

RODRIGO PITA ROLLE

**Arquitetura para Construção de Gêmeos Digitais  
com foco na Indústria 4.0**

**Bauru  
2019**

RODRIGO PITA ROLLE

Arquitetura para Construção de Gêmeos Digitais com  
foco na Indústria 4.0

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Estadual Paulista “Júlio de Mesquita Filho” para obtenção do título de Mestre em Engenharia Elétrica.

**Área de Concentração:** Automação

**Linha de Pesquisa:** Mecatrônica

**Orientador:** Prof. Dr. Eduardo Paciência Godoy

Bauru  
2019

Rolle, Rodrigo Pita.

Arquitetura para criação de gêmeos digitais com  
foco na Indústria 4.0 / Rodrigo Pita Rolle, 2019  
95 f. : il.

Orientador: Eduardo Paciência Godoy

Dissertação (Mestrado) -Universidade Estadual  
Paulista. Faculdade de Engenharia, Bauru, 2019

1. Virtualização. 2. Indústria 4.0. 3. Realidade  
Virtual. 4. Gêmeos Digitais. I. Universidade Estadual  
Paulista. Faculdade de Engenharia. II. Título.

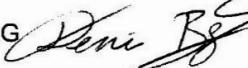
**ATA DA DEFESA PÚBLICA DA DISSERTAÇÃO DE MESTRADO DE RODRIGO PITA ROLLE,  
DISCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA, DA  
FACULDADE DE ENGENHARIA - CÂMPUS DE BAURU.**

Aos 16 dias do mês de agosto do ano de 2019, às 13:30 horas, no(a) ICTS - UNESP - Campus Sorocaba, 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 / Sorocaba, Prof. Dr. DENIS BORG do(a) Departamento de Engenharia Mecatrônica / Faculdade de Engenharia de Sorocaba (FACENS), Profª. Drª. MARILZA ANTUNES DE LEMOS do(a) Departamento de Engenharia de Controle e Automação / Instituto de Ciência e Tecnologia / UNESP / Sorocaba, sob a presidência do primeiro, a fim de proceder a arguição pública da DISSERTAÇÃO DE MESTRADO de RODRIGO PITA ROLLE, intitulada **ARQUITETURA PARA CONSTRUÇÃO DE GÊMEOS DIGITAIS COM FOCO NA INDÚSTRIA 4.0**. 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. DENIS BORG



Profª. Drª. MARILZA ANTUNES DE LEMOS



# Agradecimentos

A Deus, por conceder a vida, a saúde e a capacitação para realizar este trabalho.

A meus pais e familiares pelo suporte, incentivo e compreensão nos momentos de ausência.

Ao Professor Eduardo Paciência Godoy pela confiança e apoio em todas as etapas do desenvolvimento deste trabalho.

Ao Professor Fabrício Junqueira pela colaboração com no desenvolvimento do modelo 3D animado da estação de trabalho *Sorting*.

Aos Professores Denis Borg e Marilza Antunes de Lemos pelas valiosas contribuições para a realização deste trabalho.

Aos colegas Vinícius Martucci, Pablo Melo e demais colegas pela parceria e troca de experiências ao longo desta pesquisa.

Ao corpo técnico da UNESP e a todos que contribuíram direta ou indiretamente para o desenvolvimento deste trabalho.

*“Experiência, tentativa e erro, constante reflexão  
e revisão do itinerário - tais são os únicos meios pelos quais  
um homem pode, com a graça de Deus, adquirir conhecimento.”*  
*(Olavo de Carvalho)*

# Resumo

A evolução da computação e das ferramentas de simulação possibilita o uso de virtualização no ambiente industrial, de forma que os modelos em *software* transcendam a mera função de etapa de projeto, tornando-se provedores de informações relevantes para a avaliação do desempenho dos processos industriais. Os Gêmeos Digitais são uma abordagem para integração de sistemas físicos e virtuais, cujo principal objetivo é elevar o desempenho do sistema real através da informação gerada no ambiente virtual, que replica o funcionamento das partes físicas. Este trabalho consiste no desenvolvimento de uma arquitetura para construção de gêmeos digitais aderente à Indústria 4.0, considerando alguns dos principais requisitos do ambiente industrial tais como ganho de desempenho, aumento da flexibilidade dos processos, redução de tempos de parada e reprogramação, entre outros. A arquitetura desenvolvida para a implementação do gêmeo digital é composta por cinco módulos integrados através de redes Ethernet/IP utilizando os protocolos TCP e UDP. O primeiro é o *OpenPLC*, elemento central de controle de processos em conformidade com a norma IEC 61131-3, que é um *software* que incorpora as funcionalidades de um Controlador Lógico Programável (CLP), capaz de se comunicar com *hardware* de E/S remoto e com aplicativos de virtualização. O segundo é o processo de automação real, no qual é incorporado *hardware* apropriado para controle via *OpenPLC*. O terceiro é o modelo matemático do processo de automação em ambiente virtual, implementado neste trabalho através da ferramenta *State Transition Table* do aplicativo Matlab. Este modelo representa o comportamento lógico do processo de automação. O quarto é o módulo de visualização, composto por um modelo 3D animado que reproduz os estados do processo de automação em tempo de execução. O quinto é o módulo de análise de dados do processo de automação, que recebe dados dos ambientes físico e virtual para realizar operações de comparação, avaliação operacional e detecção de falhas no processo. A validação da arquitetura desenvolvida foi realizada com a implementação e testes de operação dos gêmeos digitais de dois processos de automação (estações de teste e separação de peças). Os ensaios de validação permitiram a verificação do funcionamento correto da arquitetura, bem como as análises de desempenho de comunicação realizadas mostraram que a temporização dos eventos de comunicação entre os módulos é adequada para aplicações de sistemas discretos.

**Palavras-chave:** Virtualização, Indústria 4.0, OpenPLC, Gêmeos Digitais.

# Abstract

The evolution of computing science and simulation tools enables the usage of virtualization in the industrial environment, so that software models transcend the mere function of project step and become relevant information providers for the evaluation of field equipment. Digital Twins are an approach for intercommunicating physical and virtual systems, whose main aim is to improve performance of the real system by using information generated on the virtual system that replicates the work of physical parts. This work presents the development of an architecture for building digital twins in the context of the Industry 4.0, considering some of the main requisites of the industrial environment, such as performance gains, increase of process flexibility, decrease of setup time, among others. The developed architecture for implementing the digital twin is composed by five modules integrated through Ethernet/IP networks utilizing TCP and UDP protocols. The first one is *OpenPLC*, central process control element in conformity with IEC 61131-3 standard, central element of process control that incorporates the functionalities of a Programmable Logic Controller (PLC), that is a software capable of communicating with I/O hardware and virtualization applications. The second one is the real automation process, in which it is included appropriate hardware for enabling control via *OpenPLC*. The third one is the mathematical model of the automation process in virtual environment, that in this work is implemented with the State Transition Table tool on Matlab software. This model represents the logical behavior of the automation system and is fed by PLC data in execution time. The fourth one is the visualization module, composed by an animated 3D model that reproduces the process' states in execution time. The fifth is the data analysis module, that receives data from both physical and virtual environments to perform operations such as comparison, operational evaluation and failure detection in the process. The validation of the developed architecture was performed by its implementation and operational tests involving the digital twins of two automation processes (stations for testing and sorting of plastic pieces). The validation tests made it possible to verify correct execution of the architecture, as well as communication performance analysis demonstrated that timing on communication events between modules is appropriate for discrete systems.

**Keywords:** Virtualization, Industry 4.0, OpenPLC, Digital Twins.

# Lista de ilustrações

Figura 1 – Representação esquemática das intersecções entre IoT, IIoT, CPS e Indústria 4.0. . . . .	14
Figura 2 – Esquemas de comissionamento de sistemas. . . . .	17
Figura 3 – Fluxo de desenvolvimento de um sistema de comissionamento virtual. . . . .	18
Figura 4 – Implementação de um gêmeo digital. . . . .	19
Figura 5 – Proposta de arquitetura de gêmeo digital de chão-de-fábrica. . . . .	20
Figura 6 – Proposta de arquitetura conceitual de implementação de gêmeos digitais para pequenas e médias empresas. . . . .	21
Figura 7 – Exemplo de sobreposição de elementos virtuais sobre o ambiente real. . . . .	23
Figura 8 – Comparativo entre os conceitos de RV e RA. . . . .	23
Figura 9 – Arquitetura desenvolvida para a implementação de Gêmeos Digitais. . . . .	25
Figura 10 – Esquemático da comunicação implementada pela interface de comunicação UDP <i>SimLink</i> . . . . .	27
Figura 11 – Esquemático de uso das entradas e saídas do SoC ESP8266 para o <i>OpenPLC</i> . . . . .	28
Figura 12 – Sistema Modular de Automação Festo MPS 200. . . . .	28
Figura 13 – Esquemático da modelagem de um processo físico. . . . .	30
Figura 14 – Representação de processos industriais no aplicativo <i>Factory I/O</i> . . . . .	33
Figura 15 – Diagrama <i>Ladder</i> utilizado para refletir um sinal de entrada real para o ambiente virtual. . . . .	36
Figura 16 – Estação <i>Testing</i> e peças que serão utilizadas no processo. . . . .	36
Figura 17 – Módulo de seleção de peças por característica ( <i>Sorting</i> ). . . . .	37
Figura 18 – Placa de acondicionamento de sinais para uso do SoC ESP8266 como interface de E/S. . . . .	38
Figura 19 – Diagrama <i>Ladder</i> desenvolvido para o controle da estação <i>Testing</i> . . . . .	39
Figura 20 – Diagrama <i>Ladder</i> desenvolvido para o controle da estação <i>Sorting</i> . . . . .	40
Figura 21 – Medição de intervalo de tempo entre dois eventos no aplicativo <i>LabVIEW</i> . . . . .	41
Figura 22 – Medição de um KPI no módulo de análise de dados. . . . .	42
Figura 23 – Inicialização de uma TET. . . . .	44
Figura 24 – TET elaborada para o processo-exemplo. . . . .	46
Figura 25 – Definição das variáveis da TET. . . . .	46
Figura 26 – Simulação completa do modelo do processo-exemplo em TET. . . . .	47
Figura 27 – Diagrama esquemático da sequência de estados utilizada para modelagem da estação <i>Testing</i> . . . . .	48
Figura 28 – Diagrama esquemático da sequência de estados utilizada para modelagem da estação <i>Sorting</i> . . . . .	49
Figura 29 – TET em operação com dados recebidos via UDP. . . . .	50

Figura 30 – Aplicativo desenvolvido para a visualização 3D da estação Festo MPS <i>Sorting</i>	51
Figura 31 – Aplicação cliente UDP.	51
Figura 32 – Aplicação cliente UDP para múltiplas portas.	52
Figura 33 – Aplicação cliente UDP final.	52
Figura 34 – Biblioteca de componentes de processos industriais do aplicativo <i>Factory I/O</i> .	53
Figura 35 – Configuração do módulo escravo Modbus do aplicativo <i>Factory I/O</i> .	53
Figura 36 – Representação da estação de trabalho <i>Testing</i> aplicativo <i>Factory I/O</i> .	55
Figura 37 – Representação da estação de trabalho <i>Testing</i> aplicativo <i>Factory I/O</i> .	56
Figura 38 – Parâmetros de ajuste dos elementos emissores no <i>Factory I/O</i> .	57
Figura 39 – Interface de comunicação UDP em execução.	59
Figura 40 – Esquema montado para a realização dos experimentos com a arquitetura.	61
Figura 41 – Ciclo de comunicação entre o <i>OpenPLC</i> e o escravo ESP8266.	63
Figura 42 – Tempo gasto para a comunicação e cálculo do ciclo de <i>polling</i> para o escravo ESP8266.	64
Figura 43 – Tempo real de <i>polling</i> do escravo ESP8266 na presença de um outro escravo declarado mas desligado.	64
Figura 44 – Medições para o primeiro ESP8266 no ensaio com dois escravos em operação.	65
Figura 45 – Medições de tempo para o segundo escravo ESP8266 no ensaio com dois escravos em operação simultânea.	65
Figura 46 – Ciclo de comunicação entre o <i>OpenPLC</i> e o escravo <i>Factory I/O</i> .	66
Figura 47 – Medições de tempo de comunicação e <i>polling</i> para o aplicativo <i>Factory I/O</i> .	66
Figura 48 – Medições de <i>polling</i> dos aplicativos de comunicação UDP <i>Simulink</i> e <i>SimLink</i> .	67
Figura 49 – Ciclo de comunicação entre o aplicativo <i>LabVIEW</i> e o banco de dados de E/S do <i>OpenPLC</i> .	68
Figura 50 – Medições de tempo de resposta do <i>OpenPLC</i> às requisições Modbus realizadas pelo aplicativo <i>LabVIEW</i> .	69
Figura 51 – Análise de desempenho da comunicação dos módulos no teste completo.	72
Figura 52 – Comparativo entre os tempos de acionamento dos atuadores reais e do modelo virtual.	73
Figura 53 – Dispersão dos tempos de subida ao longo de 150 ensaios na estação <i>Testing</i> .	74
Figura 54 – Dispersão dos tempos de descida ao longo de 150 ensaios na estação <i>Testing</i> .	74
Figura 55 – Dispersão dos tempos de processo de peças pretas ao longo de 150 ensaios na estação <i>Sorting</i> .	75
Figura 56 – Dispersão dos tempos de processo de peças pretas ao longo de 150 ensaios na estação <i>Sorting</i> .	76
Figura 57 – Tela de exibição de avisos e erros.	78
Figura 58 – Resultados do teste com injeção de falhas.	79
Figura 59 – Diagrama de Estados e Transições desenvolvido para a modelagem dos estados da estação de trabalho <i>Testing</i> .	90

Figura 60 – Diagrama de Estados e Transições desenvolvido para a modelagem dos estados da estação de trabalho <i>Sorting</i> . . . . .	91
Figura 61 – Esquemático completo de simulação implementado no software <i>Simulink</i> . . .	92

# Listas de tabelas

Tabela 1 – Família de <i>toolkits</i> e ferramentas do aplicativo <i>Simulink</i> . . . . .	31
Tabela 2 – Dispositivos escravos declarados no <i>OpenPLC</i> . . . . .	35
Tabela 3 – Correlação entre os tipos de peças e as possíveis leituras dos sensores da estação <i>Sorting</i> . . . . .	37
Tabela 4 – Indicadores-chave escolhidos para a avaliação das estações de trabalho <i>Testing</i> e <i>Sorting</i> . . . . .	42
Tabela 5 – Correlação entre as cores das peças reais e as peças disponíveis no aplicativo <i>Factory I/O</i> . . . . .	55
Tabela 6 – Lista dos parâmetros de temporização dos diversos módulos da arquitetura.	70
Tabela 7 – KPIs da planta lidos através do módulo de análise de dados do processo.	73
Tabela 8 – Definições das métricas de geração de avisos e erros implementada no aplicativo de análise de dados do processo produtivo. . . . .	77
Tabela 9 – Resultados do teste com injeção de falhas. . . . .	78
Tabela 10 – Lista de Entradas e Saídas do sistema. . . . .	88

# Lista de abreviaturas e siglas

CAD	<i>Computer-Aided Design</i>
CLP	Controlador Lógico Programável
CPS	<i>Cyber-Physical Systems</i>
DP	Desvio padrão da amostra
IIoT	<i>Industrial Internet of Things</i>
IoT	<i>Internet of Things</i>
KPI	<i>Key Process Indicator</i>
M	Média de tempos medidos
OLE	<i>Object Linking and Embedding</i>
OPC	<i>OLE for Process Control</i>
OPC-UA	<i>OPC - Unified Architecture</i>
PHM	<i>Prognostics and Health Management</i>
PLM	<i>Product Lifecycle Management</i>
RA	Realidade Aumentada
RFID	<i>Radio-frequency identification</i>
RV	Realidade Virtual
SoC	<i>System on Chip</i>
TET	Tabela de Estados-Trasições
TI	Tecnologia da Informação
TO	Tecnologia Operacional
To	Tempo de Operação
UDP	<i>User Datagram Protocol</i>
VI	<i>Virtual Instrument</i>

# Sumário

<b>1</b>	<b>INTRODUÇÃO . . . . .</b>	<b>10</b>
<b>1.1</b>	<b>JUSTIFICATIVA . . . . .</b>	<b>10</b>
<b>1.2</b>	<b>OBJETIVO . . . . .</b>	<b>11</b>
<b>1.3</b>	<b>ESTRUTURA E CONTEÚDO . . . . .</b>	<b>12</b>
<b>2</b>	<b>DIGITALIZAÇÃO INDUSTRIAL . . . . .</b>	<b>13</b>
<b>2.1</b>	<b>INTERNET DAS COISAS E INDÚSTRIA 4.0 . . . . .</b>	<b>13</b>
<b>2.2</b>	<b>SIMULAÇÃO DE SISTEMAS INDUSTRIAIS . . . . .</b>	<b>15</b>
2.2.1	COMISSIONAMENTO VIRTUAL . . . . .	16
2.2.2	GÊMEOS DIGITAIS . . . . .	18
2.2.3	REALIDADE VIRTUAL E REALIDADE AUMENTADA . . . . .	21
<b>3</b>	<b>DESCRIÇÃO DA ARQUITETURA . . . . .</b>	<b>25</b>
<b>3.1</b>	<b>CONTROLE DO PROCESSO - <i>OpenPLC</i> . . . . .</b>	<b>25</b>
3.1.1	INTERFACE DE COMUNICAÇÃO UDP <i>SimLink</i> . . . . .	26
3.1.2	DISPOSITIVO ESP8266 COMO INTERFACE DE E/S . . . . .	27
<b>3.2</b>	<b>PROCESSO DE MANUFATURA . . . . .</b>	<b>28</b>
<b>3.3</b>	<b>ANÁLISE DE DADOS DO PROCESSO . . . . .</b>	<b>29</b>
<b>3.4</b>	<b>MODELO DO PROCESSO . . . . .</b>	<b>29</b>
<b>3.5</b>	<b>VISUALIZAÇÃO 3D . . . . .</b>	<b>31</b>
3.5.1	<i>VISUAL STUDIO</i> . . . . .	32
3.5.2	<i>FACTORY I/O</i> . . . . .	32
<b>4</b>	<b>DESENVOLVIMENTO . . . . .</b>	<b>34</b>
<b>4.1</b>	<b>CONTROLE DO PROCESSO - <i>OpenPLC</i> . . . . .</b>	<b>34</b>
<b>4.2</b>	<b>PROCESSO DE MANUFATURA . . . . .</b>	<b>36</b>
<b>4.3</b>	<b>ANÁLISE DE DADOS DO PROCESSO . . . . .</b>	<b>41</b>
<b>4.4</b>	<b>MODELO DO PROCESSO . . . . .</b>	<b>43</b>
4.4.1	CRIAÇÃO DE TETs . . . . .	44
4.4.2	CRIAÇÃO DAS TETs DAS ESTAÇÕES <i>TESTING</i> E <i>SORTING</i> . . . . .	47
<b>4.5</b>	<b>VISUALIZAÇÃO 3D . . . . .</b>	<b>50</b>
4.5.1	<i>VISUAL STUDIO</i> . . . . .	50
4.5.1.1	EDIÇÃO DA CLASSE “ClienteTCP” . . . . .	51
4.5.2	<i>FACTORY I/O</i> . . . . .	52
<b>4.6</b>	<b>IMPLEMENTAÇÃO DA COMUNICAÇÃO VIA PROTOCOLO UDP</b>	<b>57</b>

<b>5</b>	<b>RESULTADOS E DISCUSSÃO . . . . .</b>	<b>61</b>
<b>5.1</b>	<b>ANÁLISE DE DESEMPENHO DO GÊMEO DIGITAL . . . . .</b>	<b>62</b>
5.1.1	DISPOSITIVOS ESP8266 . . . . .	62
5.1.2	<i>FACTORY I/O</i> . . . . .	65
5.1.3	COMUNICAÇÃO UDP . . . . .	67
5.1.4	<i>LABVIEW</i> . . . . .	68
5.1.5	TEMPORIZAÇÃO DO SISTEMA COMPLETO . . . . .	69
<b>5.2</b>	<b>ANÁLISE DE DADOS DO PROCESSO . . . . .</b>	<b>72</b>
5.2.1	LEVANTAMENTOS ESTATÍSTICOS . . . . .	73
5.2.2	CRIAÇÃO DE ALARMES . . . . .	76
5.2.3	TESTE COM INJEÇÃO DE FALHAS . . . . .	78
<b>6</b>	<b>CONCLUSÕES . . . . .</b>	<b>81</b>
<b>6.1</b>	<b>TRABALHOS FUTUROS . . . . .</b>	<b>82</b>
<b>7</b>	<b>REFERÊNCIAS BIBLIOGRÁFICAS . . . . .</b>	<b>83</b>
<b>8</b>	<b>APÊNDICES . . . . .</b>	<b>88</b>
	<b>APÊNDICE A – LISTA DE ENTRADAS E SAÍDAS DO PROCESSO INDUSTRIAL . . . . .</b>	<b>88</b>
	<b>APÊNDICE B – TABELAS DE ESTADOS E TRANSIÇÕES . . . . .</b>	<b>90</b>
	<b>APÊNDICE C – CÓDIGO EM C# IMPLEMENTADO NO VISUAL STUDIO . . . . .</b>	<b>93</b>

# 1 INTRODUÇÃO

## 1.1 JUSTIFICATIVA

O atual cenário tecnológico é um grande desafio para as indústrias, uma vez que a acirrada competição aumenta as demandas por processos eficientes, ágeis, confiáveis e com o menor custo possível (HARRISON, 2011), de forma que se obtenha maior produtividade com o maquinário disponível, buscando minimizar os tempos de *setup* e de parada, bem como as perdas por processamento incorreto e falhas de equipamentos.

Um importante foco de pesquisa nesta área é o monitoramento e otimização de processos, visando não apenas a minimização de prejuízos materiais, mas também a integridade física e a segurança de operadores e equipamentos de campo. Os conceitos e recursos trazidos pela chamada “Quarta Revolução Industrial” (ou Indústria 4.0) abriram novas possibilidades para o estudo de técnicas de diagnóstico de falhas em automação industrial e outras aplicações. Esta revolução industrial é marcada pelo advento da chamada Internet das Coisas (IoT - acrônimo inglês para “*Internet of Things*”) (TRAPPEY et al., 2017), que tem por característica a conexão de objetos à Internet, de forma que informações e ações sejam realizadas e transmitidas sem a ação do homem (AYDOS et al., 2016).

Um campo de aplicação impulsionado pelo contexto da IoT foi a integração de sistemas físicos a contrapartes simuladas ou modeladas. O uso de simulação no desenvolvimento de sistemas mecatrônicos tem crescido conforme o desenvolvimento das tecnologias de informática e aplicativos para tal. Nota-se também que os modelos computacionais têm tido crescente relevância e diferentes escopos de aplicação ao longo do tempo.

A evolução das ferramentas computacionais permitiu o uso de novas técnicas tais como o comissionamento virtual, no qual se permite a integração de modelos de simulação a entidades do mundo real, de forma a analisar o comportamento do sistema antes de ser implementado na prática (MAKRIS; MICHALOS; CHRYSSOLOURIS, 2012). A ideia básica do comissionamento virtual é conectar um modelo digital da planta com um controlador real, de forma que se possa avaliar a corretude do controle implementado antes de executá-lo na planta real (LIU; SUCHOLD; DIEDRICH, 2012).

Ainda no contexto da IoT, nota-se a crescente integração entre os sistemas de campo e a nuvem, através de alternativas como os Sistemas Cyber-Físicos (CPS - acrônimo inglês para *Cyber Physical Systems*). Este tipo de sistema é caracterizado pela inserção de elementos de computação no ambiente produtivo. Estes elementos se comunicam com sensores e atuadores, produzindo alterações no ambiente físico nos quais operam. CPSs em geral utilizam sensores para conectar toda a inteligência distribuída no sistema de forma

a obter maior conhecimento do ambiente e proporcionar ações mais precisas, agregando flexibilidade e eficiência aos sistemas (ZANNI, 2015). O desenvolvimento de equipamentos de campo inteligentes tem permitido o crescente uso de sensores e máquinas integradas em rede, que por sua vez apresentam um grande potencial de integração para alcançar resiliência, auto-adaptatividade e inteligência (LEE; BAGHERI; KAO, 2015).

Posteriormente, já na era da Indústria 4.0, desenvolveu-se a ideia de “gêmeo digital” (*Digital Twin*), cujo objetivo é não apenas simular em ambiente virtual componentes e sistemas reais, mas integrar estes ambientes através de protocolos de comunicação, de forma que haja convergência e cooperação entre eles (GRIEVES, 2014). Tao et al. (2018) discorre sobre a aplicação do conceito de gêmeos digitais nas três fases do ciclo de vida de produtos: *design*, manufatura e serviço. A área de PLM (acrônimo inglês para *Product Lifecycle Management*) é uma área de grande potencial para o uso de gêmeos digitais, atraindo a atenção do segmento industrial e também dos pesquisadores.

A pesquisa na área de gêmeos digitais é bastante recente, sendo assim muitas barreiras técnicas ainda precisam ser superadas. O trabalho de Tao e Zhang (2017) apresenta uma arquitetura conceitual de gêmeo digital de chão-de-fábrica. De acordo com os autores, é preciso superar dificuldades quanto à conectividade dos ambientes real e virtual em tempo real, o tratamento de inconsistências entre os modelos virtuais e as entidades reais e o tratamento do crescente volume de dados gerados. Uhlemann et al. (2017) propõem uma arquitetura conceitual de gêmeos digitais para pequenas e médias empresas, destacando como desafios a falta de padronização na aquisição de dados, a dificuldade de sincronização entre os ambientes real e virtual em aplicações de tempo real e a falta de sistemas centralizados de informação. Zheng, Yang e Cheng (2018) apresentam um *framework* de implementação de gêmeos digitais, no qual é implementado o gêmeo digital de uma estação de soldagem, abordando os desafios de coleta de dados, modelagem virtual e processamento de dados em tempo real.

Este trabalho tem a intenção de gerar como contribuição uma arquitetura para construção de gêmeos digitais, implementando técnicas aderentes à Indústria 4.0 para solucionar os desafios de comunicação em tempo de execução, criação de modelos virtuais, integração dos ambientes real e virtual e análise aprofundada do processo. Será priorizado o uso de ferramentas de código aberto e livre distribuição, uma vez que os trabalhos relacionados a esta área frequentemente se utilizam de soluções proprietárias.

## 1.2 OBJETIVO

O objetivo deste trabalho é desenvolver uma arquitetura para construção de gêmeos digitais que integre ferramentas de controle de processos industriais, aplicativos de simulação computacional e tratamento de dados utilizando protocolos de comunicação

aceitos no contexto da Indústria 4.0. O sistema proposto deve seguir um padrão modular, com escalabilidade e baixa interdependência entre os módulos.

### **1.3 ESTRUTURA E CONTEÚDO**

O presente trabalho está dividido em 6 capítulos considerando a presente introdução e desconsiderando referências bibliográficas e apêndices.

O capítulo 2 trata da revisão bibliográfica, onde é apresentado um panorama dos conceitos utilizados neste trabalho, bem como as aplicações apresentadas na literatura recente.

O capítulo 3 apresenta o material e os métodos utilizados na elaboração deste trabalho. São descritas as ferramentas de controle de processos, modelagem e simulação, visualização 3D e o sistema industrial utilizado nas aplicações de teste e validação.

No capítulo 4 é descrito o desenvolvimento do trabalho, sobretudo a arquitetura, a preparação e a implementação dos módulos.

O capítulo 5 trata dos resultados e discussões.

O capítulo 6 apresenta as conclusões do trabalho e propostas de trabalhos futuros.

## 2 DIGITALIZAÇÃO INDUSTRIAL

### 2.1 INTERNET DAS COISAS E INDÚSTRIA 4.0

O contexto tecnológico recente, fortemente caracterizado pelo uso da Internet como meio de transmissão e de dispositivos inteligentes (*smart devices*) ligados à rede, é usualmente denominado por Internet das Coisas (IoT - acrônimo inglês para *Internet of Things*). A IoT é uma onda tecnológica na qual diversos dispositivos (“coisas”) são conectados à Internet e trocam informações entre si, expandindo o paradigma de comunicação de “pessoas e pessoas” para “pessoas e coisas” e também “coisas e coisas” (OLIVEIRA, 2017).

A ideia de Internet das Coisas Industrial (IIoT – acrônimo inglês para *Industrial Internet of Things*) é tida como um desdobramento da IoT com aplicações voltadas para o ambiente industrial.

