration of Components into OPC UA based data-space**. 23rd IEEE International Conference on Emerging Technologies and Factory Automation, Torino, 2018.

PALONEN, O. **Object Oriented Implementation of OPC UA Information Models in Java**. 2010. 81 f. Dissertação Aalto University – School of Electrical Engineering. Espoo 2010.

PEDONE, G.; MEZGAR, I. **Model similarity evidence and interoperability affinity in cloud-ready Industry 4.0 technologies**. Computers in Industry, Elsevier, 2018.

PEREIRA, E.M. **Plataforma Industrial de Internet das Coisas**. Universidade do Porto, 2018.

PISCHING, M. A. **Arquitetura para descoberta de equipamentos em processos de manufatura com foco na Indústria 4.0**. Tese (Doutorado em Engenharia de Controle e Automação Mecânica) Universidade de São Paulo, 2017.

PISCHING, M.A.; PESSOA, M, I.A.O.; JUNQUEIRA, F. FILHO, D.J.S.; MIYAGI, P.E. **An architecture based on RAMI 4.0 to discover equipment to process operations required by products**. Computers & Industrial Engineering Journal. 2017. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0360835217306034>>. Acesso em: Outubro de 2018.

PLESIS, D.; CARL, J. **A framework for implementing Industrie 4.0 in learning factories**. Stellenbosch University, 2017.

POSTOL, M. **OPC UA Information Model Deployment**. CAS, 2016.

PROFANTER, S.; DOROFEEV, K.L; ZOITL, A.; KNOLL, A. **OPC UA for Plug & Produce: Automatic Device Discovery using LDS-ME**. 2017 22 nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Limassol, Cyprus, 2017, pp. 1-8.

PROFANTER, S. **List of Open Source OPC UA Implementations**. Disponível em: <[https://github.com/open62541/open62541/wiki/List-of-Open-Source-OPC UA-Implementations](https://github.com/open62541/open62541/wiki/List-of-Open-Source-OPC-UA-Implementations)>. Acesso em: November, 2017.

REIMANN, J. **OPC UA solutions with Eclipse Milo**, 2017. Disponível em: <<https://dentrassi.de/2017/09/14/creating-opc-ua-solutions-eclipse-milo/>>. Acesso: Janeiro, 2018.

RELAY AUTOMATION. **OPC UA Concepts and Implementation**. Disponível em: <<https://www.relayautomation.com/category/iiot/>>. Acesso em: Dezembro de 2018.

ROHJANS, S.; USLAR, M.; APPELRATH, J. **OPC UA and CIM: Semantics for the Smart Grid**. In IEEE PES T&D.2010 IEEE Conference on, p.1-8, 2010.

ROSSIGNON, E. **Node-opcua: an implementation of an OPC UA stack fully written in JavaScript and Node.js**. Disponível em: <<https://github.com/node-opcua/node-opcua>>. Acesso em: Setembro, 2018.

RÚNARSSON, S. **Open Source Hardware and Software Alternative to Industrial PLC**. Masters of Science Dissertation, University College of Southeast Norway, Faculty of Technology, Norway, 2016.

RYKONANOV, A. **FreeOPCUA Library**. Disponível em: <<https://github.com/FreeOPCUA>> . Acesso em: Dezembro de 2018.

SANCHES, H, B. **Monitoramento da Produção e da Eficiência de Processos de Manufatura usando RFID e Internet das Coisas**. Dissertação (Mestrado em Engenharia Elétrica). Universidade Estadual Paulista. Faculdade de Engenharia, Bauru, 2018.

SANISLAV, J.; MICLEA, L. **Cyber Physical Systems- Concept, Challenges and Research Areas**. Control Engineering and Applied Informatics (CEAI), vol.14, no.2, pp. 28-33, 2012.

SAUER, O. **Developments and Trends in Shopfloor-related ICT Systems**. In IEEE International Conference on Industrial Engineering Management, 2014 IEEE International Conference on. IEEE, 2014.p.1352-1356.

SCHROTH, C; JANNER, T. **“Web 2.0 and soa: Converging concepts enabling the internet of services,”** IT professional, vol. 9, no. 3, 2007.

SCHLEIPEN, M.; GILANI, S.S.; BISCHOFF, T.; PFROMMERA, J. **OPC UA & Industrie 4.0 - enabling technology with high diversity and variability**. 49th CIRP Conference on Manufacturing Systems, Elsevier.