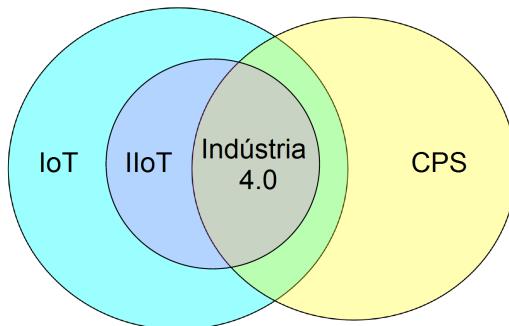
Em se tratando de um campo tecnológico bastante recente, alguns termos e conceitos são, por vezes, utilizados indistintamente, o que gera alguma confusão. A seguir apresenta-se uma breve distinção entre os principais conceitos que, embora alguns sejam bastante próximos, não podem ser substituídos indistintamente:

- IoT: o que normalmente se denomina como IoT (dispositivos inteligentes ligados à Internet) poderia ser melhor descrito como “IoT Comercial”, cujo foco está no consumidor final e aplicações do cotidiano do público alvo (PALATTELLA et al., 2016);
- IIoT: na IoT Industrial o foco está na integração entre Tecnologia Operacional (TO) e Tecnologia da Informação (TI) e como os dispositivos inteligentes e redes de sensores podem colaborar para a otimização da produção, seja do ponto de vista do aumento de flexibilidade da produção ou da análise intensiva de dados (*big data*) para otimização de processos fabris (SISINNI et al., 2018);
- Indústria 4.0: este conceito surgiu na Alemanha em 2011 (JAZDI, 2014) e ganhou visibilidade global, sendo atualmente utilizado para abordar o uso de tecnologias de Internet para elevar a eficiência de produção através de serviços inteligentes (*smart services*) em fábricas inteligentes (*smart factories*). Do ponto de vista de aplicação, é uma mistura da ideia de IoT com Sistemas Cyber-Físicos (CPS - acrônimo inglês para *Cyber-Physical Systems*) (SISINNI et al., 2018);
- CPS: sistema na qual há integração da computação com os processos físicos, ou seja, uma “intersecção” entre o físico e o virtual. De forma geral, são sistemas automatizados que realizam processos controlados por computadores.

zados que permitem conexão das operações da realidade física com infraestrutura de computação e comunicação (JAZDI, 2014). Para ser adequado à Indústria 4.0, um CPS deve abranger clientes, máquinas, produtos, estoques e prestadores de serviço, proporcionando interação e a execução autônoma de operações entre eles (PISCHING et al., 2017).

A Figura 1 apresenta as intersecções entre os conceitos de IoT, IIoT, Indústria 4.0 e CPS. Nota-se que a IIoT é mostrada como um subconjunto da IoT, enquanto a Indústria 4.0 é uma área de intersecção entre os conceitos de IIoT e CPS.

Figura 1 – Representação esquemática das intersecções entre IoT, IIoT, CPS e Indústria 4.0.



Fonte: Adaptado de Sisinni et al. (2018).

Alguns princípios fundamentais são recorrentes nas diversas definições de Indústria 4.0 introduzidas ao longo do tempo (BASSI, 2017). São eles:

- Uso extensivo da Internet: o uso de serviços de comunicação cabeada ou sem fio nos dispositivos inteligentes possibilita acesso direto às camadas superiores de processos e serviços. Isto eleva o valor agregado, promove suporte para o uso otimizado de recursos e o controle inteligente do processo de fabricação (JAZDI, 2014). Uma das principais vertentes é o uso de *big data* para a predição de falhas, de forma que se faça a manutenção antes que ocorra a quebra do equipamento. Isto reduz drasticamente o tempo de parada da planta e os prejuízos decorrentes da interrupção da linha (BASSI, 2017);
- Flexibilidade: a capacidade de operar linhas produtivas flexíveis, com capacidade de customização e minimização do tempo de reprogramação (*setup*), de forma a atender as necessidades individuais do consumidor sem custo adicional por parada e reprogramação da linha de produção (JAZDI, 2014). Algumas tecnologias habilitadoras típicas são a impressão 3D e a rastreabilidade e identificação do produto na linha de produção (BASSI, 2017);

- Comunicação, Virtualização e CPS: a criação de modelos unificados em contraposição aos diversos protocolos industriais atuais (AS-i, PROFIBUS, PROFINET, CAN etc.) que em geral são incompatíveis entre si ou de difícil interoperabilidade. A disponibilização de um *framework* comum de comunicação permite o uso de virtualização, conectando os sistemas físicos a contrapartes virtuais. Uma das propostas mais promissoras é o protocolo OPC-UA (acrônimo inglês para *OLE for Process Control - Unified Architecture*), que facilita a troca de informações ao longo de toda a infraestrutura de controle e gerenciamento (BASSI, 2017).

Os CPS são sistemas que proporcionam uma rede integrada de recursos físicos e processos que permitem a fusão dos mundos físico e virtual, onde sistemas embarcados integrados controlam processos físicos com operações de requisição e resposta, sendo que os processos físicos interferem nos processos virtuais e vice-versa. Os CPS são baseados na relação estreita entre sistemas embarcados, dispositivos de controle, computação, comunicação e redes (PESSOA et al., 2018).

Um CPS consiste de uma unidade de controle (normalmente um ou mais micro-controladores) que controla os sensores e atuadores que são necessários para interagir com o mundo real e processar os dados obtidos. Requer ainda uma interface que habilite a comunicação com outros sistemas embarcados ou a nuvem. Tal troca de dados é o elemento fundamental de um CPS, desde que os dados podem ser associados e avaliados remotamente em servidores centrais, por exemplo (JAZDI, 2014). No ambiente produtivo, os CPS são capazes de otimizar processos compartilhando informação em tempo real entre as máquinas, cadeia de suprimentos, sistemas de gestão de negócio e consumidores. Podem ainda implementar auto-monitoramento e controle dos processos de produção, adaptando a produção às preferências dos clientes (ZANNI, 2015).

## 2.2 SIMULAÇÃO DE SISTEMAS INDUSTRIAIS

A simulação é uma poderosa ferramenta para a análise de sistemas complexos, desempenhando um papel significante na avaliação do *design* e da *performance* operacional de sistemas de manufatura (NEGAHBAN; SMITH, 2014). O uso tradicional de simulação se limitava às etapas de planejamento e *design*, utilizando distribuições estatísticas para modelar o comportamento estocástico do ambiente. Entretanto, um passo natural é o uso destas ferramentas para controlar eventos num sistema real, dado que elas apresentam a capacidade de capturar complexas interações do sistema (SON et al., 2002).

No ambiente de manufatura atual, a análise de simulações é utilizada para diversos tipos de aplicações, tais como análise de *throughput* e validação de sistemas (VISWA-NATHAN et al., 2011). Negahban e Smith (2014) elencam onze áreas nas quais se emprega simulação no contexto da manufatura. Estas áreas vão desde o design de sistemas de

manufatura até a metamodelagem e métodos de otimização para sistemas, passando por controle em tempo real, análise de performance, previsão de operações de manutenção, entre outros.

É comum nomear-se uma simulação como sendo de “tempo real” (*real-time*) quando esta é iniciada por um sinal de entrada vindo da planta física em tempo real. Esta abordagem é útil no estudo de sistemas de produção em massa, na qual os resultados podem ser utilizados para prever o desempenho do sistema por um determinado período de tempo. Simulações assíncronas são úteis em estudos para redução de custos ou aumento de produtividade, porém não são aplicáveis para monitoramento do desempenho (SAEZ et al., 2018).

O controle em tempo real de sistemas de manufatura é um problema difícil por conta da complexidade e da natureza estocástica destes sistemas. Neste contexto, embora a simulação seja uma alternativa efetiva, é presente a dificuldade em se obter tempos de resposta adequados e também problemas quanto à agregação e à coleta de dados (NEGAHBAN; SMITH, 2014).

Neste contexto, a literatura apresenta algumas abordagens quanto ao uso de simulações no projeto, desenvolvimento e execução de sistemas de automação, que serão apresentadas nos tópicos seguintes. Cabe ressaltar que a digitalização ao longo da cadeia de valor tem se provado como um item essencial para o futuro da indústria, especialmente no que tange à flexibilização, modularidade e adaptabilidade em sistemas de fabricação (KOCH et al., 2014).

Embora tópicos relacionados à Indústria 4.0 estejam no foco da maior parte dos projetos empresariais, a falta de conhecimento específico tem inibido a implementação de tecnologias de digitalização (UHLEMANN et al., 2017). Com o rol de tecnologias inserido no contexto da Indústria 4.0, abre-se caminho para a implementação de monitoramento e sincronização em tempo real das atividades do mundo real com o espaço virtual, através de redes de comunicação (NEGRI; FUMAGALLI; MACCHI, 2017).

### 2.2.1 COMISSIONAMENTO VIRTUAL

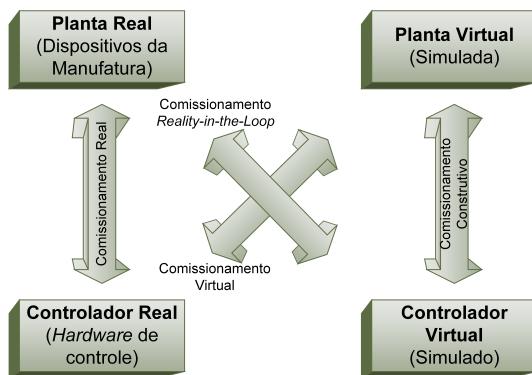
O comissionamento virtual é o conceito de utilizar uma réplica digital de um sistema mecânico para reduzir o tempo necessário na fase de comissionamento real (JOHANSSON, 2017). Comissionamento é definido como “todas as atividades com o objetivo de colocar em operação um sistema completamente montado e mecanicamente revisado” (REINHART; WÜNSCH, 2007). Em outras palavras, o comissionamento virtual tem por finalidade a validação de um modelo em software do sistema automatizado antes que o equipamento real seja colocado em operação. Com uso correto, estima-se que o comissionamento virtual pode reduzir o tempo de inicialização de sistemas em até 50%, além de reduzir o risco e o

custo nesta etapa. Ainda ajuda a fazer úteis os modelos de simulação em diversas etapas do projeto, bem como acelerar o desenvolvimento de produtos (JOHANSSON, 2017).

Existem quatro configurações possíveis (LEE; PARK, 2014) para o comissionamento de sistemas (Figura 2):

- Comissionamento real: ligação entre o sistema de controle real e a planta real;
- Comissionamento virtual (*Hardware-in-the-Loop*): ligação entre o sistema de controle real e uma planta simulada;
- Comissionamento *Reality-in-the-Loop*: ligação entre um controlador simulado e a planta real;
- Comissionamento construtivo (*Software-in-the-Loop*): ligação entre um controlador simulado e uma planta simulada.

Figura 2 – Esquemas de comissionamento de sistemas.

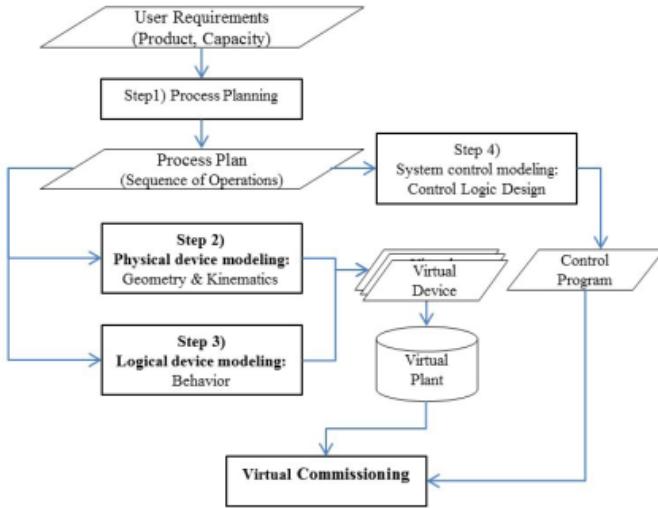


Fonte: Adaptado de Lee e Park (2014).

O comissionamento virtual permite a verificação completa de um sistema de manufatura realizando uma simulação envolvendo um processo (planta) virtual e um controlador real. Para tal, requer-se um modelo virtual completo da planta com descrição detalhada ao nível de sensores e atuadores (JOHANSSON, 2017).

Ko, Ahn e Park (2013) propõem um procedimento de design para o comissionamento virtual apresentado na Figura 3. Este é composto por quatro etapas principais: (1) Planejamento do processo; (2) Modelagem do equipamento físico (modelo CAD 3D com cinemática); (3) Modelagem lógica do equipamento (comportamento da planta - formalismos) e (4) Modelagem do sistema de controle. Diferentemente dos métodos tradicionais de projeto, este procedimento preconiza o desenvolvimento concorrente da parte mecânica e da parte elétrica de forma a reduzir o tempo necessário para a finalização do projeto.

Figura 3 – Fluxo de desenvolvimento de um sistema de comissionamento virtual.



Fonte: Ko, Ahn e Park (2013).

Diversos trabalhos acadêmicos têm explorado o campo do comissionamento virtual no ambiente industrial, dos quais são destacados alguns exemplos. Johansson (2017) desenvolve em sua tese um modelo virtual de uma célula de produção de engrenagens, de forma a imitar o funcionamento do sistema real em ambiente de *software*. A comunicação com o CLP é implementada através do protocolo de comunicação OPC. Makris, Michalos e Chryssolouris (2012) apresentam um estudo de caso numa indústria automotiva, na qual uma célula de soldagem de peças para o assoalho dos carros. Este estudo envolve a presença de dois robôs trabalhando de forma cooperativa, na qual a operação entre eles deve ser sincronizada. O modelo virtual da célula foi implementado no *software* Winmod. O trabalho desenvolvido por Cardoso, Rangel e Bastos (2013) realizou o comissionamento virtual de estações didáticas de manufatura utilizando comunicação OPC entre os CLPs e o aplicativo de simulação Arena.

## 2.2.2 GÊMEOS DIGITAIS

Os gêmeos digitais (tradução para a nomenclatura inglesa “*Digital Twins*”) são um conceito apresentado inicialmente por Grieves (2014), entretanto atualmente já existem diversas explicações e definições propostas para o termo (TAO et al., 2018). Uma definição geral mais utilizada até o momento diz que gêmeos digitais são uma simulação probabilística integrada de um produto complexo, utilizando os melhores modelos físicos disponíveis de forma a espelhar a vida do seu gêmeo correspondente (GLAESSGEN; STARGEL, 2012).

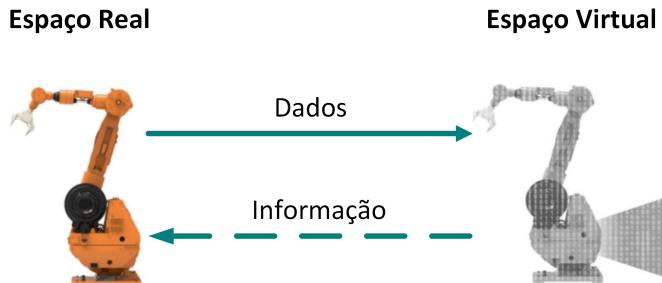
Discutindo a complexidade e o custo computacional das simulações, Grieves (2014) afirma que é possível criar versões leves do modelo virtual de forma que sejam selecionados os aspectos mais relevantes do sistema e desprezar detalhes desnecessários, o que reduz

o tamanho dos modelos e permite um processamento mais rápido. O objetivo desta simplificação é permitir que visualização e simulação de sistemas complexos e sistemas-de-sistemas em tempo real e com custo computacional aceitável, reduzindo também o tempo e o custo da comunicação da informação gerada.

Um gêmeo digital é constituído de três partes principais, sendo (1) o produto físico no espaço real, (2) o produto virtual no espaço virtual e (3) a conexão de dados e informações que conecta os espaços real e virtual. A parte virtual não apenas armazena histórico da parte física, mas também pode prover otimização e predição para ela, de forma que se busca sempre a convergência entre as partes (TAO; ZHANG, 2017).

A implementação de gêmeos digitais se dá como mostrado na Figura 4.

Figura 4 – Implementação de um gêmeo digital.



Fonte: Adaptado de Grieves (2014).

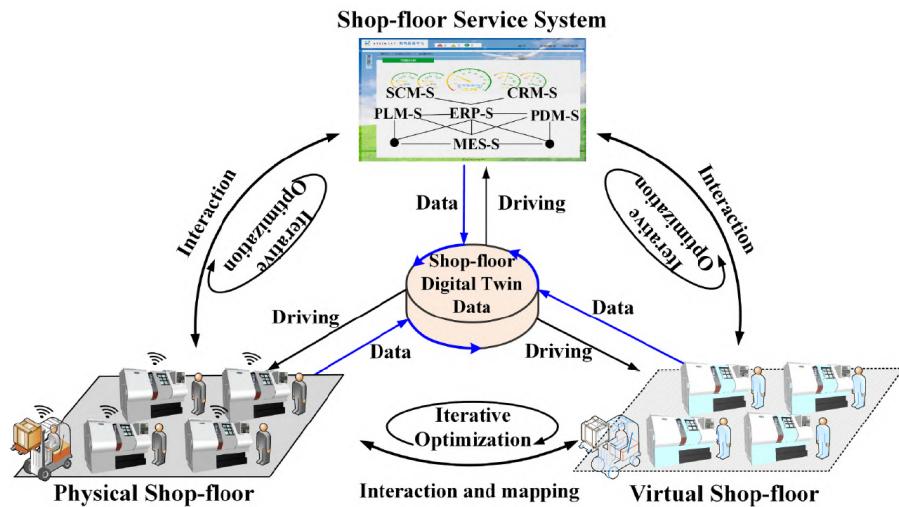
Os gêmeos digitais têm sido aplicados em diversos segmentos da prática industrial, desde o controle de chão-de-fábrica até em segmentos mais elaborados como a gestão do ciclo de vida (PLM - acrônimo inglês para *Product Lifecycle Management*) de produtos industrializados (TAO et al., 2018).

O trabalho desenvolvido por Vachálek et al. (2017) apresenta a implementação de gêmeos digitais utilizando as soluções da fabricante Siemens para controle (CLP) e implementação do espaço virtual (através do software “*Plant Simulation*”) sobre estações de fabricação modulares. A interface de comunicação foi baseada no protocolo OPC. Foi definida a sequência de processamento no espaço real, em seguida foi criado o ambiente virtual detalhando os processos físicos envolvidos no processo. Este modelo virtual permite que se alterem parâmetros de produção de modo a monitorar o comportamento do sistema sem que haja o risco de prejuízos com a implementação no sistema real. É possível também utilizar este gêmeo digital para otimizar a produção, redefinindo os parâmetros do sistema físico a partir dos dados coletados na contraparte virtual.

Tao e Zhang (2017) apresentam uma arquitetura conceitual de gêmeos digitais para o chão-de-fábrica que integra os espaços real e simulado, bem como plataformas de serviços agregados ao sistema, encapsulando funções de algoritmos, modelos e ferramentas na forma

de sub-serviços que são combinados e executados de forma a atender às demandas do sistema físico e do sistema virtual. É implementada a comunicação de forma que o espaço físico, o espaço virtual e o sistema de serviços interagem entre si de forma a otimizar os parâmetros de desempenho do sistema a partir de metodologias como PLM, MES e ERP. A Figura 5 ilustra o funcionamento do sistema proposto.

Figura 5 – Proposta de arquitetura de gêmeo digital de chão-de-fábrica.

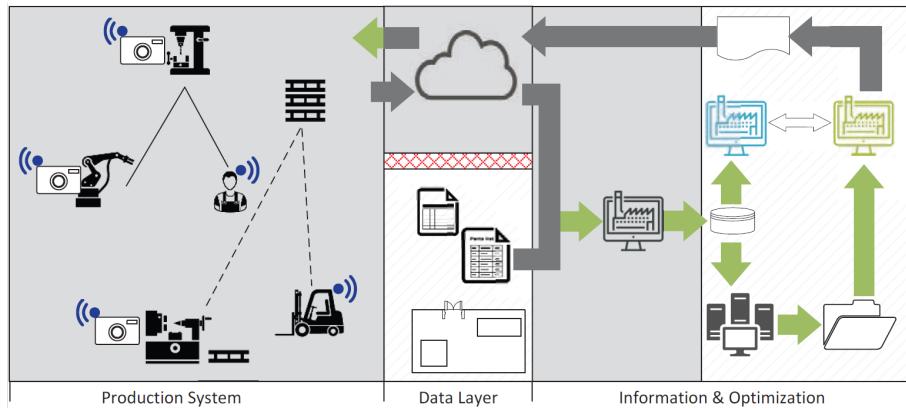


Fonte: Tao e Zhang (2017).

Os autores destacam que apenas quatro componentes da arquitetura foram discutidos, sendo que ainda é necessário implementar os demais. Os principais desafios envolvem a conexão e interação no sistema físico, o estabelecimento de comunicação de duas vias entre os espaços físico e virtual, o desenvolvimento de modelos de alta fidelidade para o espaço virtual entre outros.

Por sua vez, Uhlemann et al. (2017) apresentam uma arquitetura conceitual de construção de gêmeos digitais com foco em pequenas e médias empresas baseada em visão de máquinas e rastreamento baseado em sensores. Propondo um sistema físico descentralizado e autônomo onde cada equipamento se comunica com uma camada central de dados, propõe-se um tratamento de dados onde se distinguem 1) dados que não dependem de atualização em tempo real, tais como *layout* da planta, finalidade da máquina, lista de peças e 2) dados voláteis de tempo dependente, tais como fluxo de produção, rotas de transporte e outros. Todos estes dados são armazenados e utilizados para simulação e otimização do processo, gerando novos *sets* de parâmetros otimizados ao longo do tempo, que seriam implementados nos dispositivos de campo. O autor observa que soluções individuais para os desafios da arquitetura já são existentes, mas é preciso superar uma série de obstáculos para integrá-las. A Figura 6 ilustra a proposta.

Figura 6 – Proposta de arquitetura conceitual de implementação de gêmeos digitais para pequenas e médias empresas.



Fonte: Uhlemann et al. (2017).

O trabalho desenvolvido por Lee et al. (2013) discute implementações de gêmeos digitais na nuvem, implementando a metodologia PHM (acrônimo inglês para *Prognostics and Health Management*) para estudo do ciclo de vida de equipamentos e previsão de falhas, de forma que os dados obtidos através das redes e sensores inteligentes (*smart sensors/networks*) se transformem em informação sobretudo quanto aos aspectos de manutenção e produtividade. No modelo proposto pelo autor, os dados coletados por sensores inteligentes são enviados para uma plataforma de análise na nuvem, na qual se executam os serviços de análise, previsão e visualização dos sistemas produtivos.

Zheng, Yang e Cheng (2018) apresentam um *framework* de aplicação de gêmeos digitais acompanhado de um caso de estudo. Neste, o espaço físico é construído com base nos conceitos de IoT, incluindo tecnologias como RFID (acrônimo inglês para “*Radio-frequency identification*”), QR code e *smart sensors*. A comunicação entre os espaços físico e virtual é realizada por uma camada de processamento de informação, cujas funções são mapear, processar e armazenar os dados. O espaço virtual é implementado em duas partes, sendo uma plataforma de ambiente virtual e uma aplicação de PLM. A plataforma de ambiente virtual estabelece um modelo 3D animado do sistema real e interage com o subsistema de PLM. O *framework* é implementado numa estação de soldagem para estudo e validação. Como resultados, foi possível monitorar o sistema em tempo real, obtendo transparência nos estados dos equipamentos, comparação do processo real com os padrões gerados virtualmente e geração de alertas de anormalidades em tempo de execução.

### 2.2.3 REALIDADE VIRTUAL E REALIDADE AUMENTADA

A Realidade Virtual (RV) é uma tecnologia aplicada através de uma interface, a qual possibilita a interação entre os sistemas eletrônicos e humanos. O termo é comumente usado pela mídia popular para descrever mundos que só existem em computadores e nossas

mentes (JERALD, 2015).

Como discutido por Steuer (1992), o termo RV tradicionalmente se refere a uma configuração de hardware que consiste em itens como um visor estereoscópico, computadores, fones de ouvido, alto-falantes e dispositivos de entrada 3D. Mais recentemente, o termo tem sido amplamente usado para descrever qualquer programa que inclua um componente 3D, independentemente do hardware que eles utilizam. Uma definição básica de uma experiência de RV é a substituição de um ou mais sentidos físicos por sentidos virtuais. A RV é promovida através do uso de dispositivos de visualização virtual associados a computadores, tendo como objetivo principal criar um cenário no meio digital, de forma que este seja o mais parecido possível com a realidade. Efeitos visuais, sonoros e até táteis, possibilitam um “mergulho” em um ambiente virtual simulado (COBURN; FREEMAN; SALMON, 2017).

A RV e sua eficácia podem ser classificadas nas seguintes categorias:

- Imersão: o grau em que o sistema de RV pode invocar estímulos do usuário para a imersão do sistema. Como por exemplo, um sistema de rastreamento do movimento da cabeça exibindo imagens em uma tela de acordo com o movimento aumentaria o nível de imersão, fazendo com que o sistema seja mais eficaz. Da mesma forma, na tentativa de recriar o efeito da chuva caindo sobre o usuário, se sistemas hápticos fossem implantados para simular o sentido de chuva caindo, faria o sistema mais imersivo (TOSHNIWAL; DASTIDAR, 2014);
- Vivacidade: refere-se à riqueza de detalhes que o sistema fornece. Isso varia em uma grande escala de fatores como a reprodução de cores para a resolução na qual o sistema é projetado. Com os avanços em tecnologias de exibição, a vivacidade em sistemas de RV tem melhorado muito (TOSHNIWAL; DASTIDAR, 2014);
- Interatividade: um sistema de RV tenta recriar não só a aparência da realidade, mas também o nível de interatividade que pode incorporar o sistema. A resposta adequada às ações do usuário é tomada em consideração ao avaliar a eficácia de um sistema de RV (TOSHNIWAL; DASTIDAR, 2014).

Derivado da Realidade Virtual em si, a Realidade Aumentada refere-se à superposição de informações sobre realidade atual (TOSHNIWAL; DASTIDAR, 2014). A realidade aumentada (RA) é uma visão direta ou indireta de um ambiente físico real, cujos elementos são aumentados por informações sensoriais geradas por computador, como som, vídeo, gráficos ou dados de GPS (CHAVAN, 2014). A RA é a realidade na qual os objetos ou imagens virtuais são montados com os objetos do mundo real. É uma tecnologia que complementa os objetos virtuais gerados por computador com os objetos do mundo real, o

que faz parecer que eles coexistem no mesmo espaço que o mundo real (TIWARI et al., 2016).

Ao contrário da RV, que cria um ambiente totalmente artificial, a RA usa o ambiente existente e sobrepõe novas informações sobre ele (CHAVAN, 2014), não recriando cenas em geral, mas apenas fornecendo informações úteis sobre a realidade atual (TOSHNIWAL; DASTIDAR, 2014). A Figura 7 demonstra um exemplo de utilização da tecnologia de RA, onde imagens e interações virtuais são sobrepostas ao ambiente real ao redor do usuário.

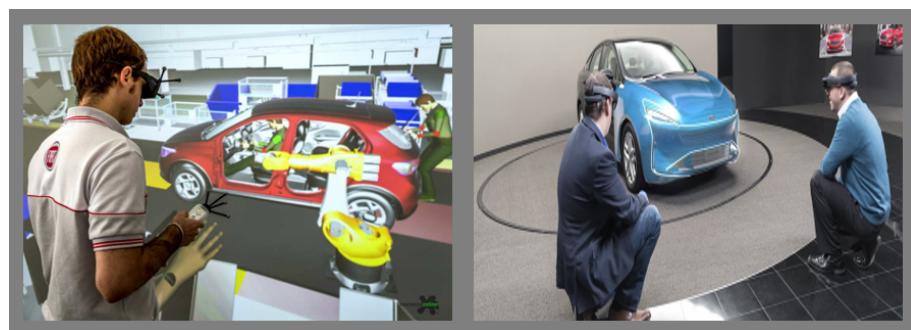
Figura 7 – Exemplo de sobreposição de elementos virtuais sobre o ambiente real.



Fonte: Chavan (2014).

Apesar da similaridade dessas tecnologias, que na maioria das vezes estão juntas em algum tipo de aplicação, existem diferenças nas suas características. Podemos dizer que RV insere o usuário em um ambiente virtual, enquanto a RA insere componentes virtuais em um ambiente real. A Figura 8 ilustra a diferença entre a RV e a RA, onde à esquerda é demonstrada uma aplicação em que o usuário interage com um ambiente totalmente virtual, e à direita é demonstrada uma aplicação que os usuários interagem com objetos virtuais inseridos em um ambiente real.

Figura 8 – Comparativo entre os conceitos de RV e RA.



Fonte: Ciriaco (2017).

Paelke (2014) desenvolve um sistema de RA de assistência à montagem de produtos, onde o sistema analisa os passos da montagem e fornece ao trabalhador instruções de como

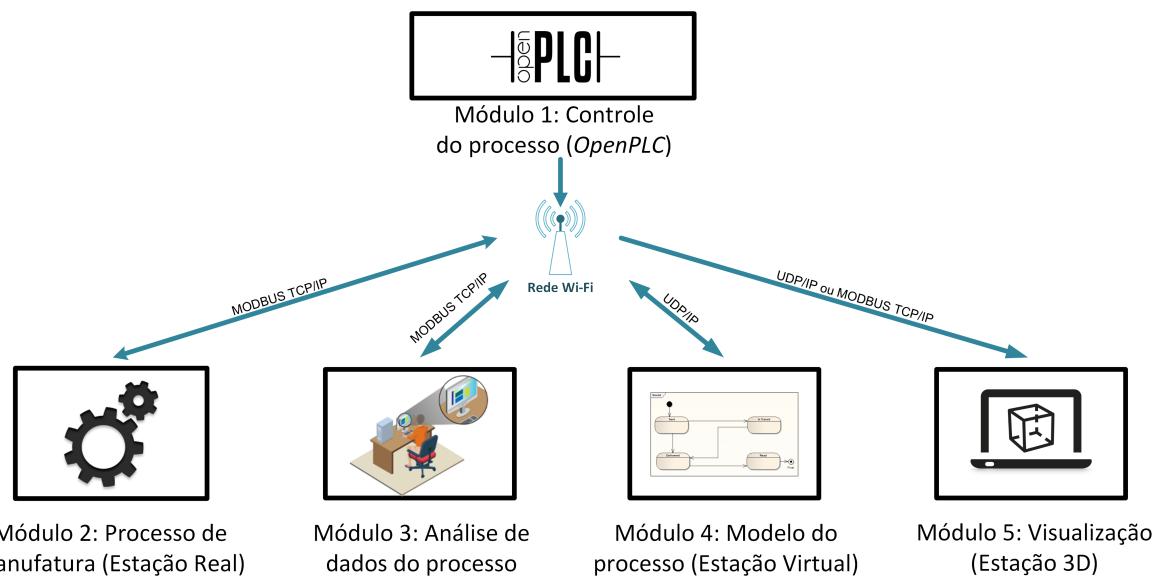
finalizá-la. Para o estudo foram utilizadas peças de Lego, no qual a montagem pode ser realizada por um trabalhador humano, por um robô ou em colaboração entre os dois. O objetivo do sistema não é apenas elevar a produtividade dos trabalhadores, mas também fornecer instruções para cada tipo de montagem que pode ser diferente, melhorando as condições de trabalho. Nesta aplicação o montador tem que executar duas tarefas, que são escolher os componentes a serem montados e montar o objeto, onde o número de componentes e os tipos de montagem dos objetos podem ser diferentes. Neste trabalho é utilizado um visor óptico que suporta exibição 3D estereoscópica apoiado na cabeça do usuário, o que mantém as mãos livres para processos manuais, que também contém uma câmera integrada utilizada para o rastreamento óptico. O aplicativo usa informações baseadas em imagens do local de trabalho para determinar a posição do usuário. Um dos objetivos centrais da pesquisa foi realizar experimentos com diferentes técnicas de visualização para estabelecer um catálogo de diferentes técnicas, tarefas, ambientes, grupos de usuários e configurações de *hardware*, e com isso poder melhorar o aplicativo para o usuário.

Malý, Sedláček e Leitao (2016) descrevem experimentos com o desenvolvimento de um aplicativo de Realidade Aumentada utilizado para tarefas de chão de fábrica como manutenção ou trabalho cooperativo de humanos e robôs. A aplicação de RA foi projetada e desenvolvida para óculos inteligentes com detecção de gestos e movimentos e para *smartphones*, onde para visualização do robô, o aplicativo de RA utiliza contorno virtual ou sem visualização do robô que consiste de uma pinça mecânica utilizada para pegar e soltar objetos. Os experimentos foram realizados com seis participantes, que realizaram 4 tarefas para avaliar as técnicas de aplicação e interação do aplicativo. A partir das avaliações são discutidos os resultados obtidos a partir da experiência dos usuários. Também são descritos os requisitos e arquitetura necessários para o desenvolvimento da aplicação e detalhes do aplicativo de RA que incluem *hardware* e *software* utilizados.

# 3 DESCRIÇÃO DA ARQUITETURA

De forma a alcançar os objetivos desta pesquisa, é proposta uma arquitetura aberta para implementação de Gêmeos Digitais, na qual são integradas as ferramentas para a execução das atividades de controle, simulação e visualização 3D. O esquema geral é mostrado na Figura 9 e será explicado detalhadamente nas seções seguintes.

Figura 9 – Arquitetura desenvolvida para a implementação de Gêmeos Digitais.



Fonte: Autoria própria.

## 3.1 CONTROLE DO PROCESSO - *OpenPLC*

O controle de processos industriais é, na maioria dos casos, implementado através de CLPs, que são computadores de processo que implementam determinadas funções, como lógicas, sequenciamento, contagem, cronometragem e aritmética. O CLP interfere no processo através da troca de informação com sensores (transdutores) e atuadores da planta, realizando o controle do processo em tempo real (FRANCHI; CAMARGO, 2008). Entretanto, os CLPs comerciais nem sempre dispõem de interfaces de comunicação apropriadas para comunicação com a nuvem e aplicativos de simulação. Desta forma, foi escolhida a plataforma *OpenPLC* (ALVES et al., 2014) como ferramenta de controle de processos.

O software *OpenPLC* é um programa de código aberto (ALVES et al., 2014) que incorpora as funcionalidades de um CLP e permite o uso de diversos tipos de *hardware*

como interfaces de Entrada e Saída (E/S). Através do editor *OpenPLC Editor*, é possível desenvolver lógicas de programação em todas as cinco linguagens definidas pela norma IEC 61131-3. Atualmente o programa oferece suporte aos seguintes SoCes: Arduino, Raspberry Pi, UniPi *Industrial Platform*, dispositivos escravos Modbus, ESP8266 e PiXtend. Ainda funciona como CLP virtual (*soft PLC*) nos sistemas operacionais Linux e *Windows*, permitindo a integração com outros *softwares* através de interfaces adequadas. Atualmente o programa está na versão 3.0, que foi a utilizada neste trabalho.

Este é um *software* licenciado na forma GPL (acrônimo inglês para *General Public License*) para uso comercial, modificação, distribuição, uso privado e uso de patente de colaboradores. O *OpenPLC* é ainda um escravo acessível através da rede industrial Modbus/TCP. Isto significa que é possível ler e escrever dados no CLP virtual através de dispositivos Modbus, bem como pode ser integrado a Sistemas de Supervisão e Aquisição de Dados (SCADA).

O servidor do *OpenPLC* é acessado através de um navegador *web*. Através da tela inicial, pode-se interagir com o CLP virtual inicializando ou paralisando o seu funcionamento, visualizando *logs* e carregando arquivos de configuração do CLP.

### 3.1.1 INTERFACE DE COMUNICAÇÃO UDP *SimLink*

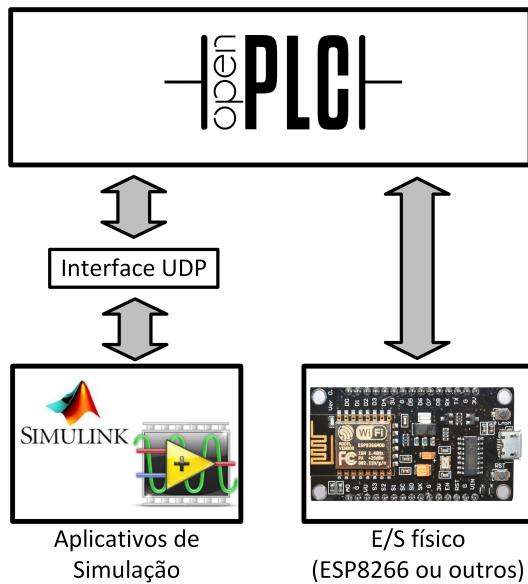
O criador do *OpenPLC* desenvolveu também uma interface, chamada *SimLink*, para permitir que o CLP virtual se comunique com aplicativos de simulação que suportam o protocolo de comunicação UDP (acrônimo inglês par *User Datagram Protocol*), como é o caso do *LabVIEW*, *Simulink*, ambiente de simulação integrado ao *software MATLAB* e outros similares. De forma análoga ao *OpenPLC*, é um *software* desenvolvido para a plataforma GNU/Linux, por isso deve ser executado em sistemas *Windows* através do compilador *Cygwin*.

A interface é parametrizada através de um arquivo de configuração, que é um breve código no qual são listadas as portas UDP que serão utilizadas na comunicação entre o *OpenPLC* e o aplicativo de simulação, bem como o tipo de sinal que trafegará (análogo ou digital). Cada endereço de E/S do CLP possui um endereço UDP associado, que deve ser declarado na interface UDP e utilizado na parametrização dos blocos de comunicação UDP do aplicativo de simulação. Esta interface possui ainda um parâmetro no qual pode ser definido um valor de atraso de comunicação. Este deve ser compatibilizado com o ciclo de *scan* do CLP.

Uma vez estabelecida a conexão entre o *OpenPLC* e o aplicativo de simulação, pode-se realizar as rotinas de integração entre o CLP e sistema de automação simulado ou modelado. Paralelamente à conexão via UDP, o CLP pode operar interfaces de E/S físicas (*hardware*), uma vez que os espaços de endereçamento das E/S físicas e virtuais são

separados. A Figura 10 apresenta um esquemático do exposto.

Figura 10 – Esquemático da comunicação implementada pela interface de comunicação UDP *SimLink*.



Fonte: Autoria própria.

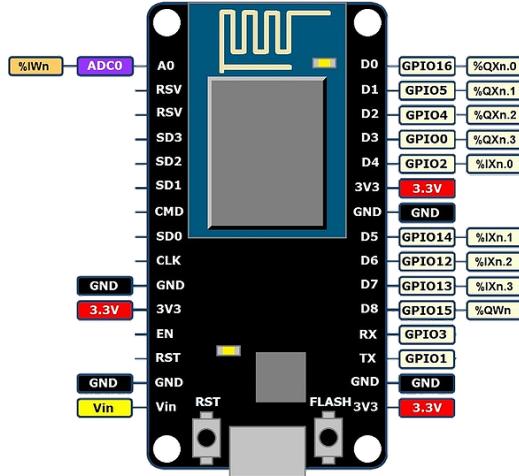
### 3.1.2 DISPOSITIVO ESP8266 COMO INTERFACE DE E/S

Um dos tipos de *hardware* suportados pelo *OpenPLC* é a família de dispositivos *System on Chip* (SoC) ESP8266. Trata-se de um *chip* cuja característica marcante é a integração *Wi-Fi* aliada à programação simplificada da interface do Arduino. Por estes motivos, tem sido um dos principais dispositivos utilizados em automação e robótica, particularmente em aplicações relacionadas à IoT (RODRIGUES, 2018).

No contexto do *OpenPLC*, o *firmware* desenvolvido para o ESP8266 disponibiliza para o usuário 4 entradas digitais, 4 saídas digitais, 1 entrada analógica e 1 saída analógica. A comunicação entre a placa e o servidor do *OpenPLC* é realizada através da rede *Wi-Fi* por meio do protocolo Modbus TCP, portanto é necessário que tanto o servidor como a placa estejam conectados à mesma rede.

A Figura 11 mostra o esquemático do uso dos pinos do SoC ESP8266 no contexto do *OpenPLC*.

Figura 11 – Esquemático de uso das entradas e saídas do SoC ESP8266 para o *OpenPLC*.



Fonte: Adaptado de Rodrigues (2018).

### 3.2 PROCESSO DE MANUFATURA

Os testes e ensaios necessários para o desenvolvimento deste projeto foram realizados no sistema modular de automação Festo MPS 200, montado no Laboratório de Automação do câmpus de Sorocaba da UNESP. Trata-se de um sistema modular no qual cada módulo realiza um trabalho específico sobre as peças, de forma a simular uma linha de produção industrial. Dispõe de equipamentos elétricos e pneumáticos e permite diversos arranjos e sequências de produção. A Figura 12 apresenta uma imagem da planta.

Figura 12 – Sistema Modular de Automação Festo MPS 200.



Fonte: Autoria própria.

### 3.3 ANÁLISE DE DADOS DO PROCESSO

No contexto da Indústria 4.0, tem-se presente a preocupação em adquirir e processar os dados dos sistemas produtivos, de forma a gerar informação útil para a avaliação do desempenho. O módulo de análise de dados tem como função recolher dados dos endereços de E/S do *OpenPLC* e realizar processamento adicional, implementando métricas de desempenho, fazendo cálculos e gerando alertas e históricos da ocorrência de anomalias.

Não existe uma metodologia única para a determinação das variáveis de interesse de processos industriais. De forma geral, busca-se avaliar o processo e observar os “indicadores-chave”, ou KPIs (acrônimo inglês para *Key Process Indicators*). De acordo com o processo, são elencados seus KPIs para a realização de medições e geração e de alertas relacionados ao desempenho.

A criação deste módulo foi realizada no aplicativo *LabVIEW*, um *software* de engenharia desenvolvido para aplicações que requerem teste, medição e controle. Também é baseado numa abordagem de programação gráfica, de forma a permitir representação de lógicas complexas em forma de diagrama, desenvolvimento de algoritmos de análise de dados e projeto de interfaces de usuário, sendo esta última a característica que mais diferencia este *software* em relação aos demais.

### 3.4 MODELO DO PROCESSO

Para viabilizar a simulação do sistema de automação é necessário representá-lo em ambiente virtual. Neste contexto, duas alternativas são possíveis: a simulação individual de componentes de sistemas de automação e a modelagem do comportamento do sistema automatizado.

A simulação individual de componentes é um processo meticuloso, no qual cada equipamento presente na planta de automação é representado no ambiente de *software*, onde se busca detalhar ao máximo as especificações de trabalho, tais como dimensões físicas e características intrínsecas. Por exemplo, a simulação de equipamentos hidráulicos inclui parâmetros como a viscosidade do óleo, temperatura de operação, aeração, diâmetros de válvulas etc.

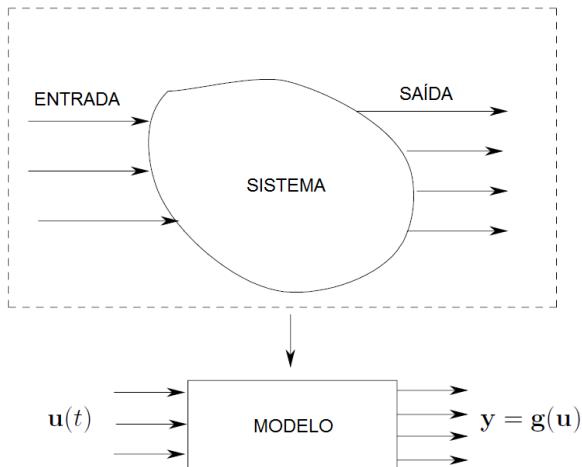
A modelagem de sistemas de automação, por sua vez, consiste resumidamente nos seguintes passos, conforme apresentado por Cassandras e Lafourne (2009)

- Obter um conjunto de variáveis mensuráveis de entrada;
- Obter um conjunto de variáveis de saída, que variem de acordo com as variáveis de entrada;

- Definir funções matemáticas que relacionem as entradas e saídas mensuradas, de forma a obter um modelo do comportamento do sistema.

Este é o meio mais simples possível de se modelar um processo (CASSANDRAS; LAFORTUNE, 2009). A Figura 13 ilustra o exposto.

Figura 13 – Esquemático da modelagem de um processo físico.



Fonte: Adaptado de Cassandras e Lafourture (2009).

Entende-se que o processo de modelagem dispensa o conhecimento dos pormenores de cada elemento da planta de automação, podendo ser realizado de forma sistemática, através do uso de formalismos e metodologias próprias. Sabe-se que muitos dos sistemas industriais não variam de acordo com o tempo, mas sim com a ocorrência de dados “eventos” (tais como um sensor ativo, botão pressionado), podendo ser assim chamados Sistemas a Eventos Discretos (SEDs)(CARDOSO; VALETTE, 1997). A modelagem de SEDs pode ser realizada através de abordagens próprias, tais como máquinas de estado, autômatos finitos, redes de Petri e outros.

O aplicativo *Simulink* é parte do *software MATLAB* e é uma poderosa ferramenta de simulação, dispondo de um ambiente de programação gráfica. As simulações são montadas através da conexão de blocos que realizam diversas funções. Este formato possibilita reusabilidade e facilidade de depuração de erros (BODEMANN; ROSE, 2004). Possui ainda diversas *toolboxes* que permitem o desenvolvimento de sistemas específicos, como mostrado na Tabela 1.

Tabela 1 – Família de *toolkits* e ferramentas do aplicativo *Simulink*.

Família	Produtos
Modelagem baseada em eventos	Stateflow
	SimEvents
Modelagem física	Simscape (Fluids, Electronics, Power Systems, Driveline)
Sistemas de controle	Simulink Control Design
	Simulink Design Optimization
	Robotics System Toolbox
	Powertrain Blockset
	Vehicle Dynamics Blockset
Processamento de sinais e comunicação sem fio	DSP System Toolbox
	Communications System Toolbox
Simulação e teste em tempo real	Simulink Real-Time

Fonte: Adaptado de Mathworks (2018).

Uma das ferramentas disponibilizadas nesse ambiente é o bloco *State Transition Table* (“Tabela de Estados-Transições” em tradução livre) pertence à biblioteca *Stateflow*. Trata-se de um recurso que permite a descrição do comportamento lógico de um sistema através de relações condicionais (*if-else*), permitindo que se definam os estados das variáveis de saída, as condições de transição e a implementação de temporização. Uma vez concluída a descrição do sistema, a ferramenta gera um diagrama ilustrativo dos estados e condições de transição, de forma semelhante a uma máquina de estados.

Este bloco possui compatibilidade com as interfaces de comunicação UDP disponíveis no *Simulink*, que por sua vez permitem a troca de informação com o servidor do *OpenPLC*. Desta forma, pode-se utilizar a máquina de estados como modelo do sistema de automação e fazer com que este modelo troque informação com o CLP continuamente. Para simplificar a sequência do texto, o bloco será citado através da sigla TET (Tabela de Estados-Transições).

### 3.5 VISUALIZAÇÃO 3D

Existem inúmeras alternativas para realizar a tarefa de implementar a visualização 3D de sistemas. No escopo deste trabalho, interessa replicar em ambiente virtual de visualização os estados da planta de automação em tempo de execução, ou seja, mostrar remotamente ao usuário o que está acontecendo no sistema (se está em movimento, em repouso, se há peças sendo processadas etc.). Nesta seção são apresentadas duas alternativas utilizadas neste trabalho.

### 3.5.1 VISUAL STUDIO

Uma alternativa aos aplicativos convencionais de simulação robótica e industrial é o desenvolvimento de aplicações próprias, que se ajustem aos requisitos particulares de cada implementação. Com base no exposto por Santos e Carvalho (2016), é possível criar sistemas com interface em realidade virtual, permitindo ao usuário visualizar a operação da planta de automação por meio de um modelo 3D animado. Os autores descrevem uma aplicação baseada na ferramenta *Visual Studio* da Microsoft, utilizando como interface gráfica o motor de jogos Microsoft XNA. A criação de um modelo 3D animado de realidade virtual requer o cumprimento das seguintes tarefas:

- Obter um modelo 3D através de CAD (acrônimo inglês para *Computer-Aided Design*);
- Preparar o modelo 3D para animação em *software* específico (AutoCAD 3Ds Max);
- Implementar o *software*, incluindo interface de usuário, carregamento e animação do modelo 3D e comunicação;

Em seguida, deve ser realizada a comunicação do aplicativo com o elemento central de controle de processos para possibilitar a replicação dos estados dos atuadores no ambiente virtual. No escopo deste trabalho, deseja-se visualizar em tempo de execução no ambiente 3D o estado do sistema real de automação. Para isso, serão necessários os modelos animados em 3D das estações de trabalho e o estabelecimento de comunicação entre o servidor do *OpenPLC* e o aplicativo mediante protocolo adequado.

### 3.5.2 FACTORY I/O

O aplicativo *Factory I/O* é uma ferramenta de simulação destinada à aprendizagem de tecnologias de automação, que permite a rápida construção de processos de fabricação virtuais utilizando peças industriais comuns, tais como esteiras, desviadores, sensores, elevadores, manipuladores e outros (REALGAMES, 2019). O programa pode ser utilizado de dois modos, sendo que no modo manual o usuário aciona cada atuador através do *mouse*. No modo automático, o acionamento dos equipamentos de campo é realizado por meio de sinais externos, que podem ser obtidos por meio de placas de aquisição de dados, comunicação com CLPs compatíveis ou por meio de protocolos de comunicação, tais como o Modbus.

A principal vantagem deste programa é a agilidade na criação dos modelos. Não é necessário conhecimento em desenho CAD ou gerenciamento de interação de objetos 3D, pois o programa já apresenta bibliotecas com peças prontas para uso, bastando posicionar os componentes no espaço de trabalho de forma correta. Como desvantagem, tem-se a impossibilidade da criação de peças customizadas ou redesenho das mesmas, limitando a

capacidade de reprodução dos aspectos visuais do ambiente de produção real dentro do aplicativo de simulação 3D.

O *Factory I/O* pode ser integrado ao *OpenPLC* como um dispositivo escravo, mantendo comunicação em tempo de execução de acordo com o tempo de comunicação cíclica (*polling*) determinado nas configurações do CLP. A Figura 14 mostra um processo industrial representado no aplicativo.

Figura 14 – Representação de processos industriais no aplicativo *Factory I/O*.



Fonte: Adaptado de RealGames (2019).

# 4 DESENVOLVIMENTO

Neste capítulo é apresentado o desenvolvimento do trabalho. São discutidos os aspectos de implementação de cada um dos módulos da arquitetura e apresentados os pormenores de cada aplicação.

## 4.1 CONTROLE DO PROCESSO - *OpenPLC*

A arquitetura tem como elemento central a ferramenta de controle de processos *OpenPLC*, já descrita no capítulo anterior. Visto que está adequada ao padrão IEC 61131-3 quanto às linguagens de programação e blocos funcionais, pode ser utilizada em substituição aos CLPs convencionais disponíveis nas instalações da Universidade, por apresentar a conectividade em redes *Wi-Fi* e suporte ao protocolo UDP, que será essencial para o funcionamento de outros módulos. A linguagem de programação *Ladder* foi escolhida como padrão para a escrita dos programas.

Através do uso do *software* de interface UDP, os dados do CLP podem ser acessados por diversos clientes (“*listeners*”) UDP ao mesmo tempo, ou seja, diversos aplicativos podem receber os dados de um mesmo *bit* de saída do CLP simultaneamente. Esta característica favorece a modularidade desejada para a arquitetura, pois assim cada módulo pode receber dados do CLP a partir de um cliente UDP separado. Obviamente os *bits* de entrada não podem ser acessados da mesma forma, pois uma fonte de dados sobreescriveria a outra constantemente, provocando mau funcionamento do sistema.

Por questões de implementação do *OpenPLC* e do aplicativo de interface *SimLink*, a comunicação UDP tem como alvo os primeiros *bytes* do espaço de endereços de E/S, iniciando-se pelo *byte* 0. Por ser um arquivo à parte que acessa os *bytes* do *OpenPLC*, os dispositivos UDP não são declarados na página de escravos, pois a própria interface gerencia o envio e recebimento de dados dos aplicativos remotos (*Simulink* e *Visual Studio*). Os escravos, por sua vez, têm seus dados de E/S alocados em *bytes* a partir do 100, seguindo a ordem de declaração dos mesmos. São considerados dispositivos escravos tanto os ESP8266 quanto os computadores que executam o aplicativo *Factory I/O*. A Tabela 2 apresenta os dispositivos escravos e seus respectivos endereços de E/S.

Tabela 2 – Dispositivos escravos declarados no *OpenPLC*.

Nome	Tipo	Bytes de E/S
ESP - <i>Sorting</i>	ESP8266	IX100.0 - IX100.7 QX100.0 - QX100.7
<i>Factory I/O - Sorting</i>	Dispositivo TCP genérico	IX101.0 - IX101.7 QX101.0 - QX101.7
ESP - <i>Testing</i>	ESP8266	IX102.0 - IX102.7 QX102.0 - QX102.7
<i>Factory I/O - Testing</i>	Dispositivo TCP genérico	IX103.0 - IX103.7 QX103.0 - QX103.7

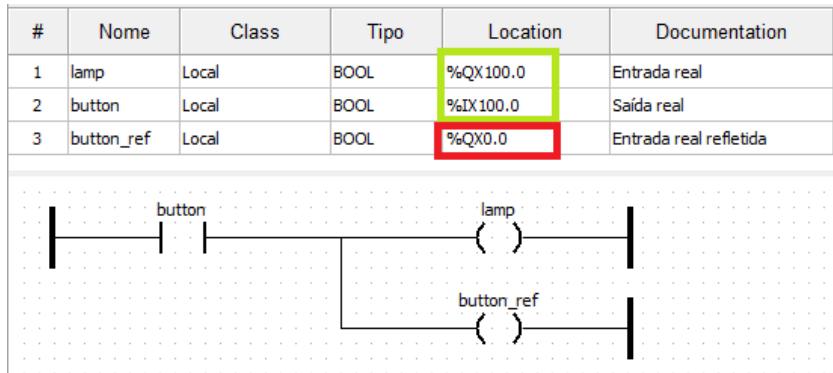
Fonte: Autoria própria.

De maneira geral, tem-se que:

- O processo de automação real é regido pela lógica expressa em diagrama *Ladder* definida no ambiente do CLP, no qual são utilizados apenas os dados de sensores e atuadores reais;
- O modelo lógico recebe os sinais dos sensores reais refletidos para a comunicação com a parte virtual e retorna para o CLP os estados dos atuadores virtuais regidos pela TET;
- Os aplicativos de visualização 3D recebem os sinais dos sensores reais refletidos para a comunicação com a parte virtual para que sejam gerados os objetos de processo. São refletidos também os dados dos atuadores reais para que se possa replicar o comportamento dos mesmos nos ambientes 3D.

É necessário que os sinais dos sensores e atuadores reais sejam refletidos para a comunicação com a parte virtual. Isto se dá no diagrama *Ladder* através de uma lógica simples de replicação, mostrada na Figura 15. Neste exemplo, o sinal de um botão real (entrada) é utilizado para acionar uma saída real e ao mesmo tempo é refletido para o ambiente virtual, a fim de que possa simultaneamente disparar uma transição na TET. Este procedimento garante que a TET será alimentada com os mesmos sinais que alimentam a lógica de programação do sistema real (diagrama *Ladder*). Assim, dispondo dos mesmos sinais de entrada, tanto a planta quanto seu modelo devem produzir as mesmas variáveis de saída.

Figura 15 – Diagrama *Ladder* utilizado para refletir um sinal de entrada real para o ambiente virtual.

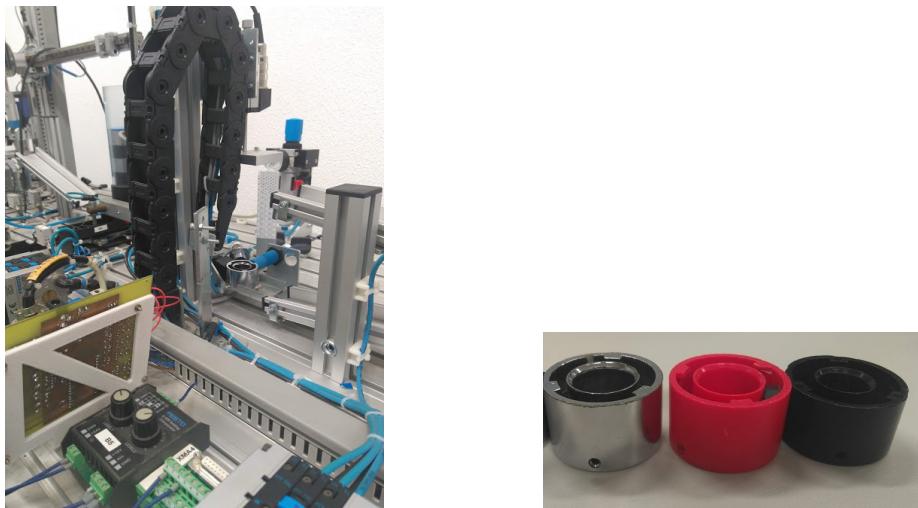


Fonte: Autoria própria.

## 4.2 PROCESSO DE MANUFATURA

O processo de manufatura utilizado no arranjo experimental é composto por duas estações modulares do conjunto Festo MPS 200, que são as estações *Testing* e *Sorting*. A estação *Testing* é responsável por receber peças plásticas de um processo produtivo e repassá-las para a estação seguinte. A Figura 16 mostra a estação *Testing* e a especificação das peças que serão utilizadas no sistema de seleção e descarte.

Figura 16 – Estação *Testing* e peças que serão utilizadas no processo.



Fonte: Autoria própria.

Uma vez terminado o processo na estação *Testing*, ela é encaminhada à estação *Sorting*, que é responsável por fazer a separação das peças por cor em três bandejas distintas. As peças são identificadas a partir de um arranjo de sensores que inclui um sensor de presença, um sensor fotoelétrico (utilizado para identificar coloração) e um sensor capacitivo (utilizado para identificar se o material é metálico). A Figura 17 apresenta o módulo.

Figura 17 – Módulo de seleção de peças por característica (*Sorting*).



Fonte: Autoria própria.

A relação entre as leituras dos sensores e os tipos de peça é apresentada na Tabela 3.

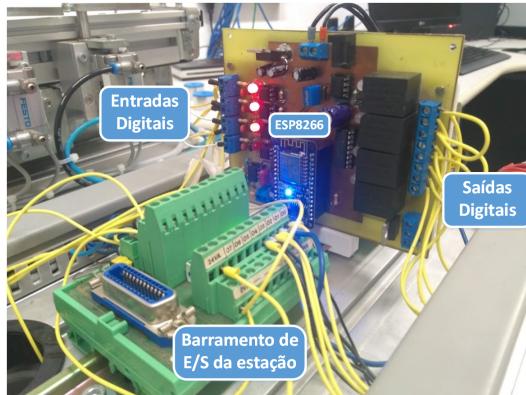
Tabela 3 – Correlação entre os tipos de peças e as possíveis leituras dos sensores da estação *Sorting*.