SCHULTE, D.; COLOMBO, W. **RAMI 4.0 based digitalization of an industrial plate extruder system: Technical and infrastructural challenges**. IECON – 43rd Annual Conference of the IEEE Industrial Electronics Society, Beijing, 2017.

SCHWAB, K. **The fourth industrial revolution**. World Economic Forum. World Economic Forum, 2016. Disponível em: <https://www.weforum.org/about/the-fourth-industrial-revolution-by-klaus-schwab>. Acesso em dezembro de 2018.

SNIDERMAN, B; MAHTO, M; COT TELER, M.J. **Industry 4.0 and Manufacturing Ecosystems – Exploring the World of Connected Enterprises**. Deloitte University Press, 2016. Disponível em https://www2.deloitte.com/content/dam/insights/us/articles/manufacturing-ecosystems-exploringworldconnectedenterprises/DUP_2898_Industry4.0ManufacturingEcosystems.pdf. Acesso em: outubro, 2017.

STEVAN, S.L.; LEME, M.O.; SANTOS, M.M.D. **Industria 4.0 - Fundamentos Perspectivas e aplicações**- 2018, Érica, 2018.

STOPPER, M; KATALINIC, B. **Service-oriented Architecture Design Aspects of OPC UA for Industrial Applications**. In Proceedings of the International Multiconference of Engineers and Computer Scientists (IMECS), Hong Kong, China, 2009.

SON, M.; YI, M.J. **A Study on OPC Specifications: Perspective and Challenges**. In International Forum on Strategic Technology. 2010 IEEE International Conference on, IEEE, 2010.p.193-197.

TAN, L.; WANG, N. **Future Internet: The Internet of Things**. In International Conference on Advanced Computer Theory and Engineering (ICACTE), 2010 IEEE International Conference on. IEEE, 2010.p.376-380.

TELLO, D. **Industry 4.0: Application of advanced services in logistics.** Universidade Politénica de Catalunya Barcelonatech, 2018.

TEIXEIRA, P. **Hands – Ebook: On Node.js.** LeanPub, 2012.

UNIPI, **Unipi technical documentation – REV 1.1**, 2019 – Disponível em: https://kb.unipi.technology/_media/files:products:unipi1:unipi_1_technical_documentation-en.pdf. Acesso em dezembro de 2019.

XU, L.D.; He, W.; LI, S. **Internet of Things in Industries: A Survey.** Industrial Informatics, IEEE Transactions on, v.10, n. 4, p. 2233-2243, 2014.

WANG, Y.; TOWARA, T.; ANDERL, R. **Topological Approach for Mapping Technologies in Reference Architectural Model Industrie 4.0 (RAMI 4.0).** Proceedings of the World Congress on Engineering and Computer Science 2017 Vol II WCECS 2017, October 25-27, 2017, San Francisco, USA.

WANT, R. **"An introduction to RFID technology,"** in Pervasive Computing, IEEE, vol.5, no.1, pp.25-33, Jan.-March, 2016.

WITKOWSKI, K. **Internet of Things, Big Data, Industry 4.0 – Inovative Solutions in Logistics and Supply Chains Managements.** 7th International Conference on Engineering, Project, and Production Management - Procedia Engineering – 2017.

WOLLSCHLAEGER, M.; SAUTER, T.; JASPERNEITE, J. **The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0,** IEEE Industrial Electronics Magazine, vol.11, pp.17-27, 2017.

YE, X.; HONG, H, S. **Toward Industry 4.0 Components: Insights into and Implementation of Asset Administration Shells,** In IEEE Industrial Electronics Magazine, vol. 13, no 1, pp.13-25, Março, 2019.

ZAHEER, M.A. **RAMI 4.0 (Part 3): Smart Electronic Industry 4.0 Hierarchy Level.** DZone/IoT Zone, 2017. Disponível em: <https://dzone.com/articles/rami-40-part-3-smart-electronic-industry-40-hierar>. Acesso em: dezembro de 2017.

ZEZULKA, F. MARCON, P.; VESELY, I.; SAJDL, O. **An Introduction in the phenomenon.** IFAC-PapersOnLine, 49(25), 2016, pp. 8-12.

ZEZULKA, F. MARCON, P.; BRADAC, Z.; ARM, J.; BENESL, T.; VESELY, I. **Communication Systems for Industry 4.0 and the IIoT.** 15th IFAC Conference on Programmable Devices and Embedded Systems, Ostrava, 2018.