<b>Tipo de Peça</b>	<b>Sensor de Presença</b>	<b>Sensor Fotoelétrico</b>	<b>Sensor Capacitivo</b>
<b>Preta</b>	1	0	0
<b>Vermelha</b>	1	1	0
<b>Metálica</b>	1	1	1

Fonte: Autoria própria.

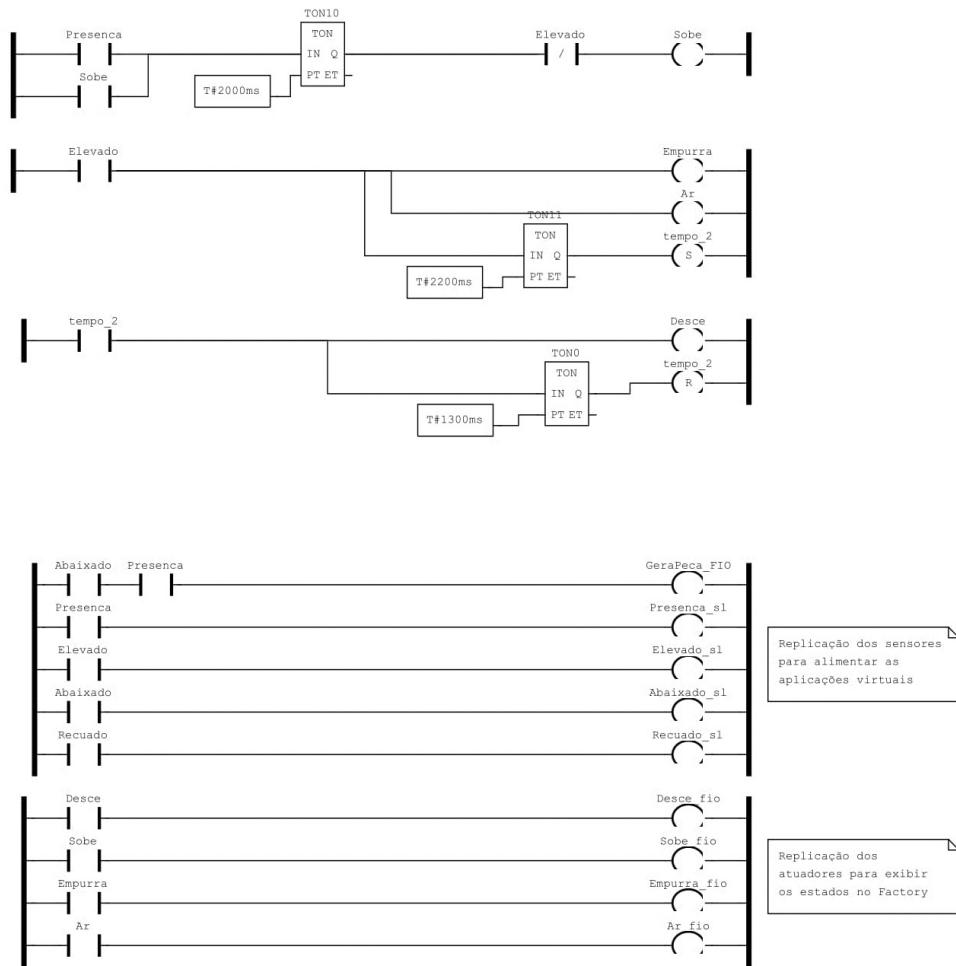
Para permitir que as estações sejam controladas pelo *OpenPLC*, há necessidade de hardware adicional. Placas de acondicionamento de sinais foram instaladas nos terminais de E/S das estações de trabalho. A Figura 18 mostra uma placa instalada.

Figura 18 – Placa de acondicionamento de sinais para uso do SoC ESP8266 como interface de E/S.



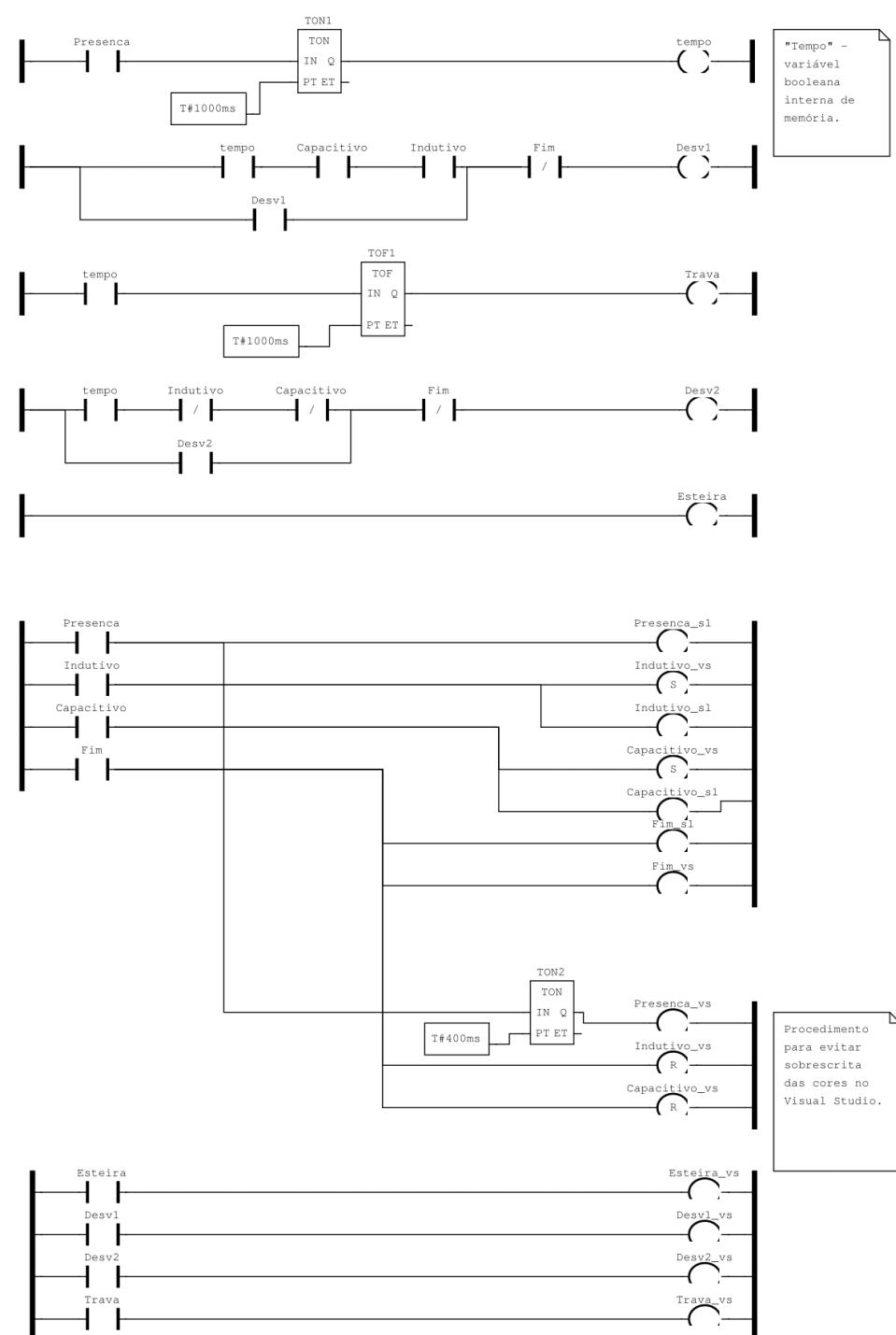
Fonte: Autoria própria.

Com o *hardware* de controle instalado, foram desenvolvidos os códigos em linguagem *Ladder* para o controle do processo de automação. Os *bits* utilizados na comunicação UDP também estão declarados no diagrama, de acordo com a separação dos espaços de endereçamento mencionada na seção 4.1. As Figuras 20 e 19 apresentam os diagramas *Ladder* para as estações *Sorting* e *Testing*, respectivamente. Nota-se ao final dos diagramas a replicação de sinais para o espaço de endereçamento dos módulos virtuais. A listagem completa das entradas e saídas é apresentada no Apêndice A.

Figura 19 – Diagrama *Ladder* desenvolvido para o controle da estação *Testing*.

Fonte: Autoria própria.

Figura 20 – Diagrama Ladder desenvolvido para o controle da estação Sorting.



Fonte: Autoria própria.

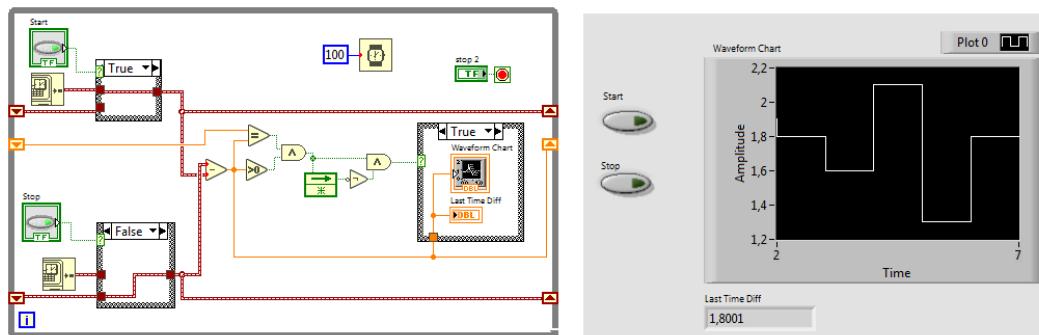
## 4.3 ANÁLISE DE DADOS DO PROCESSO

O desenvolvimento do módulo de análise de dados do processo envolve três componentes fundamentais: a coleta de dados, a realização de cálculos e a geração de históricos, métricas e alertas. Todos são implementados em um único programa, chamado no ambiente *LabVIEW* de VI (acrônimo inglês para *Virtual Instrument*).

A coleta de dados é realizada por meio do acesso aos *bytes* de E/S do *OpenPLC* utilizando o protocolo de comunicação Modbus TCP/IP. O *LabVIEW* disponibiliza blocos para acesso a variáveis de entrada (*inputs*) e saída (*coils*) discretas. É necessário declarar o endereço IP do servidor do *OpenPLC*, a porta Modbus utilizada por ele (por padrão, o *OpenPLC* utiliza a porta 502) e o endereço Modbus desejado. Estes blocos são executados cicличamente de forma a manter atualizados os estados das variáveis de E/S do *OpenPLC* para exibição no painel do VI e para a realização dos cálculos.

Através dos sinais dos sensores, é possível mensurar tempos de deslocamento das peças, que proporcionam indicativos de desempenho do sistema como um todo. O *LabVIEW* não possui mecanismos de cronometragem preconfigurados, mas apresenta os elementos necessários para criá-los. A lógica básica de contagem de tempo entre dois eventos (habilitação de sensores, por exemplo) é realizada pelo diagrama apresentado na Figura 21. Nota-se a presença de dois *loops* com registradores de deslocamento, cada um correspondendo a um sensor. Quando o primeiro evento ocorre, o primeiro registrador de deslocamento armazena a data/hora da ocorrência. Quando o segundo evento ocorre, o segundo registrador de deslocamento armazena a data/hora de forma similar. Em seguida, é realizado o cálculo do tempo do segundo evento menos o tempo do primeiro evento. Esta diferença é plotada num gráfico de histórico e mostrada na forma de *string*.

Figura 21 – Medição de intervalo de tempo entre dois eventos no aplicativo *LabVIEW*.



Fonte: Autoria própria.

Como introduzido na Seção 3.3, o desempenho do processo é analisado de acordo com o monitoramento de KPIs, ou seja, indicadores-chave. Os KPIs escolhidos para avaliação das estações de trabalho são apresentados na Tabela 4.

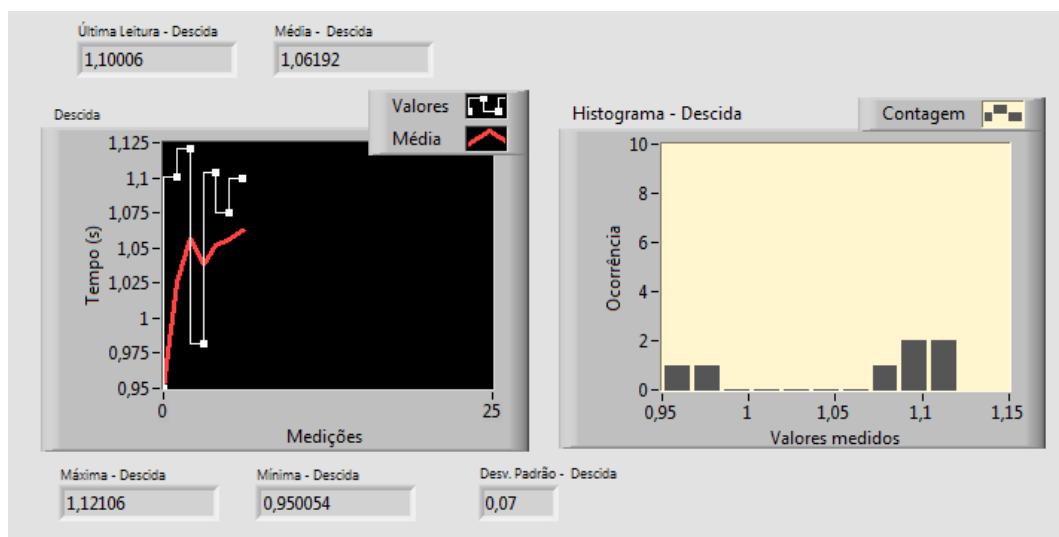
Tabela 4 – Indicadores-chave escolhidos para a avaliação das estações de trabalho *Testing* e *Sorting*.

Estação	KPI	Condição inicial	Condição final
<i>Testing</i>	Tempo de subida da plataforma	Sensor de plataforma abaixada	Sensor de plataforma elevada
	Tempo de descida da plataforma	Sensor de plataforma elevada	Sensor de plataforma abaixada
<i>Sorting</i>	Tempo de processamento de peças cor prata	Sensor de presença	Sensor de fim-de-curso
	Tempo de processamento de peças cor preta	Sensor de presença	Sensor de fim-de-curso
	Tempo de processamento de peças cor prata	Sensor de presença	Sensor de fim-de-curso

Fonte: Autoria própria.

Replicando-se o método de medição de tempo entre dois eventos, é possível obter os KPIs desejados. Com base nestes dados, é possível verificar a dispersão dos resultados obtidos. Para cada KPI, é realizado o armazenamento na forma de *array* e o cálculo do valor médio e desvio padrão à medida que novas medições são realizadas. É apresentado ainda o histograma, para que se possa visualizar os intervalos de medida com maior reincidência. A Figura 22 mostra os cálculos de média, desvio padrão e histograma, bem como a geração dos gráficos.

Figura 22 – Medição de um KPI no módulo de análise de dados.



Fonte: Autoria própria.

Além dos KPIs da planta real, tem-se ainda outra medida capaz de apontar desvios ou defeitos no sistema de produção. Uma vez que as TETs replicam o funcionamento da planta real no ambiente virtual, é possível comparar os dados de saída das TETs (atuadores virtuais) com o estado dos atuadores reais. Desta forma podem ser detectadas

discrepâncias ou eventuais atrasos na planta real em relação ao modelo ideal (TET), que podem indicar oscilações de *performance* ou falhas.

A geração de alarmes é baseada na temporização implementada. Cada etapa de processo possui um tempo médio e um desvio padrão, que são o referencial. Uma vez que ocorra atrasos de pequena ordem, estes são lidos e listados como avisos. Ocorrendo atrasos de grande ordem, estes são listados como erros, e esta ocorrência pode gerar, além da geração do alerta, o envio de comandos do aplicativo de análise de dados para a central do *OpenPLC*. O LabVIEW tem a capacidade de escrever *bits* de entrada no *OpenPLC*, assim se pode criar rotinas de emergência no diagrama *Ladder* que implementem medidas protetivas diversas.

## 4.4 MODELO DO PROCESSO

Em atenção ao requisito de modularidade do sistema, convém realizar separadamente a modelagem dos estados dos processos *Testing* e *Sorting* no ambiente *Simulink*. O passo-a-passo da criação de um modelo do processo com a ferramenta de Diagrama de Estados e Transições é a seguinte:

1. Identificar os estados e transições: analisar o comportamento do sistema e observar quais são os estados e condições de transição (eventos) que regem o funcionamento esperado da planta;
2. Realizar a tomada dos tempos de transição: medir o tempo aproximado para eventos como o deslocamento de peças entre trilhos, elevação de plataformas e outros;
3. Transcrever cada estado no diagrama, definindo a condição de cada atuador (ligado ou desligado) enquanto o sistema estiver no mesmo;
4. Transcrever as condições de transição (tempo, acionamento de sensores) e o sequenciamento (definir o subsequente de cada estado);
5. Na primeira tentativa de execução do diagrama, confirmar quais são as variáveis de entrada e as de saída;
6. Configurar os blocos UDP que recebem e enviam dados para o CLP.

Através da interface UDP, o CLP fornece para o aplicativo de simulação os estados dos sensores da planta em tempo de execução. Com base nestes dados, o Diagrama de Estados e Transições é executado e gera em sua saída os estados dos atuadores. Esta informação pode ser transmitida de volta para o CLP ou enviada para outros módulos através dos blocos UDP *Send* disponíveis no *Simulink*. O exemplo mais simples de detecção

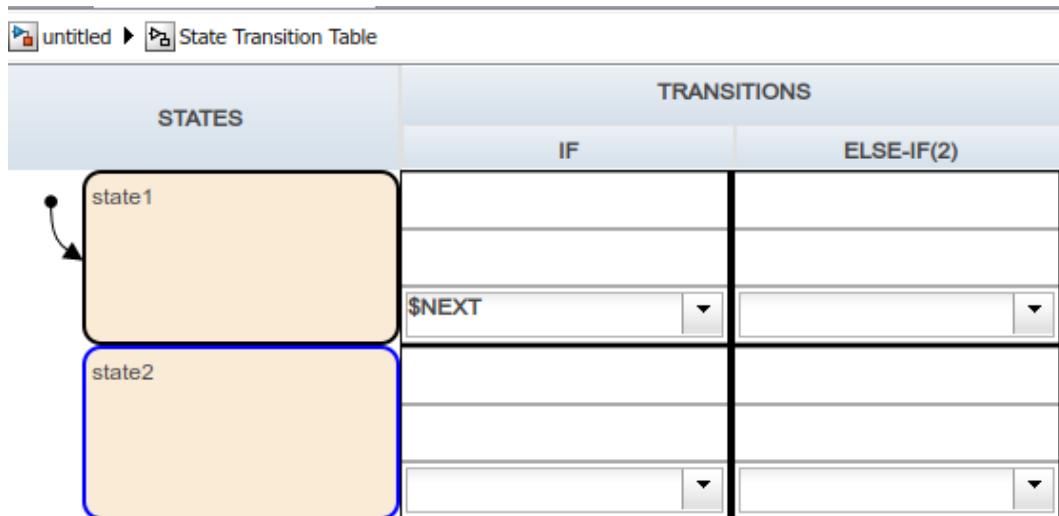
de falhas baseado na execução do modelo seria a comparação direta entre os estados dos atuadores reais e os estados gerados pelo aplicativo de simulação, entretanto neste trabalho deseja-se explorar alternativas mais elaboradas.

Uma vez que não há sincronização de tempo entre o *OpenPLC* e o aplicativo de simulação, é necessário utilizar um bloco adicional no simulador para reduzir a velocidade de simulação, de forma que se obtenha uma base de tempo próxima ao tempo real (*wall clock*). Apesar de não haver sincronização, as bases de tempo do *Simulink* e do *OpenPLC* se tornaram suficientemente próximas para o contexto deste trabalho, de forma que se tornou possível verificar visualmente a mudança dos estados do sistema, bem como reduziu-se o consumo de processador e memória do computador que executa a simulação. Outra observação importante é que o *Simulink* deve ser ajustado para executar a simulação por tempo indeterminado (“*Inf*”). Para garantir a corretude das medições de tempo na TET (*delay* e outras), é necessário ajustar as configurações do *solver*, utilizando a compilação com passo de tempo fixado (*fixed-step*).

#### 4.4.1 CRIAÇÃO DE TETs

Para criar uma TET, basta iniciar um modelo no aplicativo *Simulink* e inserir o bloco através da biblioteca. Após inserido no modelo, basta dar dois cliques sobre o bloco para que se abra sua janela de edição. A Figura 23 mostra a inicialização do bloco.

Figura 23 – Inicialização de uma TET.



Fonte: Autoria própria.

Para fins de demonstração da ferramenta, será realizada a modelagem de um processo-exemplo com apenas dois estados, uma variável de entrada e uma de saída. Quando a entrada for acionada (assumir valor 1), a saída será acionada por cinco segundos, retornando para o estado desligado após este período de tempo.

Inicialmente deve-se definir os estados do sistema. Os retângulos coloridos na coluna “*States*” são os espaços onde tais estados são descritos. A primeira linha de texto expressa o título. Abaixo desta linha, deve ser definido o valor que cada variável de saída recebe durante este estado na forma:

```
entry: <nome da variável> = <valor>;
```

Na coluna “*Transitions*” são definidas as condições de transição. Cada estado pode ter um ou mais estados como sucessores, seguindo a lógica condicional *if-elseif*. Por padrão há duas colunas, porém podem ser adicionadas mais colunas se necessário.

A sintaxe dos comandos de transição é feita na primeira célula, na forma:

```
[<nome da variável = <valor>]
```

Na última célula é apresentado um seletor onde deve ser indicado o estado que será habilitado quando ocorrer a condição de transição descrita. Existem dois comandos predefinidos, “*SELF*” e “*NEXT*” que indicam manter-se no mesmo estado e ir para o próximo estado listado, respectivamente. A transição também ser disparada após um certo período de tempo (segundos), definido através da sintaxe:

```
after(<valor>, sec)
```

Desta forma, o passo-a-passo da implementação do processo-exemplo é:

1. Inicializar o estado “desligado”:

*desligado*

```
entry: saída = 0;
```

2. Inicializar o estado ”ligado”:

*ligado*

```
entry: saída = 1;
```

3. Definir a primeira transição na coluna *if* para ser disparada quando a entrada é acionada:

```
[entrada > 0]
```

e selecionar o estado “ligado” como próximo.

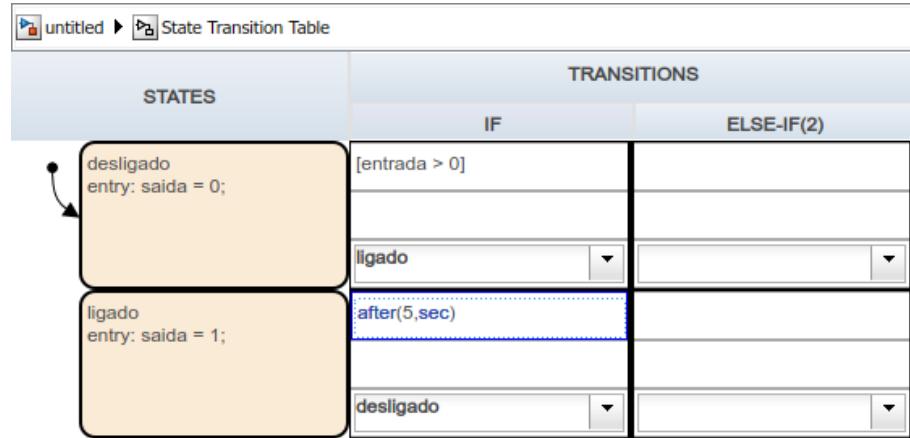
4. Definir a segunda transição na coluna *if* para ser disparada após 5 segundos:

```
after(5, sec)
```

e selecionar o estado “ligado” como próximo.

O diagrama final é apresentado na Figura 24.

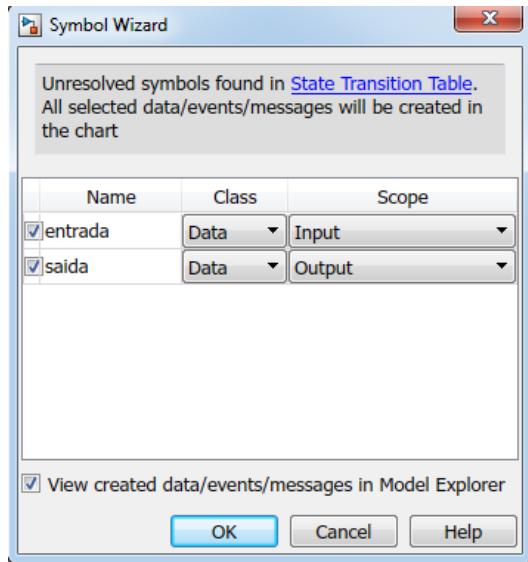
Figura 24 – TET elaborada para o processo-exemplo.



Fonte: Autoria própria.

Com a tabela pronta, podemos rodar o modelo. A primeira execução apresentará um erro, porque as variáveis ainda não foram definidas pelo *Simulink*. Será exibida a janela apresentada na Figura 25.

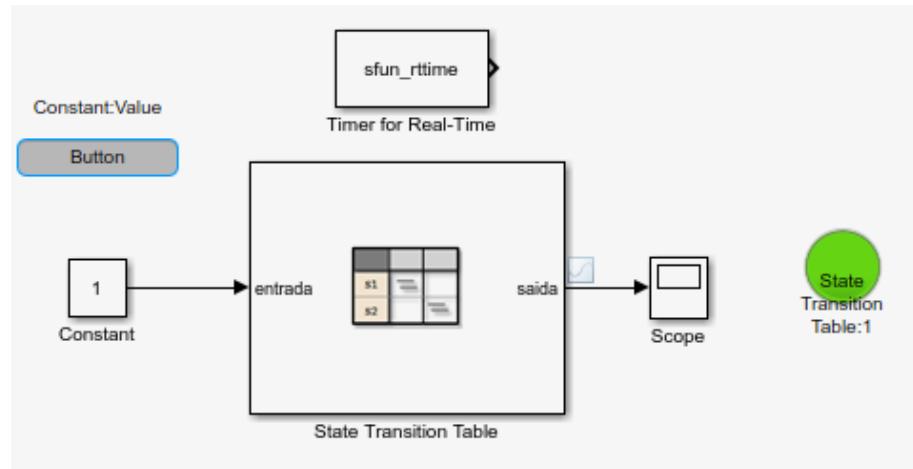
Figura 25 – Definição das variáveis da TET.



Fonte: Autoria própria.

Se a escrita dos estados e transições foi correta, as variáveis já aparecerão com o *Scope* correto (entrada ou saída), neste caso basta clicar em “*Ok*” e dar sequência à simulação. Note que após esta primeira execução, o bloco será mostrado no *Simulink* com as portas de entrada e saída disponíveis para a conexão dos demais elementos (chaves, blocos de comunicação etc.). Basta então inserir o bloco para controle do tempo de execução, a chave para controlar a variável de entrada e o mostrador para visualizar a variável de saída. Finalmente, o tempo de execução padrão (10.0) deve ser alterado para “*Inf*” para que a simulação seja executada indefinidamente. O modelo finalizado é apresentado na Figura 26.

Figura 26 – Simulação completa do modelo do processo-exemplo em TET.



Fonte: Autoria própria.

#### 4.4.2 CRIAÇÃO DAS TETs DAS ESTAÇÕES *TESTING* E *SORTING*

Uma vez desenvolvida a programação das estações, a etapa seguinte corresponde à modelagem da planta de automação na forma de Diagrama de Estados e Transições. Foram desenvolvidos dois blocos separados (sendo que um representa cada estação). Os blocos referentes às duas estações de trabalho foram testados e validados *off-line*, utilizando variáveis de entrada fornecidas manualmente. Uma vez verificada a corretude dos modelos, foi implementada a comunicação UDP. O código que foi implementado nos diagramas e as máquinas de estado resultantes são apresentados no Apêndice B.

A TET da estação *Testing* foi modelada considerando-se os seguintes estados:

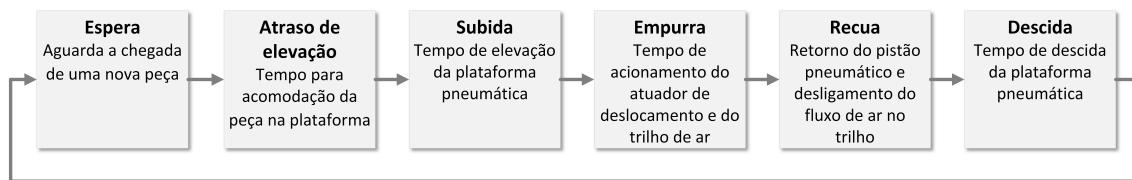
- Espera: estado inicial de espera de uma nova peça;
- Atraso de elevação: foi inserido um atraso de tempo de 2 segundos para impedir a elevação imediata da plataforma quando há chegada de novas peças, isto serve para

evitar colisão com outros atuadores quando as peças provém das estações anteriores ou acidentes quando a inserção das peças é feita manualmente;

- Subida: tempo necessário para que a plataforma pneumática faça a elevação da peça;
- Empurra: estado no qual a peça é deslocada da plataforma para o trilho de ar, através de um pistão. O trilho de ar reduz o atrito da peça com o escorregador metálico, agilizando sua chegada na estação *Sorting*. Este estado permanece ativo por dois segundos, para que haja tempo suficiente para o deslocamento completo da peça;
- Recua: tempo para que o pistão recue à posição original antes da descida da plataforma;
- Descida: tempo de retorno da plataforma pneumática à posição original. Após este estado, a estação retorna ao estado “Espera”.

A Figura 27 apresenta de maneira esquemática o exposto.

Figura 27 – Diagrama esquemático da sequência de estados utilizada para modelagem da estação *Testing*.



Fonte: Autoria própria.

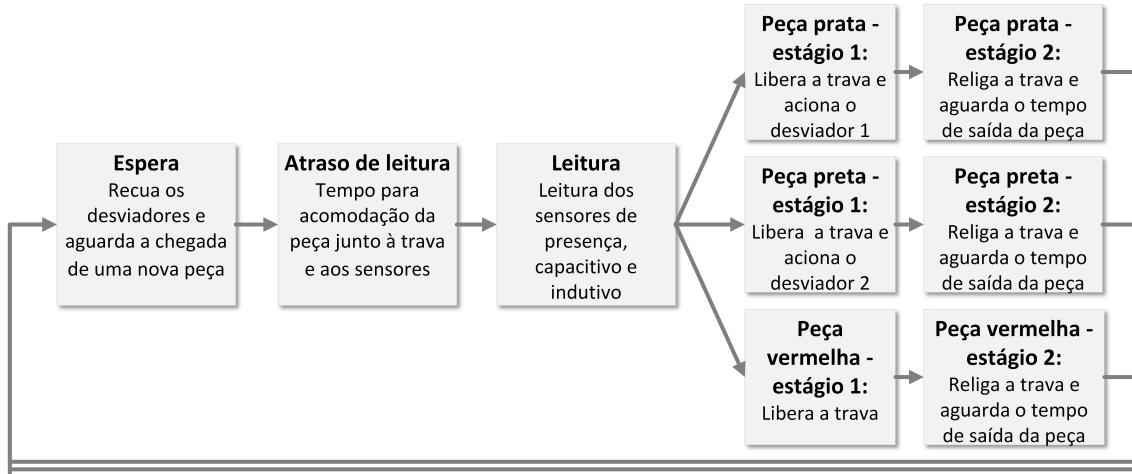
A TET da estação *Sorting* foi modelada considerando-se os seguintes estados:

- Espera: estado inicial de aguardo de uma nova peça;
- Atraso de leitura: quando a peça chega ao conjunto de sensores, convém que ela seja mantida paralisada por um segundo, de forma que haja tempo hábil para que a peça pare seu movimento e todos os sensores sejam excitados de acordo com o material da peça;
- Leitura: este estado não corresponde a um movimento específico da peça na estação de trabalho. Nele a TET aciona os estados correspondentes à cor da peça de acordo com as informações dos sensores. Este estado é executado no tempo de um único ciclo de leitura da TET (este tema será tratado adiante), tendo a função de “rotar” a TET;
- Estágio 1: este estado tem a duração de 300 milissegundos, sendo que há liberação da trava de peças (bloqueador) e o desviador correspondente à cor da peça é habilitado;

- Estágio 2: neste estado, é realizada a reativação da trava de peças e é implementado o atraso de tempo (*delay*) correspondente ao tempo médio de deslocamento da peça até o sensor de saída. Ao final deste tempo, a TET retorna para o estado “Espera”.

A Figura 28 apresenta de maneira esquemática o exposto.

Figura 28 – Diagrama esquemático da sequência de estados utilizada para modelagem da estação *Sorting*.



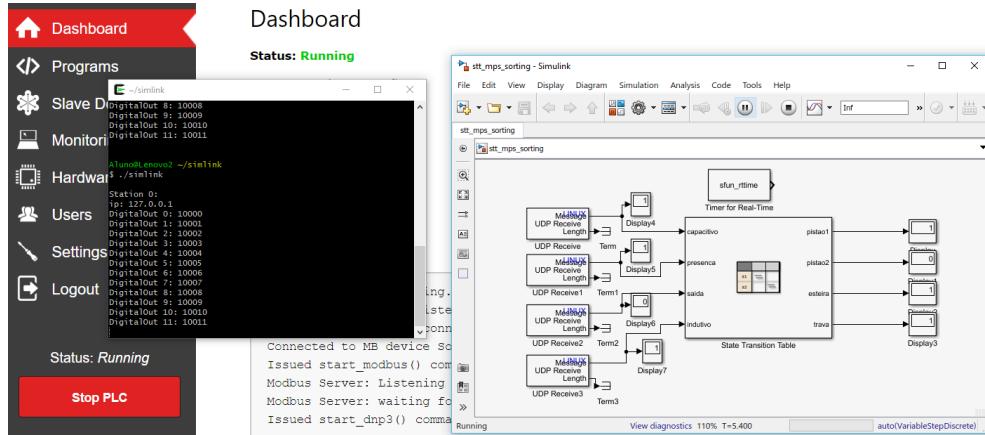
Fonte: Autoria própria.

O modelo das estações inclui os tempos médios de funcionamento dos atuadores. A medição dos tempos foi realizada manualmente para os testes iniciais e depois refinada com o uso do módulo de análise de dados do processo. Desta forma, as TETs correspondem ao funcionamento nominal do processo, sob as condições normais de pressão pneumática e suprimento de energia. Os dados gerados pelas TETs servem como referência na comparação com o processo real, possibilitando a identificação de anomalias quando se compara a temporização do processo em curso com os dados do modelo.

Uma outra forma de se utilizar as TETs, que não foi utilizada neste projeto, é a replicação do estado atual da planta. Neste caso, seriam usados apenas os dados de sensores, atuadores e atrasos de tempo definidos pelo programador (como por exemplo o atraso de leitura ou o tempo de acionamento do trilho de ar). Neste tipo de implementação, não há medição dos tempos de funcionamento normal do processo e a TET reflete exatamente o que está acontecendo na planta, porém não há geração de dados que possam ser utilizados para fins comparativos, como desejado neste projeto.

A Figura 29 mostra a execução da TET referente à estação *Sorting* operando em conjunto com o *OpenPLC* através da comunicação UDP. A TET da estação *Testing* é executada de forma análoga.

Figura 29 – TET em operação com dados recebidos via UDP.



Fonte: Autoria própria.

## 4.5 VISUALIZAÇÃO 3D

A visualização 3D da estação é realizada por diferentes aplicações nas estações *Testing* e *Sorting*. Nesta seção são apresentados os aspectos de implementação de cada uma delas.

### 4.5.1 VISUAL STUDIO

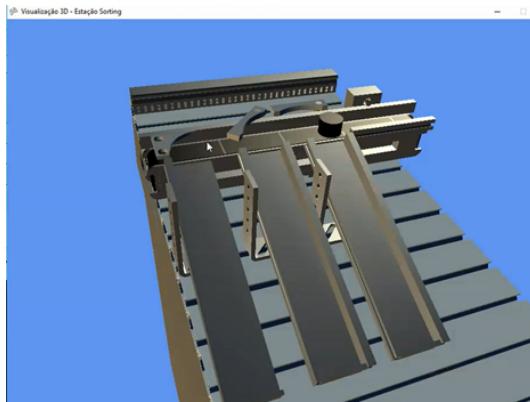
Uma das ferramentas de visualização 3D do sistema de automação foi implementada com base no trabalho desenvolvido por Santos e Carvalho (2016), uma vez que a proposta é implementada numa plataforma com maior flexibilidade quanto à implementação de padrões de comunicação. Houve a necessidade de se desenvolver um novo mecanismo de comunicação, condizente com o *OpenPLC*.

A troca de informações entre o CLP e o aplicativo é definida na classe “ClienteTCP”, na qual estão contidos os parâmetros de endereçamento e obtenção dos dados. Foi necessário reescrever esta classe, para que fosse possível obter os dados do *OpenPLC* através de mensagens UDP. Toda a implementação de código foi realizada no ambiente *Visual Studio* em linguagem de programação C#.

Foi desenvolvido e testado um código na linguagem C# para implementar um cliente (*listener*) UDP, como descrito na seção 4.5.1.1 que habilita a comunicação através do protocolo UDP, compatível com o *OpenPLC*. Assim, foi possível monitorar a estação *Sorting* através do aplicativo de realidade virtual, uma vez que seu modelo CAD 3D animado está finalizado. Foi necessário implementar um atraso na comunicação do sinal do sensor de presença para o aplicativo de visualização, de forma a evitar erros na identificação da cor da peça. Uma vez que o sensor de presença é o primeiro a ser acionado, sem este atraso, o programa identificava a cor antes que houvesse tempo hábil de leitura dos demais

sensores, definindo erroneamente a cor da peça de trabalho. A Figura 30 mostra o aplicativo de visualização 3D em execução.

Figura 30 – Aplicativo desenvolvido para a visualização 3D da estação Festo MPS *Sorting*



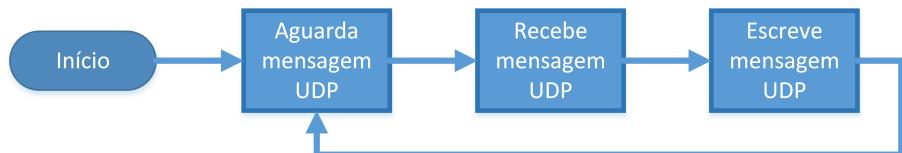
Fonte: Autoria própria.

#### 4.5.1.1 EDIÇÃO DA CLASSE “ClienteTCP”

Para que o *OpenPLC* fosse capaz de se comunicar com o aplicativo de visualização 3D, foi necessário implementar um cliente (*listener*) UDP.

O primeiro passo foi desenvolver um cliente *stand-alone*, na forma de aplicativo de *console*. Este aplicativo tem o objetivo de transcrever as mensagens UDP recebidas numa determinada porta UDP na tela de exibição. A Figura 31 mostra um esquemático simplificado do funcionamento do cliente.

Figura 31 – Aplicação cliente UDP.

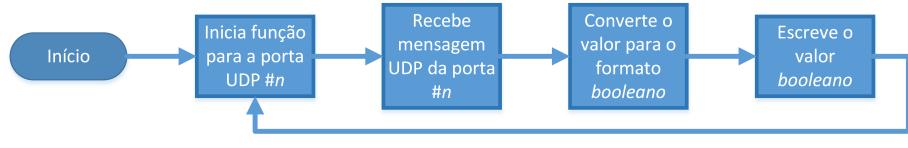


Fonte: Autoria própria.

Em seguida, o cliente foi generalizado na forma de função para possibilitar a leitura sequencial de diversas portas UDP, ainda como aplicação *stand-alone* de *console*. Seu objetivo é receber os valores das portas e replicá-los na tela. Esta função foi elaborada de forma a retornar apenas um valor *booleano true* ou *false* de acordo com os dados da mensagem UDP recebida. Isto foi feito para simplificar a leitura dos dados recebidos, uma vez que os dados oriundos do *OpenPLC* são escritos no formato *uint16* (inteiro sem sinal de dois *bytes*), enquanto as variáveis do aplicativo de visualização são definidos na forma *booleana*. Assim, a própria função se encarrega de realizar a adaptação do formato de dados, facilitando a atribuição dos valores de variáveis no ambiente das Variáveis Globais

do *software* original. A Figura 32 mostra um esquemático simplificado do funcionamento do cliente.

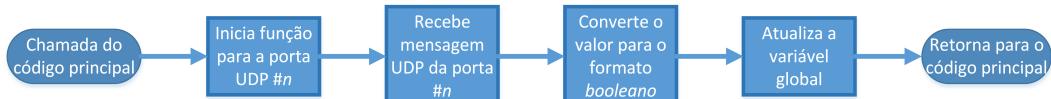
Figura 32 – Aplicação cliente UDP para múltiplas portas.



Fonte: Autoria própria.

Por fim, o código foi inserido na classe “ClienteTCP” do *software* original. O nome da classe não foi alterado para evitar erros de referência no restante do código original. A Figura 33 mostra o funcionamento da função de leitura UDP dentro do contexto do aplicativo de visualização 3D.

Figura 33 – Aplicação cliente UDP final.



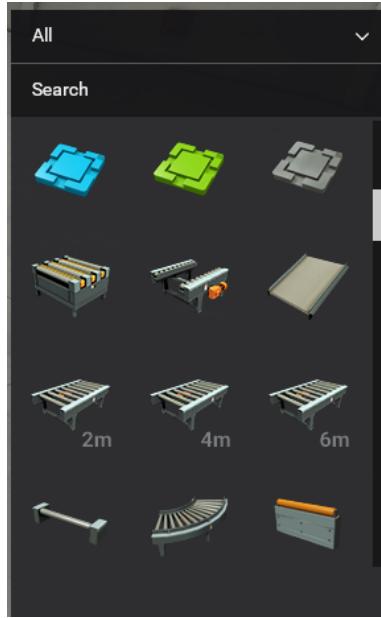
Fonte: Autoria própria.

Uma vez que os autores desenvolveram os recursos 3D para a estação *Sorting*, o cliente UDP foi integrado ao código do *software* e foi possível replicar o funcionamento real da estação no Modo Supervisório.

#### 4.5.2 FACTORY I/O

A replicação 3D do sistema de automação no ambiente do *Factory I/O* é realizada através da criação de uma planta virtual similar à física, utilizando as peças disponíveis na biblioteca do *software*. São disponibilizados componentes típicos de processos industriais, tais como esteiras, desviadores, manipuladores robóticos e sensores. A Figura 34 mostra uma parte da biblioteca de dispositivos.

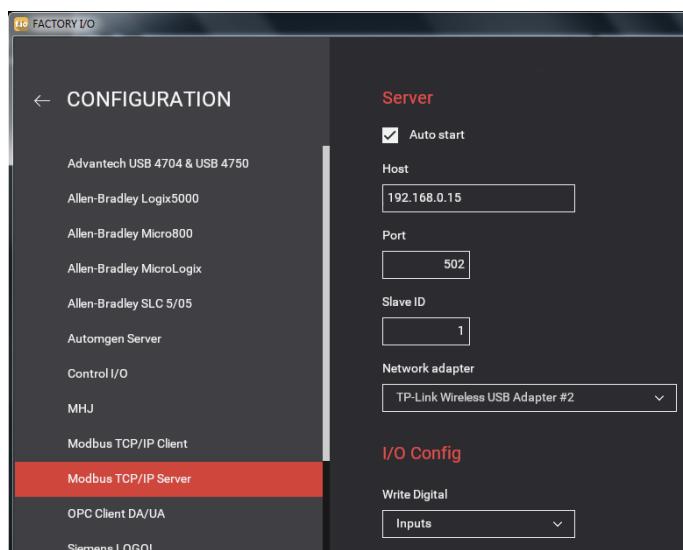
Figura 34 – Biblioteca de componentes de processos industriais do aplicativo *Factory I/O*.



Fonte: Autoria própria.

O *software* possui diversas formas de comunicação, sendo que no caso do *OpenPLC* o método adequado é a comunicação Modbus TCP/IP, que permite a troca de informação entre o aplicativo e o CLP através da rede de comunicação Ethernet. O *Factory I/O* inicia um servidor (escravo) Modbus, que pode ser personalizado de acordo com o tamanho e tipos de variáveis presentes no processo de automação. Podem ser lidas e escritas variáveis *booleanas* (discretas) ou *holding registers* (contínuas), cujas quantidades são definidas nas configurações. A Figura 35 apresenta a página de configuração do escravo Modbus.

Figura 35 – Configuração do módulo escravo Modbus do aplicativo *Factory I/O*.



Fonte: Autoria própria.

Uma vez que o processo é montado no aplicativo e o servidor Modbus é iniciado, basta declará-lo como dispositivo escravo no *OpenPLC*, declarando-o como “dispositivo TCP genérico”. Se todos os dados estiverem corretos, quando o *OpenPLC* for iniciado, já haverá comunicação entre os programas. Os parâmetros de configuração requeridos pelo *OpenPLC* são:

- *Device name*: nome do dispositivo para exibição nas listas e *logs* do *OpenPLC*;
- *Device type*: dentre as opções disponíveis, o *Factory I/O* deve ser declarado como dispositivo Modbus TCP genérico;
- *Slave ID*: número de identificação do dispositivo Modbus;
- *IP Address*: endereço IP do computador que executa o *Factory I/O*;
- *IP Port*: porta de comunicação TCP pela qual trafegará a informação entre o CLP e o dispositivo escravo (deve ser diferente da porta 502, que é por padrão utilizada pelo *OpenPLC*).

Além destes dados, é necessário indicar o número de *bits* de entrada e saída digitais e analógicos utilizados na comunicação, de acordo com o número de atuadores e sensores presentes na planta. Nos modelos desenvolvidos neste trabalho, há apenas variáveis discretas (E/S digital). Dados dos sensores virtuais do *Factory I/O* não são enviados ao *OpenPLC*, uma vez que a função do aplicativo é apenas reproduzir os estados da planta real, ou seja, gerar as peças e mostrar sua movimentação.

A replicação das estações de trabalho *Testing* e *Sorting* representa os aspectos funcionais das mesmas, não sendo possível reproduzir em detalhes os aspectos visuais dos atuadores, uma vez que a inserção de componentes é limitada aos integrantes da biblioteca do *software*, que não pode ser editada. Desta forma, algumas adaptações foram necessárias para representar o sistema.

Quanto às peças de trabalho, o *Factory I/O* disponibiliza apenas objetos quadrados (caixas) nas cores metálico, azul e verde. Assim, para fins de representação das diferentes cores, foi realizada uma correlação entre as peças de trabalho reais e as peças disponíveis no aplicativo, conforme a Tabela 5.

Tabela 5 – Correlação entre as cores das peças reais e as peças disponíveis no aplicativo *Factory I/O*.

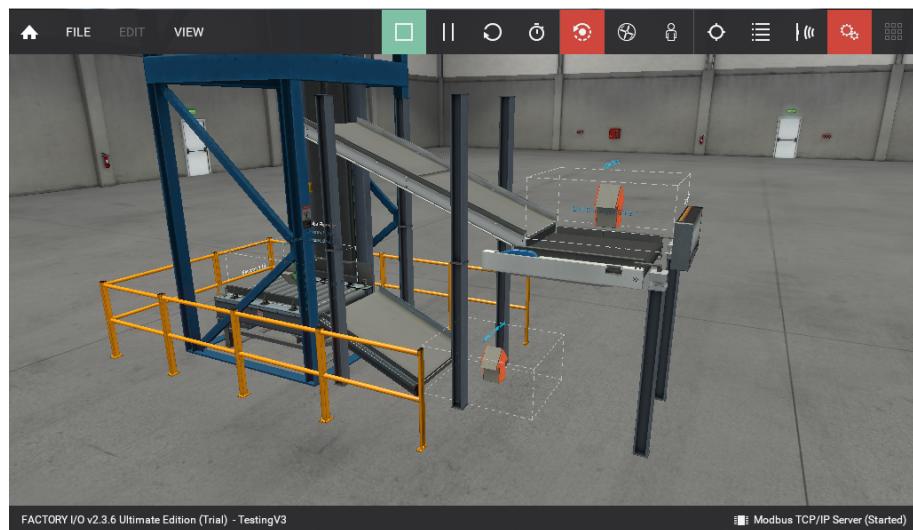
Estação real	Estação virtual
Prata	Prata
Preto	Azul
Vermelho	Verde

Fonte: Autoria própria.

As peças de trabalho são geradas automaticamente no processo virtual quando ocorre a detecção da presença de uma peça de trabalho no processo real. Um *bit* enviado pelo *OpenPLC* é responsável por comandar os geradores de peças, de acordo com a cor da peça identificada na planta. No caso da estação *Testing*, não há aparato de sensores suficiente para diferenciar as cores, sendo assim, o modelo virtual gera sempre um mesmo tipo de peça. Na estação *Sorting* há sensores suficientes para a diferenciação, sendo que as peças são geradas conforme as cores identificadas.

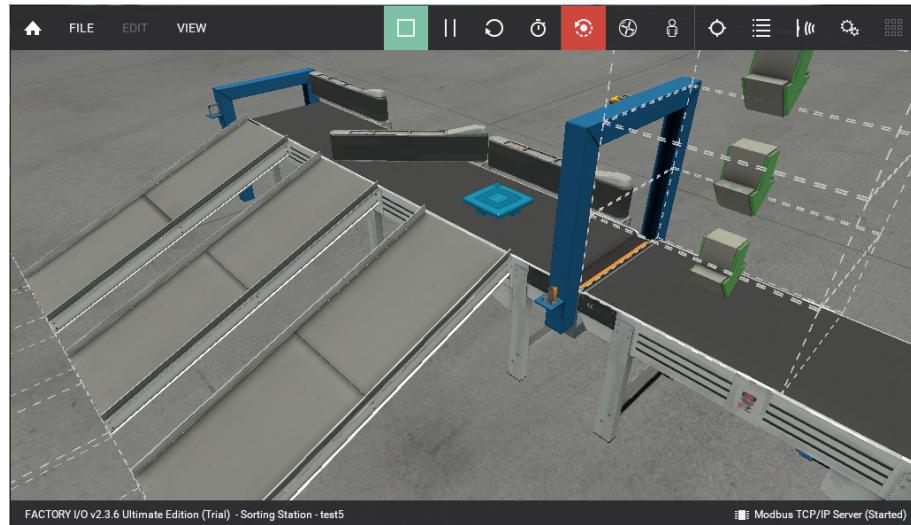
O aplicativo não disponibiliza atuadores pneumáticos, sendo assim, o elevador da estação *Testing* e os desviadores e trava da estação *Sorting* foram substituídos por equivalentes elétricos. Embora o aspecto visual seja diferente, do ponto de vista funcional eles executam a mesma função. As Figuras 36 e 37 apresentam, respectivamente, os modelos virtuais das estações *Testing* e *Sorting*.

Figura 36 – Representação da estação de trabalho *Testing* aplicativo *Factory I/O*.



Fonte: Autoria própria.

Figura 37 – Representação da estação de trabalho *Testing* aplicativo *Factory I/O*.

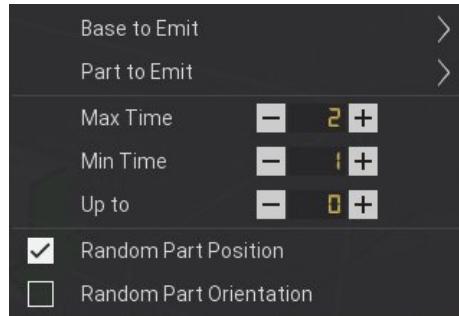


Fonte: Autoria própria.

As configurações do *Factory I/O* não permitem ajuste fino da velocidade das esteiras e movimentos de atuadores, sendo assim, alguns ajustes são necessários para que sejam compensadas as diferenças entre a velocidade dos elementos da planta real e de sua contraparte virtualizada. A primeira ferramenta utilizada é o ajuste global de velocidade de execução, acessado através do ícone de um cronômetro na tela principal. Através deste controle, a velocidade de execução da simulação pode ser ajustada em níveis que variam de 0,5 vezes até 4 vezes a velocidade padrão de movimento dos atuadores.

Além deste ajuste, é necessário ajustar os elementos emissores de peças para que não haja a geração de mais de uma peça em cada ciclo de funcionamento da planta. Isto se faz através do ajuste do parâmetro “*Min Time*”, que regula o tempo mínimo entre a emissão de novas peças. Como este elemento emissor está ligado ao sensor de presença, que permanece ativo por um ou dos segundos de acordo com a estação de trabalho, é necessário correlacionar estes tempos e evitar a geração de múltiplas peças. A página de ajuste é apresentada na Figura 38.

Figura 38 – Parâmetros de ajuste dos elementos emissores no *Factory I/O*.



Fonte: RealGames (2019).

## 4.6 IMPLEMENTAÇÃO DA COMUNICAÇÃO VIA PROTOCOLO UDP

Uma vez que o *OpenPLC* não se comunica por si próprio através do protocolo UDP, foi disponibilizada uma interface que adquire os dados dos *bytes* de E/S e os transmite através de pacotes UDP de 2 *bytes*, na formatação *uint16* (inteiro sem sinal de 16 *bits*). Este mesmo formato é utilizado tanto para a transmissão de variáveis discretas como contínuas (E/S analógica), com valores que podem variar de 0 a 65535.

Os parâmetros declarados na etapa de configuração da interface são:

- Número de estações: número de instâncias do OpenPLC que serão utilizadas (útil em aplicações de controle distribuído);
- IP do *Simulink*: endereço IPv4 do computador que está executando o aplicativo *Simulink*;
- IP da estação *n*: endereço IPv4 do computador que está executando cada instância do *OpenPLC*;
- Endereços de E/S: para cada estação, devem ser adicionados os endereços UDP pelos quais as variáveis de E/S serão disponibilizadas para acesso remoto.

A declaração de endereços de E/S segue a ordem dos *bits* do *OpenPLC* a partir do endereço 0.0 para variáveis discretas e 0 para variáveis contínuas. Assim, por exemplo, o primeiro endereço UDP de saída digital declarado na interface será relacionado ao *bit* %QX0.0, o segundo será relacionado ao *bit* %QX0.1 e assim sucessivamente. Não é possível “saltar” *bits* na declaração.

Uma vez que a interface UDP não permite *broadcast*, é necessário abrir uma instância da interface associada a cada módulo da arquitetura (simulação, visualização

etc.), considerando que cada um destes módulos estará associado a um endereço IP diferente. Sendo assim, cada instância de interface enviará os *bits* para o endereço IP informado no arquivo de configuração.

Para que o *OpenPLC* se comunique com *Simulink* e outros aplicativos via UDP, é necessário declarar as portas de comunicação UDP que serão utilizadas para os *bits* das entradas e saídas de processo que serão comunicadas. Este procedimento é realizado num arquivo de configuração que é lido pelo aplicativo quando de sua execução. Para evitar problemas, deve-se replicar os *bits* de E/S para utilizar portas de comunicação UDP separadas para cada aplicativo de simulação, ainda que seja o mesmo sinal (como por exemplo, do mesmo sensor). No caso mais simples, todos os programas (*OpenPLC*, *Simulink* e *Visual Studio*) são executados no mesmo computador, sendo necessária a execução de uma única instância da interface UDP. Caso a execução dos programas seja distribuída entre vários computadores, basta replicar o código da declaração alterando apenas o endereço IP do computador que executa cada aplicativo e executar uma instância para cada IP. Neste caso, melhores resultados são obtidos quando a interface *SimLink* é executada no mesmo computador que está rodando o *OpenPLC*. As portas declaradas para a estação *Sorting* no caso simples (todos os aplicativos no mesmo computador) são:

```

num_stations = "1"
comm_delay = "10"

# _____
#   APlicativo
# _____
simulink.ip = "127.0.0.1"
#IP - aplicativo (Simulink ou Visual Studio)

# _____
#   ESTACAO (OPENPLC)
# _____
station0.ip = "127.0.0.1"  #IP - OpenPLC

#ESTACAO SORTING - Sensores (para o Simulink)
station0.add(digital_out) = "10000" #presenca
station0.add(digital_out) = "10001" #capacitivo
station0.add(digital_out) = "10002" #indutivo
station0.add(digital_out) = "10003" #fim-de-curso

#ESTACAO SORTING - Sensores (para o Visual Studio)
station0.add(digital_out) = "10004" #presenca
station0.add(digital_out) = "10005" #capacitivo
station0.add(digital_out) = "10006" #indutivo
station0.add(digital_out) = "10007" #fim-de-curso

#ESTACAO SORTING - Atuadores (para o Visual Studio)
station0.add(digital_out) = "10008" #esteira
station0.add(digital_out) = "10009" #desviador 1
station0.add(digital_out) = "10010" #desviador 2

```

```

station0.add(digital_out) = "10011" #trava

#ESTACAO SORTING - Inputs (gerados pelo Simulink)
station0.add(digital_in) = "10100" #esteira
station0.add(digital_in) = "10101" #desv1
station0.add(digital_in) = "10102" #desv2
station0.add(digital_in) = "10103" #trava

#ESTACAO TESTING - Sensores (para o Simulink)
station0.add(digital_out) = "11000" #presenca
station0.add(digital_out) = "11001" #elevado
station0.add(digital_out) = "11002" #abaixado
station0.add(digital_out) = "11003" #recuado

#ESTACAO TESTING - Inputs (gerados pelo Simulink)
station0.add(digital_in) = "11100" #esteira
station0.add(digital_in) = "11103" #trava
station0.add(digital_in) = "11101" #desv1
station0.add(digital_in) = "11102" #desv2

```

A Figura 39 mostra a interface em execução.

Figura 39 – Interface de comunicação UDP em execução.

```

~/simlink
Aluno@Lenovo2 ~
$ cd ~/simlink
Aluno@Lenovo2 ~/simlink
$ ./simlink

Station 0:
ip: 127.0.0.1
DigitalOut 0: 10000
DigitalOut 1: 10001
DigitalOut 2: 10002
DigitalOut 3: 10003
DigitalOut 4: 10004
DigitalOut 5: 10005
DigitalOut 6: 10006
DigitalOut 7: 10007
DigitalOut 8: 10008
DigitalOut 9: 10009
DigitalOut 10: 10010
DigitalOut 11: 10011

```

Fonte: Autoria própria.

O procedimento completo de execução do gêmeo digital é o seguinte:

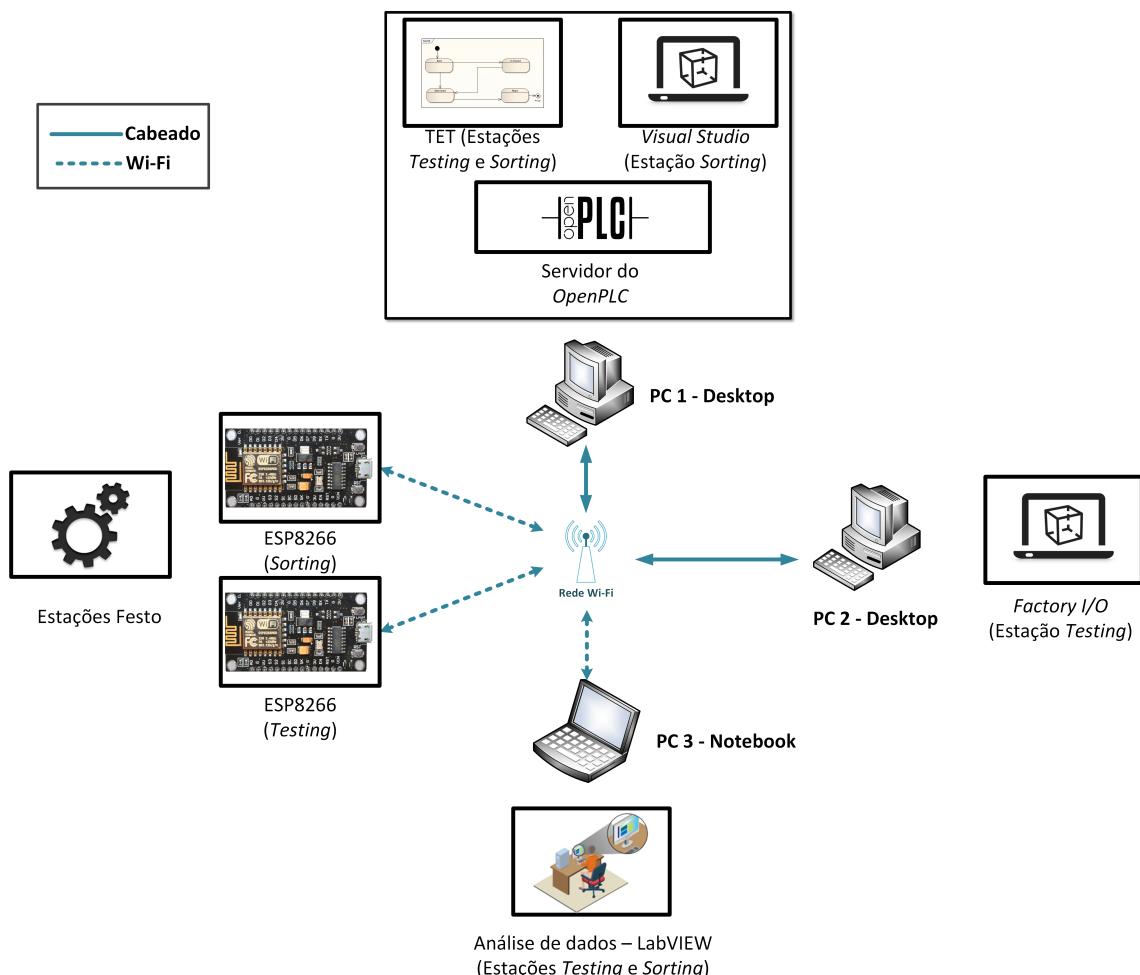
- Inicializar o *OpenPLC*;
- Selecionar o *hardware layer “Simulink”* nas configurações de *hardware* (necessário apenas na primeira execução);
- Carregar o código de programação *Ladder* no ambiente de programação do *OpenPLC*;
- Iniciar o servidor Modbus nas instâncias do aplicativo *Factory I/O* (Modelo 3D - Estação *Testing*).

- Iniciar o CLP (comando *Start PLC*);
- Declarar as portas de entrada e saída no arquivo de configuração da interface de comunicação UDP;
- Iniciar o aplicativo *Simulink* (TET);
- Iniciar a interface de comunicação UDP *SimLink*;
- Iniciar o aplicativo *Visual Studio* (Modelo 3D - Estação *Sorting*).

## 5 RESULTADOS E DISCUSSÃO

Para validar a arquitetura desenvolvida para implementação de gêmeos digitais e avaliar o funcionamento de seus módulos neste trabalho, foram desenvolvidos testes utilizando o arranjo experimental instalado no laboratório. O esquema físico montado para a realização dos experimentos é apresentada na Figura 40. Todos os testes foram realizados numa rede restrita, sem ligação com a Internet e com todas as ferramentas de *firewall* desabilitadas para evitar perdas de velocidade.

Figura 40 – Esquema montado para a realização dos experimentos com a arquitetura.



Fonte: Autoria própria.

Neste cenário temos o *OpenPLC* implementando os gêmeos digitais das duas estações de trabalho (*Testing* e *Sorting*) simultaneamente. Em função de sua construção modular, entretanto, este arranjo não é o único possível, sendo que há possibilidade de executar o gêmeo digital de apenas uma das estações separadamente, ou ainda utilizar

um servidor do *OpenPLC* para gerenciar o gêmeo digital de cada estação. A configuração apresentada foi utilizada com o fim de avaliar a escalabilidade do servidor. Quanto à distribuição dos módulos entre diferentes computadores, optou-se por instalar os módulos que trabalham com a comunicação UDP no mesmo computador, uma vez que a interface *SimLink* comunica os dados do *OpenPLC* para um único endereço IP, permitindo que os vários aplicativos possam utilizar estes dados. Outro dado que deve ser levado em conta é que não é possível executar mais de uma instância do *Factory I/O* ao mesmo tempo na mesma máquina.

## 5.1 ANÁLISE DE DESEMPENHO DO GÊMEO DIGITAL

Foi realizada uma análise de desempenho da comunicação entre os módulos componentes da arquitetura do Gêmeo Digital, motivada sobretudo pela ocorrência dos atrasos de comunicação mencionados anteriormente e para que seja possível avaliar as possibilidades quanto à expansão da plataforma (escalabilidade). O foco desta análise está no tempo de comunicação entre o servidor do *OpenPLC* (Módulo 1) e os demais módulos presentes na arquitetura. A aquisição dos dados foi realizada por meio do software *Wireshark* (OREBAUGH; RAMIREZ; BEALE, 2006), que realiza a leitura de todos os pacotes de informação que trafegam pelo adaptador de rede. O aplicativo foi executado sempre no dispositivo nomeado “PC 1 - Desktop” (vide Figura 40), que é o servidor do *OpenPLC*. Os testes apresentados a seguir foram executados separadamente, de forma a avaliar as características individuais de cada componente. Por fim foi realizado um teste com execução simultânea.

### 5.1.1 DISPOSITIVOS ESP8266

Os SoC ESP8266 são os principais componentes da arquitetura, pois eles são a interface de E/S do *OpenPLC* com as estações de automação reais. Desta forma, falhas que ocorrem nesta comunicação afetam todos os demais módulos, que são alimentados pelos dados da planta real. A comunicação entre o *OpenPLC* e os SoC requer conexão Ethernet sem fio, uma vez que não há entrada para conexão cabeada nos *chips*. Do ponto de vista do *OpenPLC*, cada ESP8266 é um dispositivo escravo (“*slave device*”), sendo que cada um é diferenciado por seu endereço IP.

A comunicação é parametrizada no servidor do *OpenPLC*, através dos parâmetros “*Polling cycle*” e “*Timeout*”, que são válidos para todos os dispositivos escravos declarados. Ela se dá através do protocolo Modbus TCP/IP, no formato “*Query - Response*”, sendo que para cada requisição do servidor (*query*), o escravo responde com os dados solicitados (*response*). As operações realizadas entre o servidor e o escravo são as seguintes:

- *Read Discrete Inputs*: leitura dos quatro *bits* de entrada digital (%IXn.0 - %IXn.3);
- *Write Multiple Coils*: escrita dos quatro *bits* de saída digital (%QXn.0 - %QXn.3);
- *Read Input Registers*: leitura do *byte* de entrada digital (%IXn);
- *Write Multiple Registers*: escrita do *byte* de saída digital (%QXn).

onde “n” é o endereço definido automaticamente pelo *OpenPLC* de acordo com o número de escravos. Para o primeiro escravo, n=100, para o segundo, n=101 e assim sucessivamente. A Figura 41 apresenta um ciclo de comunicação completo entre o servidor e o escravo.

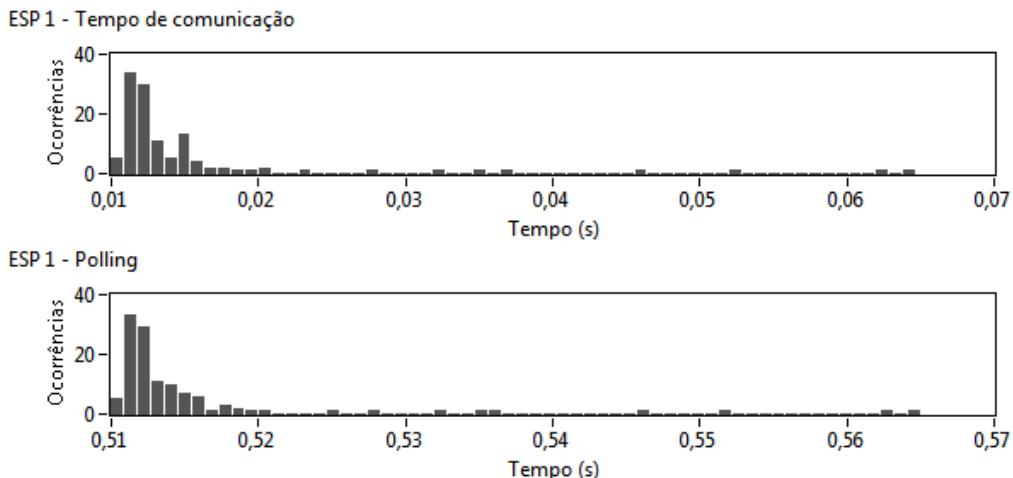
Figura 41 – Ciclo de comunicação entre o *OpenPLC* e o escravo ESP8266.

No.	Time	Source	Destination	Info
16	5.589...	192.168.0.131	192.168.0.125	Query: Trans: 1; Unit: 0, Func: 2: Read Discrete Inputs
17	5.594...	192.168.0.125	192.168.0.131	Response: Trans: 1; Unit: 0, Func: 2: Read Discrete Inputs
18	5.595...	192.168.0.131	192.168.0.125	Query: Trans: 2; Unit: 0, Func: 15: Write Multiple Coils
19	5.598...	192.168.0.125	192.168.0.131	Response: Trans: 2; Unit: 0, Func: 15: Write Multiple Coils
20	5.599...	192.168.0.131	192.168.0.125	Query: Trans: 3; Unit: 0, Func: 4: Read Input Registers
21	5.603...	192.168.0.125	192.168.0.131	Response: Trans: 3; Unit: 0, Func: 4: Read Input Registers
22	5.603...	192.168.0.131	192.168.0.125	Query: Trans: 4; Unit: 0, Func: 16: Write Multiple Registers
23	5.606...	192.168.0.125	192.168.0.131	Response: Trans: 4; Unit: 0, Func: 16: Write Multiple Registers

Fonte: Autoria própria.

O primeiro teste realizado foi realizado com um único dispositivo ESP8266 declarado na lista de escravos do *OpenPLC*. Neste ensaio foi utilizado um ciclo de *polling* de 500ms e *timeout* de 1000ms. O teste teve duração de um minuto. Os resultados são apresentados na forma de histograma, na Figura 42. Nela é mostrado tempo gasto para a realização das oito operações, bem como o intervalo de tempo real de *polling*, ou seja, o tempo entre duas solicitações (*queries*) sucessivas de leitura de entrada digital. Nota-se que o tempo real mensurado é ligeiramente superior (12 a 15ms nos casos mais frequentes) do que o *polling* definido na configuração.

Figura 42 – Tempo gasto para a comunicação e cálculo do ciclo de *polling* para o escravo ESP8266.

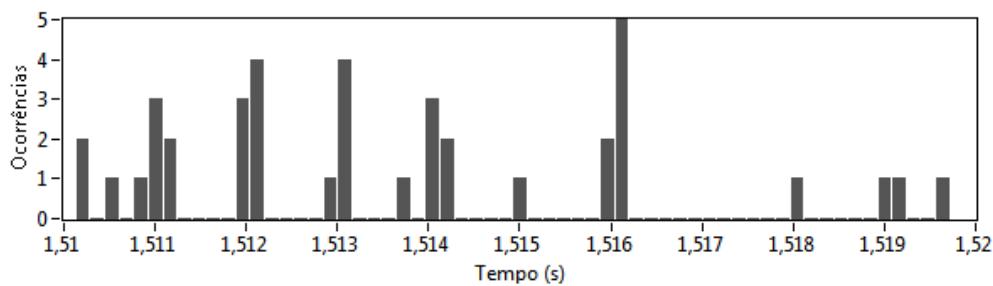


Fonte: Autoria própria.

O parâmetro “*timeout*” diz respeito ao tempo máximo que o servidor do *OpenPLC* aguarda a resposta de um escravo. Caso não haja resposta dentro deste tempo limite, o ciclo de *polling* é reiniciado e uma mensagem de erro é mostrada ao usuário, informando que o escravo não respondeu. Na prática, quando há um escravo desconectado, o *OpenPLC* aguarda o tempo de *polling* mais o tempo de *timeout* para realizar uma nova rodada de atualização dos escravos.

Para ilustrar o exposto, foi realizado um ensaio em condições similares ao anterior (*polling*: 500ms; *timeout*:1000ms) com dois ESP8266 declarados, mas apenas um ligado. A Figura 43 apresenta o histograma do tempo real de *polling* do dispositivo ligado, que neste caso fica em torno de 1500ms.

Figura 43 – Tempo real de *polling* do escravo ESP8266 na presença de um outro escravo declarado mas desligado.

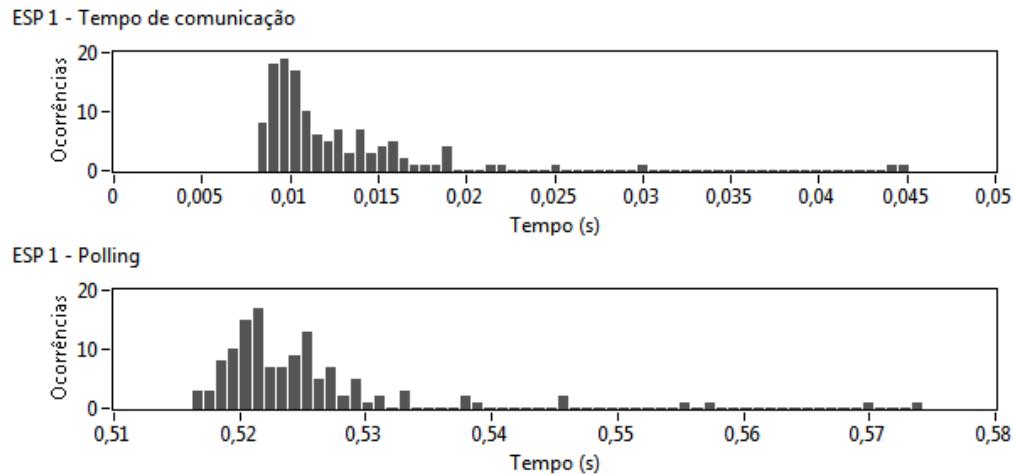


Fonte: Autoria própria.

Neste ensaio fica evidente a necessidade de se ajustar o parâmetro de *timeout* para um valor próximo ao *polling*, de forma que a queda de um escravo cause menor perda ao ciclo de *polling* dos demais escravos.

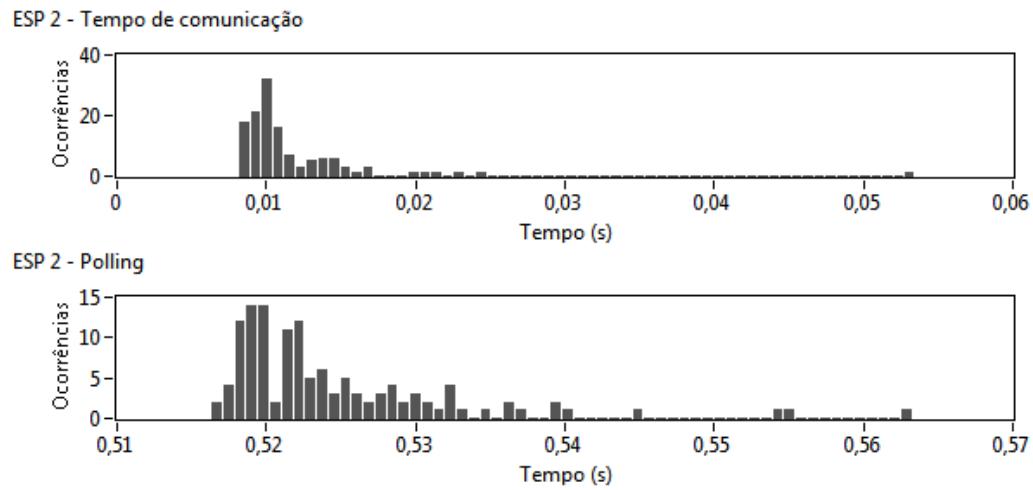
O terceiro ensaio foi realizado com os dois escravos ESP8266 ligados para verificar o desempenho da comunicação nesta condição, novamente nas mesmas condições (*polling*: 500ms; *timeout*:1000ms). Observa-se que não há prejuízo significativo ao ciclo de comunicação. As Figuras 44 e 45 apresentam, respectivamente, os tempos mensurados para o primeiro e o segundo escravo.

Figura 44 – Medições para o primeiro ESP8266 no ensaio com dois escravos em operação.



Fonte: Autoria própria.

Figura 45 – Medições de tempo para o segundo escravo ESP8266 no ensaio com dois escravos em operação simultânea.



Fonte: Autoria própria.

### 5.1.2 FACTORY I/O

O aplicativo *Factory I/O* também funciona como um escravo Modbus TCP/IP do ponto de vista do servidor do *OpenPLC*, estando sujeito às mesmas condições de *polling*

e *timeout* já descritas no caso dos ESP8266. Neste estudo o computador que executa o aplicativo está ligado à rede através de cabo Ethernet. Embora o *Factory I/O* apresente suporte para comunicação de variáveis contínuas (análogicas), estas não foram utilizadas uma vez que o processo é todo discreto. Sendo assim, as operações realizadas pelo *OpenPLC* sobre o escravo são apenas a leitura dos sinais de entrada discretos e a escrita dos sinais de saída discretos. Embora o *Factory I/O* possua sensores virtuais que podem produzir sinais para comunicação com o *OpenPLC* e outros tipos de CLP, neste experimento estes dados são descartados, pois o aplicativo é utilizado apenas para reproduzir a saída do sistema real. A Figura 46 apresenta um ciclo de comunicação completo entre o servidor e o escravo.

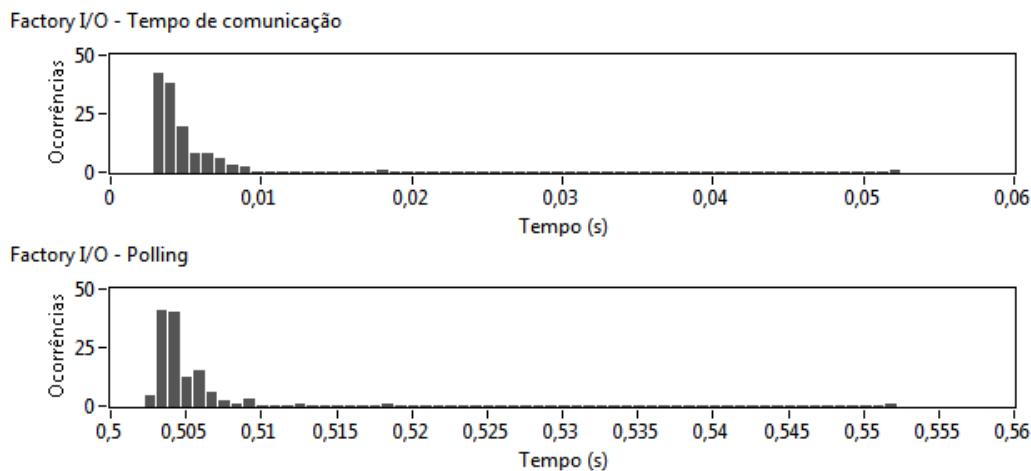
Figura 46 – Ciclo de comunicação entre o *OpenPLC* e o escravo *Factory I/O*.

No.	Time	Source	Destination	Protocol	Info
8	0.509479	192.168.0.131	192.168.0.132	TCP	63722 → 504 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=12
9	0.510611	192.168.0.132	192.168.0.131	TCP	504 → 63722 [PSH, ACK] Seq=1 Ack=13 Win=65536 Len=11
10	0.511453	192.168.0.131	192.168.0.132	TCP	63722 → 504 [PSH, ACK] Seq=13 Ack=12 Win=65536 Len=15
11	0.512587	192.168.0.132	192.168.0.131	TCP	504 → 63722 [PSH, ACK] Seq=12 Ack=28 Win=65536 Len=12

Fonte: Autoria própria.

Um ensaio foi realizado para mensurar o tempo de comunicação, utilizando os mesmos parâmetros de *polling* e *timeout* dos ensaios anteriores. A Figura 47 apresenta os resultados obtidos. Nota-se que o tempo de comunicação é menor em relação aos dispositivos ESP8266, o que pode ser justificado tanto pela conexão de rede cabeadada como pela maior capacidade de processamento do computador que executa o *Factory I/O* em relação aos ESP8266. Por conseguinte, o impacto sobre o tempo de *polling* real também é menor, indicando que dispositivos deste tipo têm um potencial menor de provocar atrasos no contexto geral da temporização da arquitetura quando adicionados.

Figura 47 – Medições de tempo de comunicação e *polling* para o aplicativo *Factory I/O*.



Fonte: Autoria própria.

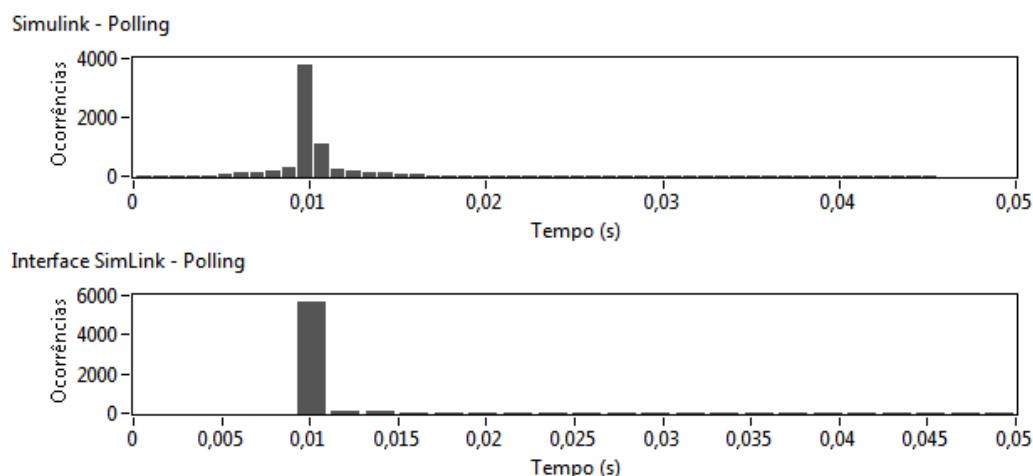
### 5.1.3 COMUNICAÇÃO UDP

A comunicação UDP é utilizada na arquitetura para alimentar a TET no aplicativo *Simulink* e o modelo 3D no *Visual Studio*. Por limitação do aplicativo *Wireshark*, não é possível visualizar a troca de mensagens UDP entre o servidor do *OpenPLC* e os aplicativos quando estes estão rodando na mesma máquina (e por conseguinte operam com o mesmo endereço IP). Desta forma, para possibilitar a análise, o servidor do *OpenPLC* foi rodado num segundo computador, conectado à rede através de cabo Ethernet, para assim possibilitar a leitura das informações trocadas. Neste caso tem-se que:

- Para cada porta UDP de saída declarada (vide lista na Seção 4.6) é enviada uma mensagem;
- Para cada porta UDP de entrada declarada é aguardada uma mensagem vinda do *Simulink* (TET).

Nota-se que, por se tratar de um protocolo leve, a geração de cada mensagem UDP é rápida. Observando os intervalos de mensagem, nota-se que 3 ou até 4 mensagens UDP são geradas dentro de um milissegundo. Um mecanismo implementado na interface *SimLink* permite que haja um intervalo entre o envio das mensagens, que neste caso foi definido em 10 milissegundos. No lado do *Simulink*, ocorre da mesma forma. As mensagens são geradas a cada passo de simulação, conforme o parâmetro de tempo declarado na configuração do *solver*, que foi estabelecido em 10 milissegundos. A Figura 48 apresenta os resultados do tempo de *polling* dos aplicativos de comunicação UDP, destacando o baixo desvio em relação ao tempo estabelecido nas configurações.

Figura 48 – Medições de *polling* dos aplicativos de comunicação UDP *Simulink* e *SimLink*.



Fonte: Autoria própria.

### 5.1.4 LABVIEW

O módulo de análise de dados do processo produtivo se comunica com o servidor do *OpenPLC* através do protocolo Modbus/TCP. Neste caso, a temporização das mensagens independe dos parâmetros de *polling* e *timeout* analisados anteriormente, pois quem faz as solicitações é o próprio *LabVIEW* e não mais o *OpenPLC*, que neste caso apenas emite as respostas às solicitações realizadas. Todo o módulo de análise de dados se baseia num laço de repetição temporizado, cujo tempo de ciclo pode ser ajustado.

Foi realizado um ensaio para observar o tempo de comunicação entre o módulo de aquisição de dados e o servidor do *OpenPLC*. Foi definido um tempo de 1000ms para o ciclo de execução do módulo. O número de mensagens está diretamente relacionado ao número de blocos de leitura dos *bytes* de E/S inseridos no módulo, sendo que neste caso, são utilizados 8 blocos (4 para leitura de entradas e 4 para leitura de saídas, todas discretas). O ciclo completo de comunicação é apresentado na Figura 49.

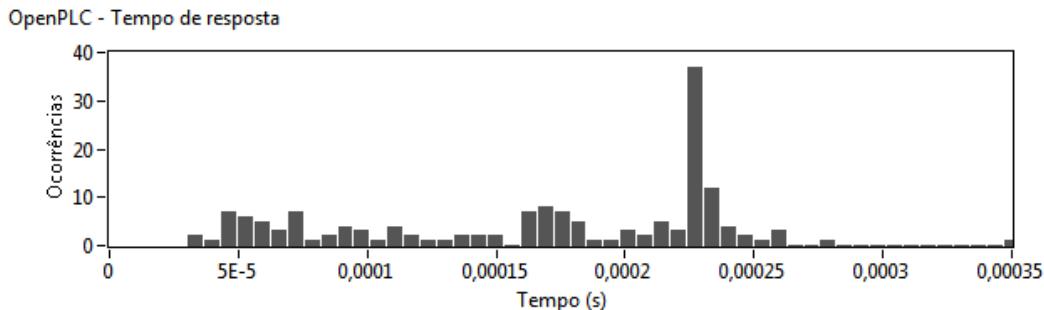
Figura 49 – Ciclo de comunicação entre o aplicativo *LabVIEW* e o banco de dados de E/S do *OpenPLC*.

No.	Time	Source	Destination	Protocol	Info
88	6.5604...	192.168.0.130	192.168.0.131	Mod...	Query: Trans: 0; Unit: 1, Func: 2: Read Discrete Inputs
89	6.5689...	192.168.0.131	192.168.0.130	Mod...	Response: Trans: 0; Unit: 1, Func: 2: Read Discrete Inputs
90	6.5702...	192.168.0.130	192.168.0.131	Mod...	Query: Trans: 0; Unit: 1, Func: 1: Read Coils
91	6.5703...	192.168.0.131	192.168.0.130	Mod...	Response: Trans: 0; Unit: 1, Func: 1: Read Coils
92	6.5718...	192.168.0.130	192.168.0.131	Mod...	Query: Trans: 0; Unit: 1, Func: 2: Read Discrete Inputs
93	6.5719...	192.168.0.131	192.168.0.130	Mod...	Response: Trans: 0; Unit: 1, Func: 2: Read Discrete Inputs
94	6.5733...	192.168.0.130	192.168.0.131	Mod...	Query: Trans: 0; Unit: 1, Func: 1: Read Coils
95	6.5735...	192.168.0.131	192.168.0.130	Mod...	Response: Trans: 0; Unit: 1, Func: 1: Read Coils
96	6.5749...	192.168.0.130	192.168.0.131	Mod...	Query: Trans: 0; Unit: 1, Func: 2: Read Discrete Inputs
97	6.5751...	192.168.0.131	192.168.0.130	Mod...	Response: Trans: 0; Unit: 1, Func: 2: Read Discrete Inputs
98	6.5765...	192.168.0.130	192.168.0.131	Mod...	Query: Trans: 0; Unit: 1, Func: 1: Read Coils
99	6.5767...	192.168.0.131	192.168.0.130	Mod...	Response: Trans: 0; Unit: 1, Func: 1: Read Coils
100	6.5781...	192.168.0.130	192.168.0.131	Mod...	Query: Trans: 0; Unit: 1, Func: 2: Read Discrete Inputs
101	6.5783...	192.168.0.131	192.168.0.130	Mod...	Response: Trans: 0; Unit: 1, Func: 2: Read Discrete Inputs
102	6.5797...	192.168.0.130	192.168.0.131	Mod...	Query: Trans: 0; Unit: 1, Func: 1: Read Coils
103	6.5798...	192.168.0.131	192.168.0.130	Mod...	Response: Trans: 0; Unit: 1, Func: 1: Read Coils

Fonte: Autoria própria.

Uma vez que o tempo de *polling* é gerenciado pelo algoritmo executado no *LabVIEW*, neste teste foi analisada a capacidade de resposta do servidor do *OpenPLC* às requisições Modbus. O histograma resultante é apresentado na Figura 50.

Figura 50 – Medições de tempo de resposta do *OpenPLC* às requisições Modbus realizadas pelo aplicativo *LabVIEW*.



Fonte: Autoria própria.

Pode-se observar que o tempo de resposta é na maior parte das vezes inferior a 0,3ms. Sendo um total de oito requisições no caso deste trabalho, o tempo de atualização médio de todos os dados de E/S utilizados seria de 3,4ms. Considerando que os dispositivos ESP8266 trabalham a uma taxa de atualização bem mais lenta, conclui-se que o desempenho da comunicação *OpenPLC - LabVIEW* é suficiente para a aquisição adequada dos dados do processo.

### 5.1.5 TEMPORIZAÇÃO DO SISTEMA COMPLETO

De posse dos dados de desempenho de comunicação entre os diferentes módulos da arquitetura, foi possível ajustar os parâmetros globais para o funcionamento do gêmeo digital.

O maior limitante observado quanto à velocidade de troca de dados é a comunicação entre o *OpenPLC* e os dispositivos ESP8266. Eles são os principais elementos da arquitetura, pois são as interfaces de E/S que permitem a operação da planta real. São também os que apresentaram desempenho menor nos testes individuais (maior variação do tempo de transmissão), até mesmo por se tratar de SoCs, cujo desempenho é obviamente inferior aos processadores de computadores.

Quanto à comunicação UDP, tem-se grande capacidade de transmissão por conta do formato pequeno dos pacotes. A interface *SimLink* tem capacidade de trocar mensagens UDP em intervalos de milissegundos, mas para que haja desempenho adequado, convém compatibilizar a velocidade de transmissão dos dados com a velocidade dos demais aplicativos que utilizam os dados gerados.

Em relação ao aplicativo *Simulink*, o tamanho do passo de simulação deve ser fixado num valor pequeno o suficiente para corresponder às atualizações recebidas através da interface de comunicação UDP.

O aplicativo de visualização 3D *Factory I/O*, por ser listado como um dispositivo

escravo no *OpenPLC* tal qual os ESP8266, executa sua comunicação na velocidade padronizada de *polling* dos escravos. Quanto à velocidade de operação dos atuadores virtuais, o controle é realizado através do mecanismo de controle de tempo de execução do aplicativo, que não interfere na comunicação Modbus.

Finalmente, o módulo de aquisição e análise de dados possui um controle próprio da frequência de acesso aos dados do *OpenPLC*. Este tempo é ligeiramente superior ao ciclo de *scan*, para evitar processamento desnecessário. A Tabela 6 descreve os principais parâmetros de temporização utilizados na arquitetura.

Tabela 6 – Lista dos parâmetros de temporização dos diversos módulos da arquitetura.

Parâmetro	Aplicativo	Descrição	Valor
Ciclo de <i>scan</i>	<i>OpenPLC</i>	Definido nas configurações do código <i>Ladder</i> gerado	30 ms
<i>Polling</i> Modbus	<i>OpenPLC</i>	Define o tempo de acesso e <i>timeout</i> para todos os escravos do <i>OpenPLC</i>	35 ms
<i>Timeout</i> Modbus	<i>OpenPLC</i>	Tempo máximo de espera pela resposta de um escravo	50 ms
Atraso de comunicação	Interface <i>SimLink</i>	Tempo de geração e recepção das mensagens UDP	30 ms
Passo de simulação	<i>Simulink</i>	Definido nas configurações da instância de simulação	30 ms
Multiplicador de velocidade de simulação (estaçao <i>Testing</i> )	<i>Factory I/O</i>	Definido nas configurações de simulação	2x
Multiplicador de velocidade de simulação (estaçao <i>Sorting</i> )	<i>Factory I/O</i>	Definido nas configurações de simulação	4x
Taxa de aquisição Modbus	<i>LabVIEW</i>	Taxa de atualização dos dados de E/S do <i>OpenPLC</i> acessados via Modbus	30 ms

Fonte: Autoria própria.

No cenário de testes (Figura 40) foi utilizado o modelo virtual implementado no aplicativo *Visual Studio* para representar a estação *Sorting*, enquanto o modelo virtual da estação *Testing* é implementado na ferramenta *Factory I/O*. Há disponibilidade de um modelo no *Factory I/O* também para a estação *Sorting*, entretanto não foi utilizada porque o outro modelo é visualmente mais fidedigno.

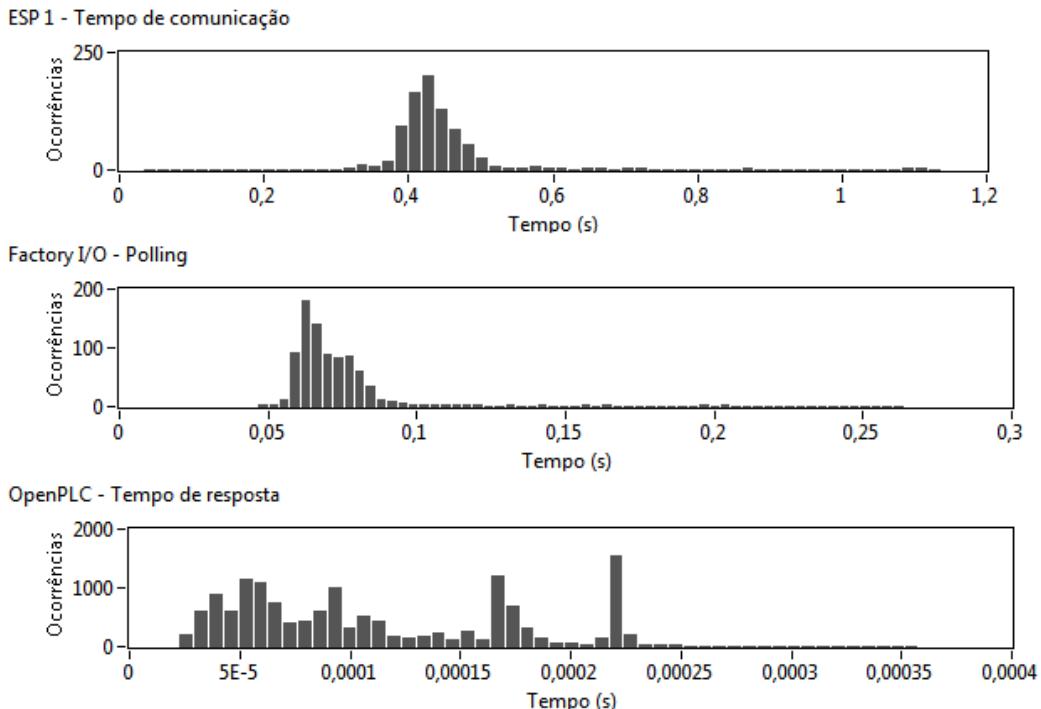
Com base neste *setup*, foi realizado um teste com todos os módulos em execução. Por limitação do aplicativo *Wireshark*, não é possível observar a troca de dados UDP, uma vez que tanto o *OpenPLC* quanto os aplicativos UDP (*Simulink* e *Visual Studio*) são

executados no mesmo computador, logo utilizam o mesmo IP, caso no qual o *Wireshark* não é capaz de realizar a captura. As demais comunicações, porém, puderam ser avaliadas no teste. O *OpenPLC* requisita dados de seus escravos na ordem em que são declarados. Desta forma, o primeiro dispositivo a ser requisitado é o ESP8266 da estação *Sorting*, o segundo é o ESP8266 da estação *Testing* e por fim o aplicativo *Factory I/O*, visto que foram declarados nesta ordem nas configurações do *OpenPLC*. O aplicativo *LabVIEW* envia requisições para o *OpenPLC* conforme a taxa de aquisição listada anteriormente. A interface UDP *SimLink* requisita dados do *OpenPLC* e gera as mensagens UDP de acordo com o tempo (atraso de comunicação) definido em suas configurações.

Constatou-se que os módulos da arquitetura em operação simultânea por vezes produzem atrasos e prejudicam o desempenho global do sistema, especialmente os dispositivos ESP8266. Essas variações de desempenho são observáveis sobretudo nos *logs* do servidor do *OpenPLC*, onde são apresentadas mensagens quando ocorre estouro de *timeouts* por atraso de comunicação, entretanto a adoção de um tempo de *timeout* mais próximo ao tempo de *polling* faz com que a operação não se torne inviável e minimiza as distorções provocadas nos processos de temporização do módulo de análise de dados. De acordo com o mecanismo de *polling* dos escravos, quando um deles atrasa sua resposta, a requisição do próximo escravo é enviada tarde, assim impactando o *polling* de todos os demais.

A Figura 51 ilustra este efeito, apresentando 3 histogramas: no primeiro, é mostrado o tempo de comunicação do ESP1, que na maior parte fica acima do tempo normal de *polling*, mas abaixo do tempo de *timeout*, forçando o tempo de *polling* dos demais para cima. O tempo de comunicação do ESP2 não é apresentado por apresentar perfil similar ao do ESP1. No segundo é apresentado o tempo de *polling* do *Factory I/O*, cujos tempos de comunicação foram todos inferiores ao *polling* definido (sempre inferiores a 10ms), mas que sofre o impacto dos atrasos dos ESP8266. No terceiro é mostrado o tempo de resposta do *OpenPLC* às requisições do módulo de análise de dados (*LabVIEW*), mostrando que os atrasos no *polling* dos escravos não reduziram o desempenho do *OpenPLC* quanto à capacidade de retornar dados para o módulo de análise.

Figura 51 – Análise de desempenho da comunicação dos módulos no teste completo.



Fonte: Autoria própria.

## 5.2 ANÁLISE DE DADOS DO PROCESSO

Uma vez observado o correto funcionamento das estações de automação reais e correta replicação virtual em ambiente 3D, as demais análises foram realizadas por meio do módulo de análise de dados. Os primeiros levantamentos realizados dizem respeito aos tempos de processo. Foram realizadas medições dos KPIs do processo indicados anteriormente (vide Tabela 4). Para obter um tempo médio de cada KPI foi realizada uma série inicial de 25 ensaios, observando a manutenção das condições de suprimento elétrico e pressão de ar. Os resultados estão expressos na Tabela 7.

Destes dados depreende-se que a maior dispersão de resultados ocorre na plataforma pneumática da estação *Testing*. Este fato pode ser visivelmente constatado ao longo da operação e deve-se a fatores construtivos da estrutura, uma vez que a pressão de trabalho foi mantida ao longo de todo o experimento. Observa-se baixo desvio padrão nos KPIs da estação *Sorting* por conta da velocidade constante da esteira.

Os valores médios obtidos foram utilizados para refinar os modelos lógicos (TETs) no aplicativo *Simulink*, uma vez que os tempos de transição foram inicialmente determinados por medição manual. De posse destes parâmetros refinados, foi possível implementar a medição comparativa entre os tempos de acionamento dos atuadores reais (estação de automação) e virtuais (dados gerados pelas TETs). Notou-se ao longo dos experimentos que por vezes o processo real é mais rápido, em outras vezes mais lento que o modelo virtual,

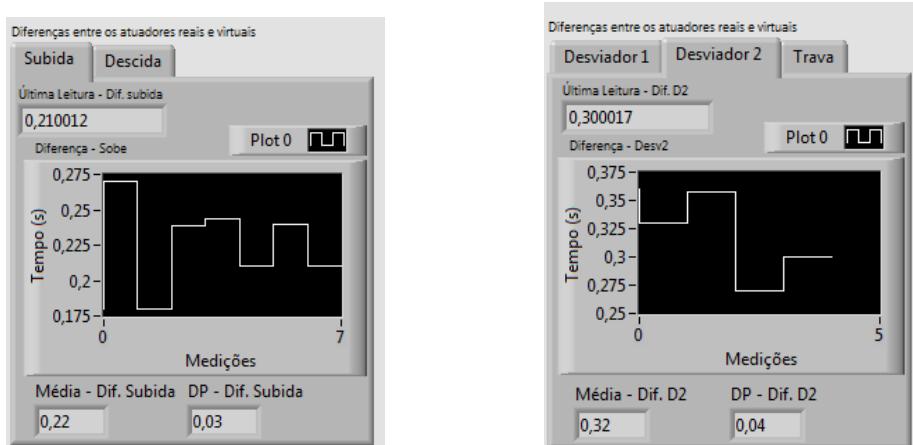
Tabela 7 – KPIs da planta lidos através do módulo de análise de dados do processo.

Estação	KPI	Média (s)	Desvio padrão (s)
Testing	Tempo de subida (plataforma)	0,46	0,10
Testing	Tempo de descida (plataforma)	1,04	0,07
Sorting	Tempo de processo (peça prata)	1,32	0,06
Sorting	Tempo de processo (peça preta)	2,17	0,06
Sorting	Tempo de processo (peça vermelha)	2,50	0,06

Fonte: Autoria própria.

o que já era esperado em decorrência das oscilações de desempenho que são naturais dos equipamentos de campo. A Figura 52 mostra alguns comparativos.

Figura 52 – Comparativo entre os tempos de acionamento dos atuadores reais e do modelo virtual.

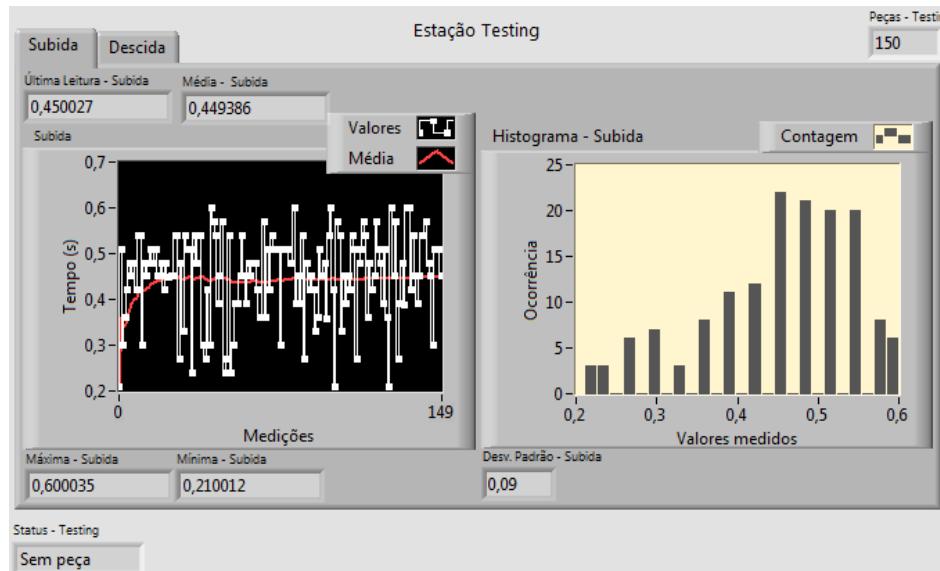


Fonte: Autoria própria.

### 5.2.1 LEVANTAMENTOS ESTATÍSTICOS

Foi realizado um levantamento estatístico para se analisar o tipo de dispersão das medidas de desempenho das estações *Testing* e *Sorting*. Na estação *Testing*, o principal atuador é a plataforma pneumática de elevação vertical das peças. Ele é a principal fonte de variações de desempenho, sendo que seu comportamento e velocidade são visivelmente variáveis, especialmente no processo da subida. Mesmo em condições de pressão constante, são observadas variações de desempenho. Foi realizado um ensaio com 150 repetições na estação *Testing* para levantar a dispersão dos resultados. A Figura 53 apresenta o resultado para o processo de subida.

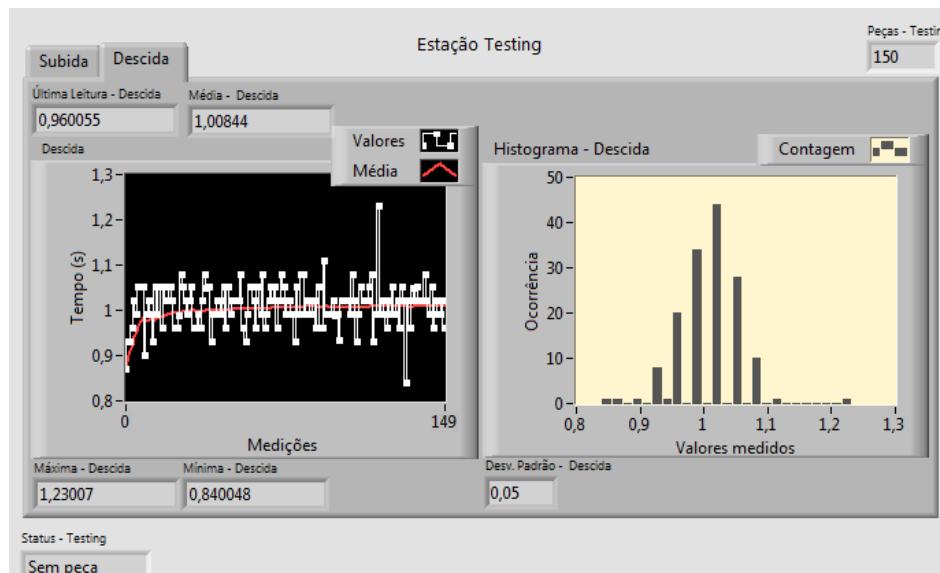
Figura 53 – Dispersão dos tempos de subida ao longo de 150 ensaios na estação *Testing*.



Fonte: Autoria própria.

Nota-se que o processo de subida é particularmente difuso, sendo que seu desvio padrão é 80% maior em relação ao desvio padrão do processo de descida. Analisando-se o histograma, percebe-se que não se trata de uma distribuição normal, apresentando também comportamento assimétrico. A Figura 54 apresenta o resultado para o processo de descida da plataforma pneumática.

Figura 54 – Dispersão dos tempos de descida ao longo de 150 ensaios na estação *Testing*.

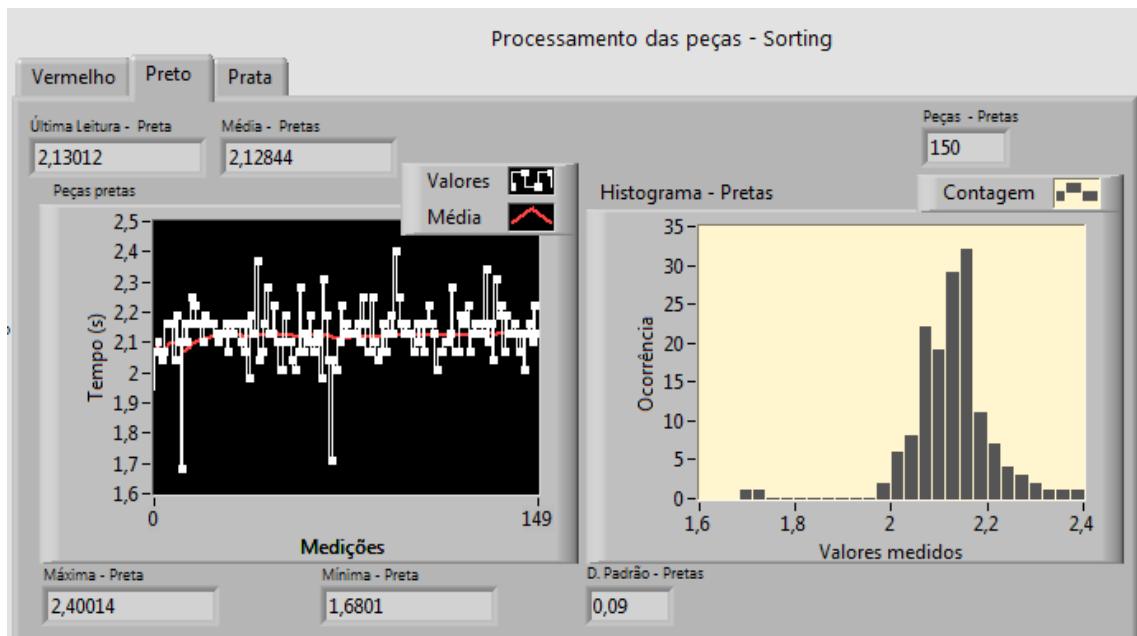


Fonte: Autoria própria.

Neste caso, nota-se uma distribuição mais próxima à distribuição normal e menor desvio padrão, apontando para um comportamento mais homogêneo do atuador.

Da mesma forma, foi analisado o comportamento da estação *Sorting*. Nesta, o atuador central é a esteira elétrica, que opera a velocidade constante e fica constantemente ligada, anulando assim impactos de aceleração ou desaceleração do motor. O principal fator gerador de variações neste processo é a transição entre a esteira e a rampa de saída das peças, onde por vezes as peças demoram mais ou menos tempo para cair na rampa. O formato curvo do desviador também faz com que as peças avancem mais rápida ou lentamente de acordo com o ponto do atuador onde ocorre o primeiro contato. A Figura 55 apresenta o levantamento estatístico obtido num ensaio com 150 peças da cor preta. As peças das demais cores seguem o mesmo padrão estatístico.

Figura 55 – Dispersão dos tempos de processo de peças pretas ao longo de 150 ensaios na estação *Sorting*.



Fonte: Autoria própria.

Nota-se que não se trata de uma distribuição normal, apresentando também característica assimétrica, uma vez que o valor de maior ocorrência não coincide com o valor da média. Para obter uma medida mais precisa, os dados do histograma foram exportados para um aplicativo de tratamento de dados e o histograma foi refeito, resultando no exposto na Figura 56. O valor de maior incidência é 2,15 segundos enquanto a média é de 2,12 segundos.

Figura 56 – Dispersão dos tempos de processo de peças pretas ao longo de 150 ensaios na estação *Sorting*.



Fonte: Autoria própria.

### 5.2.2 CRIAÇÃO DE ALARMES

Com base na avaliação do tipo de distribuição é possível projetar os valores-limite de operação das etapas de trabalho nas estações. Para fins de geração de alarmes, são classificados três níveis de operação: o nível normal, que corresponde aos valores mais próximos à média; o nível de aviso, no qual há atrasos que representam desvios menores em relação ao padrão normal de operação e, por fim, o nível de erro, no qual há atrasos de grande magnitude.

Ao longo dos testes observou-se ainda que nem todos os resultados anômalos foram resultantes de questões físico-mecânicas das estações. Algumas delas decorreram de atrasos de comunicação entre os dispositivos ESP8266 (interfaces de E/S) e o servidor do *OpenPLC*. Os atrasos por vezes são perceptíveis, especialmente através do aplicativo de análise de dados do processo. Neste aplicativo foi criado um painel com LEDs que indicam o estado de sensores e atuadores de acordo com os dados provenientes dos ESP8266. Em alguns momentos, nota-se que o estado de um dado sensor ou atuador demora a ser atualizado, provocando assim inconsistência no cálculo do tempo de trabalho.

De forma a minimizar o efeito das variações de desempenho na geração dos alarmes, a lógica de disparo foi baseada em dados atualizados iterativamente dos processos de produção. Cada tempo mensurado é armazenado num vetor, sobre o qual são realizados os cálculos de média, desvio padrão, valor máximo, valor mínimo e a geração do histograma. Assim, os parâmetros de alarme são definidos em função da média e do desvio padrão, que são atualizados constantemente, de forma a oferecer maior adaptatividade do sistema às variações, especialmente as de caráter permanente (tais como reduções de *performance* por desgaste, perda ou excesso de pressão por longos períodos de tempo etc.), que são capazes de provocar a alteração do valor médio ao longo do funcionamento das estações.

Em virtude dos diferentes tipos de distribuição estatística, não é possível utilizar uma mesma métrica de classificação de nível para todas as estações de trabalho. Sendo assim, para cada estação foram definidos parâmetros de classificação em função da média e do desvio padrão do processo analisado (subida e descida na estação *Testing*; processamento das diferentes cores na estação *Sorting*). A Tabela 8 apresenta os parâmetros de cálculo. As siglas utilizadas na tabela são:

- To - Tempo de operação: tempo decorrido desde o início do processamento da peça;
- M - Média: média dos tempos de processamento anteriores;
- DP - Desvio-padrão: desvio padrão dos tempos de processamento anteriores.

Tabela 8 – Definições das métricas de geração de avisos e erros implementada no aplicativo de análise de dados do processo produtivo.

Estação	Operação	Nível de aviso	Nível de erro
<i>Testing</i>	Subida da plataforma	To > M + 1,5*DP	To > M + 4*DP
<i>Testing</i>	Descida da plataforma	To > M + 1*DP	To > M + 4*DP
<i>Sorting</i>	Seleção das peças por cor	To > M + 2*DP	To > M + 5*DP

Fonte: Autoria própria.

No painel principal do aplicativo de análise de dados, foram inseridas duas formas de exibição da incidência de avisos e erros. A primeira e mais simples consiste em indicação luminosa amarela para avisos e vermelha para erros. Complementarmente, foi criado um mecanismo de registro (*log*) que armazena textualmente o tipo de evento (aviso ou erro) e o momento de incidência (hora). A Figura 57 apresenta uma tela do aplicativo.

Figura 57 – Tela de exibição de avisos e erros.



Fonte: Autoria própria.

### 5.2.3 TESTE COM INJEÇÃO DE FALHAS

Para mostrar a dinâmica do mecanismo de alarmes, foi realizado um experimento que ilustra alguns elementos da estratégia adotada. Foram inseridas na planta de automação 25 peças da mesma cor, no caso, a cor preta. As primeiras 15 peças foram inseridas normalmente e não houve interferência no processo, de forma a produzir dados de média e desvio padrão. A peça seguinte sofreu um leve toque durante seu deslocamento ao longo da esteira na estação *Sorting*, com o intuito de atrasar o trajeto. Em seguida foram colocadas mais sete peças sobre as quais não houve interferência. Logo após, uma peça foi inserida e sofreu um forte toque durante o deslocamento, a fim de gerar um atraso maior em relação à primeira interferência. Por fim, a última peça foi bloqueada manualmente por cinco segundos na esteira. Os resultados obtidos são apresentados na Tabela 9.

Tabela 9 – Resultados do teste com injeção de falhas.

Descrição do evento	Estação	Provocado?
Aviso de subida (peça 10)	<i>Testing</i>	Não
Aviso e erro (peça 16)	<i>Sorting</i>	Sim (leve toque)
Aviso (peça 24)	<i>Sorting</i>	Sim (forte toque)
Aviso e erro (peça 25)	<i>Sorting</i>	Sim (bloqueio)

Fonte: Autoria própria.

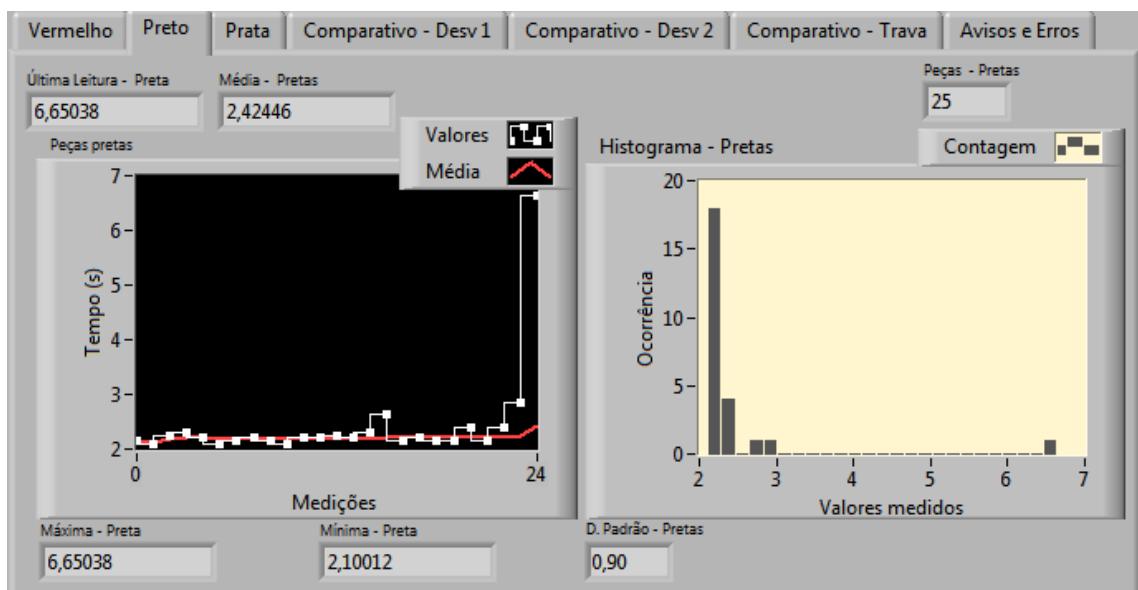
O primeiro evento gerado ao longo deste teste não foi planejado e reflete as características da plataforma pneumática da estação *Testing*, cujo comportamento é particularmente difuso. Embora nenhuma perda de desempenho visível tenha sido observada neste teste, a medição de tempo mostra uma execução mais lenta que o definido como normal na relação entre média e desvio padrão.

Em seguida foi forçado o primeiro distúrbio no processo. Nota-se que a intenção original era gerar apenas um evento de aviso, dado que o leve toque não provocou grande deslocamento ou atraso da peça, entretanto, foi suficiente para gerar também um evento de erro. Isto se dá porque o desvio padrão dos processos da estação *Sorting* são baixos, sendo assim, a faixa de tolerância em termos de desvio padrão tende a ser mais restrita.

A interferência seguinte foi executada com maior intensidade. Observa-se que mesmo assim, o evento gerado foi apenas um aviso. Uma vez que já houve uma interferência anterior, a média consequentemente já estava maior, dado que houve uma contribuição do evento de aviso anterior, bem como o desvio padrão da amostra.

Finalmente o bloqueio completo da peça para finalizar o teste, na qual é constatada a geração dos eventos de aviso, inicialmente, e de erro logo após. Após um erro com atraso drástico como este, o funcionamento do mecanismo de geração de eventos fica prejudicado, uma vez que a média se eleva consideravelmente, bem como o desvio-padrão. Sendo assim, o sistema perde boa parte da “sensibilidade” aos erros que foi notada na primeira interferência. A Figura 58 apresenta o resultado final do teste, ilustrando o impacto das intervenções (da última, em particular), sobre a média e o desvio padrão.

Figura 58 – Resultados do teste com injeção de falhas.



Fonte: Autoria própria.

Para fins de comparação, pode-se retomar os resultados obtidos no teste com 150

peças pretas (Figura 55). Naquele teste, a média obtida foi de 2,13 segundos com desvio padrão de 0,09 segundos. Neste, a média obtida foi de 2,42 segundos com desvio padrão de 0,90 segundos. Num experimento com mais amostras, o impacto de uma amostra anômala sobre as demais certamente seria menor, entretanto não pode ser anulado.

## 6 CONCLUSÕES

Este trabalho apresentou o desenvolvimento de uma arquitetura para construção de gêmeos digitais com foco nos conceitos e ferramentas da Indústria 4.0. Foi desenvolvida uma arquitetura de implementação, centrada na ferramenta de controle de processos *OpenPLC*, que permite o uso de diversos tipos de *hardware* como interfaces de E/S e ainda dispõe da possibilidade de comunicação de dados via redes *Ethernet* utilizando os protocolos TCP e UDP sobre IP. Os principais focos foram o uso de ferramentas *open source* e a manutenção da modularidade da plataforma, de modo a permitir a expansão, inclusão ou remoção de módulos de forma a impactar o mínimo possível na operação dos demais.

Nesta arquitetura foram implementadas técnicas relacionadas ao contexto da Indústria 4.0. O conjunto resultante da integração entre o processo real, o módulo de visualização 3D e o aplicativo de simulação rodando paralelamente em tempo de execução forma o gêmeo digital do sistema de automação, uma vez que o ambiente virtual converge com o real, produzindo dados fidedignos para a validação em tempo de execução. Os testes e ensaios foram implementados em estações modulares de automação.

O sistema foi modelado através das Tabelas de Estados-Transições, implementadas no aplicativo *Simulink* da família *Matlab*. Esta modelagem é baseada no formato de máquinas de estados e é capaz de enviar e receber dados em tempo de execução. Os sinais dos sensores reais são repassados para o aplicativo, que os processa, implementa a temporização e produz dados de saída, que podem ser comparados com os dados de saída do processo real. Ao longo do trabalho foram discutidos os modelos lógicos das estações de trabalho utilizadas (*Testing* e *Sorting*) e a comunicação do *Simulink* com o *OpenPLC*.

A visualização 3D é um recurso que permite o monitoramento remoto do processo, uma vez que todo o aparato está implementado sobre redes *Ethernet*. Foram apresentadas duas alternativas para este fim. A primeira é baseada em modelagem CAD, que produz uma representação bastante fidedigna, porém possui maior complexidade quanto à sua criação. A segunda é baseada no aplicativo *Factory I/O*, que possui bibliotecas de dispositivos industriais prontas, bastando ao usuário criar o seu processo de produção. A complexidade da criação do modelo é muito menor, entretanto a aparência visual fica prejudicada, pois não é possível personalizar componentes.

O módulo de análise de dados foi implementado no aplicativo *LabVIEW* e obtém dados do *OpenPLC* através de comunicação Modbus TCP/IP. Ele oferece uma visão pormenorizada do desempenho do sistema, produzindo dados de histórico, média, dispersão e gerando alertas quanto à incidência de condições de aviso ou erro. Foram criadas diversas

telas apresentando dados de leitura, comparativos do sistema real com o virtual e *logs* de alertas.

Foi realizada também uma análise de desempenho da comunicação entre os módulos, mostrando alguns gargalos de desempenho e as condições de temporização necessárias para o funcionamento do conjunto. O ciclo de comunicação do *OpenPLC* com seus dispositivos escravos deve ser observado, de forma que a incidência de um erro ou atraso por parte de algum escravo não inviabilize o funcionamento dos demais. Avaliar o *polling* real de cada escravo é importante para garantir que o *OpenPLC* receba dados com frequência suficiente para o controle adequado do processo. Foram discutidos os aspectos mais importantes quanto à definição dos parâmetros de temporização e os impactos da adição de escravos ao servidor do *OpenPLC*.

Este trabalho demonstra a flexibilidade da arquitetura, sobretudo a possibilidade de integrar diferentes tipos de aplicativos a partir dos protocolos de comunicação disponíveis. Outras ferramentas de modelagem, visualização 3D, análise de dados e até mesmo outros tipos de *hardware* de E/S podem ser utilizados dentro da arquitetura desenvolvida. Outros tipos de módulos podem ser adicionados conforme a necessidade, bem como é possível utilizar apenas parte dos módulos integrantes quando conveniente, sem inviabilizar o conjunto como um todo.

A arquitetura desenvolvida proporciona uma contribuição significativa quanto à generalidade do protocolo de comunicação implementado e ao uso de ferramentas *open source*, em contraposição a diversos trabalhos encontrados na literatura que se utilizam de soluções de automação industrial proprietárias disponibilizadas a alguns pesquisadores por meio de acordos entre empresas e universidades, mas não disponíveis para a comunidade acadêmica em geral.

## 6.1 TRABALHOS FUTUROS

Como possíveis trabalhos futuros, pode-se citar:

1. Desenvolver um método formal de modelagem de SEDs com capacidade de comunicação em rede para ser utilizado no lugar das TETs;
2. Estudo sobre as perdas de desempenho acarretadas pela expansão do Gêmeo Digital (aumento no número de módulos integrantes);
3. Inserção de ferramentas de aprendizagem de máquina no contexto da análise do processo produtivo.

## 7 Referências Bibliográficas

- ALVES, T. R. et al. Openplc: An open source alternative to automation. In: IEEE. *Global Humanitarian Technology Conference (GHTC), 2014 IEEE*. [S.l.], 2014. p. 585–589.
- AYDOS, T. F. et al. Sistema de monitoramento da manufatura baseado em rfid no âmbito da internet of things. 2016.
- BASSI, L. Industry 4.0: Hope, hype or revolution? In: IEEE. *2017 IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI)*. [S.l.], 2017. p. 1–6.
- BODEMANN, C. D.; ROSE, F. D. The successful development process with matlab simulink in the framework of esa's atv project. In: *55 th International Astronautical Congress*. [S.l.: s.n.], 2004.
- CARDOSO, J.; VALETTE, R. *Redes de petri*. [S.l.]: Editora da UFSC, 1997.
- CARDOSO, L. das D.; RANGEL, J. J. de A.; BASTOS, P. J. T. Discrete event simulation for integrated design in the production and commissioning of manufacturing systems. In: IEEE. *Simulation Conference (WSC), 2013 Winter*. [S.l.], 2013. p. 2544–2552.
- CASSANDRAS, C. G.; LAFORTUNE, S. *Introduction to discrete event systems*. [S.l.]: Springer Science & Business Media, 2009.
- CHAVAN, S. R. Augmented reality vs. virtual reality: differences and similarities. *Int. J. Adv. Res. Comput. Eng. Technol*, v. 5, p. 1–6, 2014.
- CIRIACO, D. *Ford já usa realidade aumentada da Microsoft para modelar carros*. 2017. Disponível em: <<https://www.tecmundo.com.br/produto/122255-ford-realidade-aumentada-microsoft.htm>>. Acesso em: 01 set. 2018.
- COBURN, J. Q.; FREEMAN, I.; SALMON, J. L. A review of the capabilities of current low-cost virtual reality technology and its potential to enhance the design process. *Journal of Computing and Information Science in Engineering*, American Society of Mechanical Engineers, v. 17, n. 3, p. 031013, 2017.
- FRANCHI, C.; CAMARGO, V. Controladores lógicos programáveis—1<sup>a</sup> edição. *São Paulo*: Érica, 2008.
- GLAESSEN, E.; STARGEL, D. The digital twin paradigm for future nasa and us air force vehicles. In: *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 20th AIAA/ASME/AHS Adaptive Structures Conference 14th AIAA*. [S.l.: s.n.], 2012. p. 1818.
- GRIEVES, M. Digital twin: manufacturing excellence through virtual factory replication. *White paper*, 2014.
- HARRISON, W. S. *Virtual Fusion: The Integration and Analysis of Simulation and Real Processes for Manufacturing Process Deployment*. Tese (Doutorado) — University of Michigan, 2011.

- JAZDI, N. Cyber physical systems in the context of industry 4.0. In: IEEE. *Automation, Quality and Testing, Robotics, 2014 IEEE International Conference on*. [S.l.], 2014. p. 1–4.
- JERALD, J. *The VR book: Human-centered design for virtual reality*. [S.l.]: Morgan & Claypool, 2015.
- JOHANSSON, O. *Testing and evaluation of virtual commissioning*. [S.l.]: Göteborg, 2017.
- KO, M.; AHN, E.; PARK, S. C. A concurrent design methodology of a production system for virtual commissioning. *Concurrent Engineering*, Sage Publications Sage UK: London, England, v. 21, n. 2, p. 129–140, 2013.
- KOCH, V. et al. Industry 4.0: Opportunities and challenges of the industrial internet. *Strategy & PwC*, 2014.
- LEE, C. G.; PARK, S. C. Survey on the virtual commissioning of manufacturing systems. *Journal of Computational Design and Engineering*, Elsevier, v. 1, n. 3, p. 213–222, 2014.
- LEE, J.; BAGHERI, B.; KAO, H.-A. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, Elsevier, v. 3, p. 18–23, 2015.
- LEE, J. et al. Recent advances and trends in predictive manufacturing systems in big data environment. *Manufacturing Letters*, Elsevier, v. 1, n. 1, p. 38–41, 2013.
- LIU, Z.; SUCHOLD, N.; DIEDRICH, C. Virtual commissioning of automated systems. In: *Automation*. [S.l.]: InTech, 2012.
- MAKRIS, S.; MICHALOS, G.; CHRYSSOLOURIS, G. Virtual commissioning of an assembly cell with cooperating robots. *Advances in Decision Sciences*, Hindawi, v. 2012, 2012.
- MALÝ, I.; SEDLÁČEK, D.; LEITAO, P. Augmented reality experiments with industrial robot in industry 4.0 environment. In: IEEE. *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*. [S.l.], 2016. p. 176–181.
- MATHWORKS. *Simulink - Simulation and Model-Based Design*. [S.l.], 2018. Disponível em: <<https://www.mathworks.com/products/simulink.html>>. Acesso em 12 jun. 2018.
- NEGAHBAN, A.; SMITH, J. S. Simulation for manufacturing system design and operation: Literature review and analysis. *Journal of Manufacturing Systems*, Elsevier, v. 33, n. 2, p. 241–261, 2014.
- NEGRI, E.; FUMAGALLI, L.; MACCHI, M. A review of the roles of digital twin in cps-based production systems. *Procedia Manufacturing*, Elsevier, v. 11, p. 939–948, 2017.
- OLIVEIRA, L. E. S. d. Concepção de um framework para monitoramento e teleoperação de máquinas-ferramenta cnc via internet aderente à indústria 4.0. 2017.
- OREBAUGH, A.; RAMIREZ, G.; BEALE, J. *Wireshark & Ethereal network protocol analyzer toolkit*. [S.l.]: Elsevier, 2006.
- PAELKE, V. Augmented reality in the smart factory: Supporting workers in an industry 4.0. environment. In: IEEE. *Emerging Technology and Factory Automation (ETFA), 2014 IEEE*. [S.l.], 2014. p. 1–4.

- PALATTELLA, M. R. et al. Internet of things in the 5g era: Enablers, architecture, and business models. *IEEE Journal on Selected Areas in Communications*, IEEE, v. 34, n. 3, p. 510–527, 2016.
- PESSOA, M. A. et al. Industry 4.0, how to integrate legacy devices: A cloud iot approach. In: IEEE. *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*. [S.l.], 2018. p. 2902–2907.
- PISCHING, M. et al. Arquitetura para desenvolvimento de sistemas ciber-físicos aplicados na indústria 4.0. In: *SIMPÓSIO BRASILEIRO DE AUTOMAÇÃO INTELIGENTE*. [S.l.: s.n.], 2017. v. 8.
- REALGAMES. *Factory I/O (Documentation)*. 2019. Disponível em: <<https://factoryio.com/docs/>>. Acesso em: 07 jun. 2019.
- REINHART, G.; WÜNSCH, G. Economic application of virtual commissioning to mechatronic production systems. *Production engineering*, Springer, v. 1, n. 4, p. 371–379, 2007.
- RODRIGUES, J. O que é o esp8266? - a família esp e o nodemcu. Disponível em :<<https://portal.vidadesilicio.com.br/o-que-esp8266-nodemcu/>>, 2018. Acesso em 11 abr. 2018.
- SAEZ, M. et al. Real-time manufacturing machine and system performance monitoring using internet of things. *IEEE Transactions on Automation Science and Engineering*, IEEE, 2018.
- SANTOS, G. J.; CARVALHO, H. de. *Projeto e desenvolvimento de interface tridimensional para sistema supervisório e de treinamento*. Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo, 2016.
- SISINNI, E. et al. Industrial internet of things: Challenges, opportunities, and directions. *IEEE Transactions on Industrial Informatics*, IEEE, 2018.
- SON, Y. J. et al. Simulation-based shop floor control. *Journal of Manufacturing Systems*, v. 21, n. 5, p. 380–394, 2002.
- STEUER, J. Defining virtual reality: Dimensions determining telepresence. *Journal of communication*, Wiley Online Library, v. 42, n. 4, p. 73–93, 1992.
- TAO, F. et al. Digital twin-driven product design, manufacturing and service with big data. *The International Journal of Advanced Manufacturing Technology*, Springer, v. 94, n. 9-12, p. 3563–3576, 2018.
- TAO, F.; ZHANG, M. Digital twin shop-floor: a new shop-floor paradigm towards smart manufacturing. *Ieee Access*, IEEE, v. 5, p. 20418–20427, 2017.
- TIWARI, V. et al. Augmented reality and its technologies. *International Research Journal of Engineering and Technology (IRJET)*, v. 3, n. 4, 2016.
- TOSHNIWAL, R.; DASTIDAR, K. G. Virtual reality: The future interface of technology. *International Journal of Computer Science and Information Technologies*, v. 5, p. 7032–7034, 2014.

- TRAPPEY, A. J. et al. A review of essential standards and patent landscapes for the internet of things: A key enabler for industry 4.0. *Advanced Engineering Informatics*, Elsevier, v. 33, p. 208–229, 2017.
- UHLEMANN, T. H.-J. et al. The digital twin: Demonstrating the potential of real time data acquisition in production systems. *Procedia Manufacturing*, Elsevier, v. 9, p. 113–120, 2017.
- VACHÁLEK, J. et al. The digital twin of an industrial production line within the industry 4.0 concept. In: IEEE. *Process Control (PC), 2017 21st International Conference on*. [S.l.], 2017. p. 258–262.
- VISWANATHAN, J. et al. Using hybrid process simulation to evaluate manufacturing system component choices: integrating a virtual robot with the physical system. In: WINTER SIMULATION CONFERENCE. *Proceedings of the Winter Simulation Conference*. [S.l.], 2011. p. 2827–2838.
- ZANNI, A. *Cyber-physical systems and smart cities*. 2015. Disponível em: <<https://developer.ibm.com/articles/ba-cyber-physical-systems-and-smart-cities-iot/>>. Acesso em: 01 set. 2019.
- ZHENG, Y.; YANG, S.; CHENG, H. An application framework of digital twin and its case study. *Journal of Ambient Intelligence and Humanized Computing*, Springer, v. 10, n. 3, p. 1141–1153, 2018.

# Apêndices

# APÊNDICE A – Lista de Entradas e Saídas do Processo Industrial

Este Apêndice apresenta a listagem de entradas e saídas dos módulos *Testing* e *Sorting* integrantes do conjunto Festo MPS 200 (Tabela 10). Os endereços estão separados em dois grandes grupos, Físico e Virtual, que correspondem aos espaços de endereçamentos reservados pelo *OpenPLC* para o uso dos dispositivos escravos (ESP8266 ou outros suportados) e para as aplicações que envolvem comunicação UDP. Os endereços a partir de 100.0 são dedicados aos dispositivos escravos, enquanto a comunicação UDP é iniciada nos endereços 0.0.

A coluna “*Label*” mostra os nomes atribuídos às variáveis para fins de declaração no diagrama *Ladder*.

Cabe ainda ressaltar que os endereços da coluna “*UDP in*” estão relacionados com blocos do tipo “*Send*” nos aplicativos de simulação, enquanto os endereços da coluna “*UDP out*” estão relacionados com blocos do tipo “*Receive*”.

Tabela 10 – Lista de Entradas e Saídas do sistema.

<b>Dispositivo</b>	<b>Input</b>	<b>Output</b>	<b>UDP</b>	<b>UDP</b>	<b>Label</b>
			In	Out	
<b><i>Sorting - E/S físico (ESP)</i></b>					
Sens. de presença	IX100.0				Presenca
Sens. capacitivo	IX100.1				Capacitivo
Sens. indutivo	IX100.2				Indutivo
Sens. fim-de-curso	IX100.3				Fim
Esteira		QX100.0			Esteira
Desviador 1		QX100.1			Desv1
Desviador 2		QX100.2			Desv2
Trava		QX100.3			Trava
<b><i>Sorting - E/S virtual (SimLink)</i></b>					
S. pres. (real)		QX0.0 QX0.4		10000 10004	Presenca_sl Presenca_vs
S. capacit. (real)		QX0.1 QX0.5		10001 10005	Capacitivo_sl Capacitivo_vs

*Continua na página seguinte*

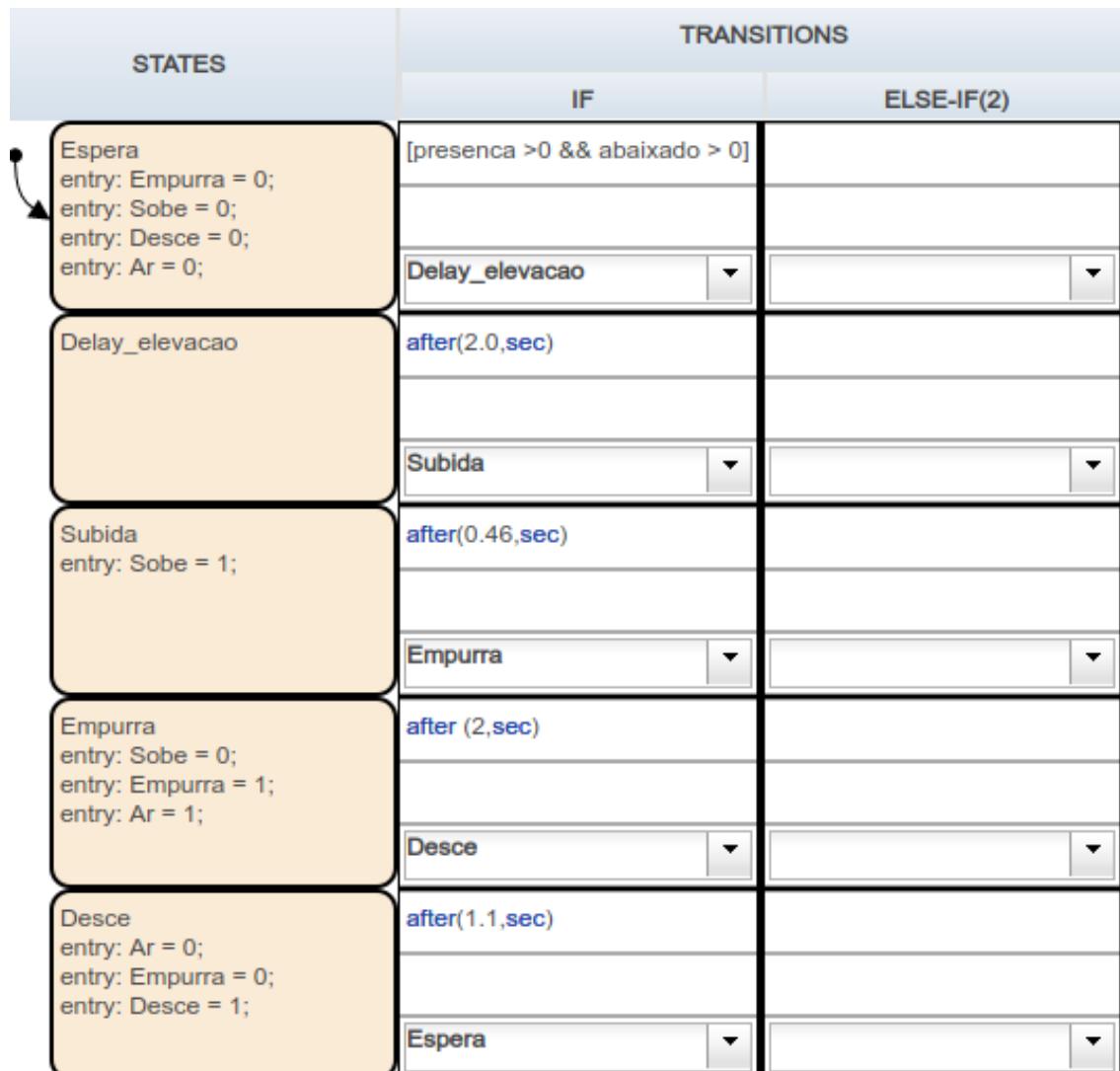
S. indut. (real)		QX0.2 QX0.6		10002 10006	Indutivo_sl Indutivo_vs
S. fim de curso (real)		QX0.3 QX0.7		10003 10007	Fim_sl Fim_vs
Esteira (sim.)	IX0.0		10100		Est_sim
Desviador 1 (sim.)	IX0.1		10101		Desv1_sim
Desviador 2 (sim.)	IX0.2		10102		Desv2_sim
Trava (sim.)	IX0.3		10103		Trava_sim
Esteira (real)		QX1.0		10008	Esteira_vs
Desv. 1 (real)		QX1.1		10009	Desv1_vs
Desv. 2 (real)		QX1.2		10010	Desv2_vs
Trava (real)		QX1.3		10011	Trava_vs
<b>Testing - E/S físico (ESP)</b>					
Sensor de presença	IX101.0				s_peca
Sensor de plataforma abaixada	IX101.1				s_elevado
Sensor de plataforma elevada	IX101.2				s_abaiiado
Sensor de pistão recuado	IX101.3				s_recuado
Descer plataforma		QX101.0			desce
Subir plataforma		QX101.1			sobe
Empurrar a peça		QX101.2			empurra
Trilho de ar		QX101.3			ar
<b>Testing - E/S virtual (SimLink)</b>					
S. presença (real)		QX1.0		11000	s_peca_ref
S. plataforma abaixada (real)		QX1.1		11001	s_abaiiado_ref
S. plataforma elevada (real)		QX1.2		11002	s_elevado_ref
S. de pistão recuado (real)		QX1.3		11003	s_recuado_ref
Desce plataforma (sim.)	IX0.4		11100		desce_sim
Sobe plataforma (sim.)	IX0.5		11101		sobe_sim
Empurra a peça (sim.)	IX0.6		11102		empurra_sim
Trilho de ar (sim.)	IX0.7		11103		ar_sim
Desce plataforma (real)		QX1.4		11004	desce_real
Sobe plataforma (real)		QX1.5		11005	sobe_real
Empurra a peça (real)		QX1.6		11006	empurra_real
Trilho de ar (real)		QX1.7		11007	ar_real

Fonte: Autoria própria.

# APÊNDICE B – Tabelas de Estados e Transições

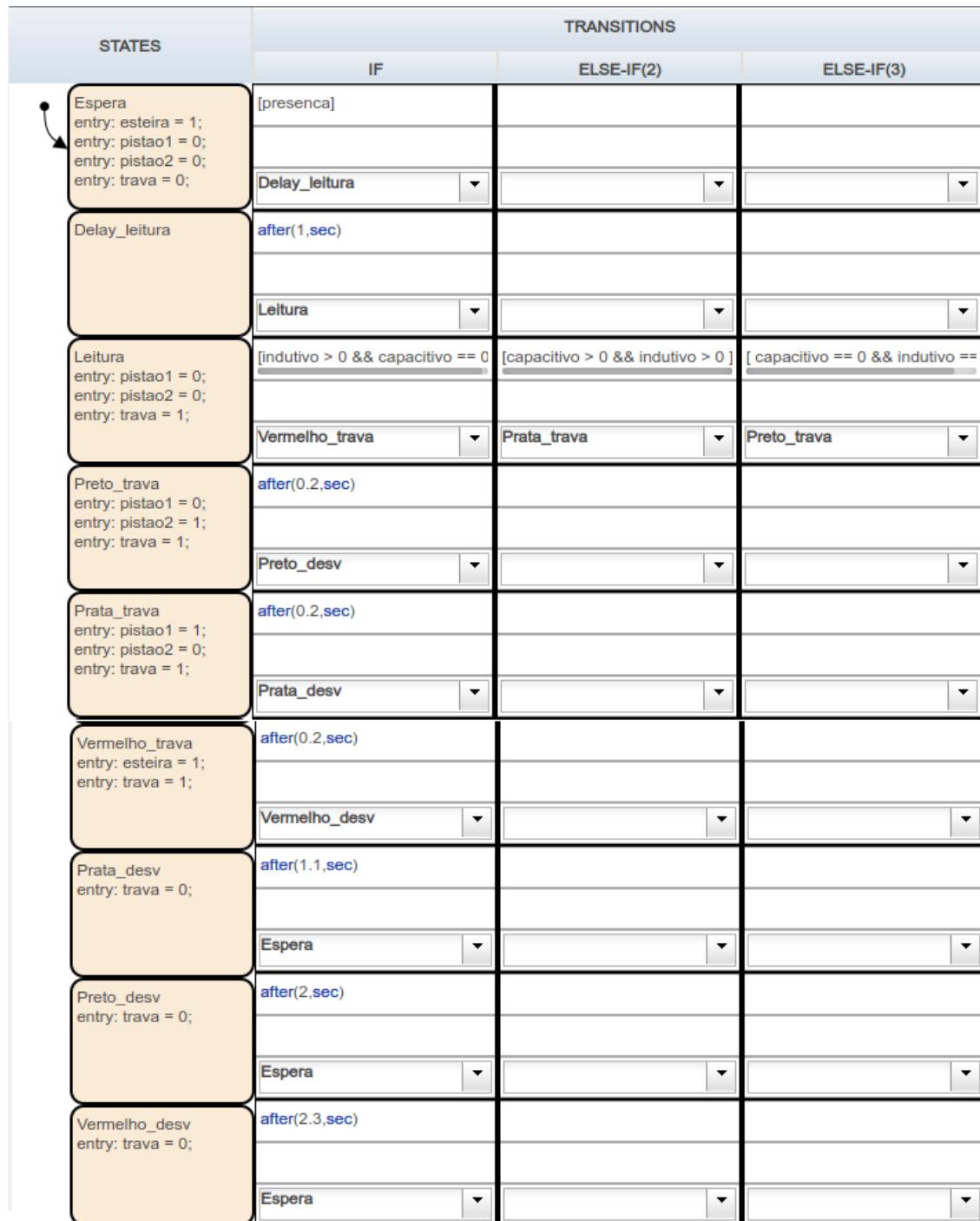
Este Apêndice apresenta as Tabelas de Estados e Transições (*State-Transition Tables*) desenvolvidos para as estações de trabalho *Testing* e *Sorting* integrantes do conjunto Festo MPS 200. No esquemático final são adicionados os blocos dos diagramas, bem como as portas UDP *Send* e UDP *Receive*. Os *displays* são colocados apenas para depuração durante a fase de testes e podem ser removidos sem prejuízo à execução.

Figura 59 – Diagrama de Estados e Transições desenvolvido para a modelagem dos estados da estação de trabalho *Testing*.

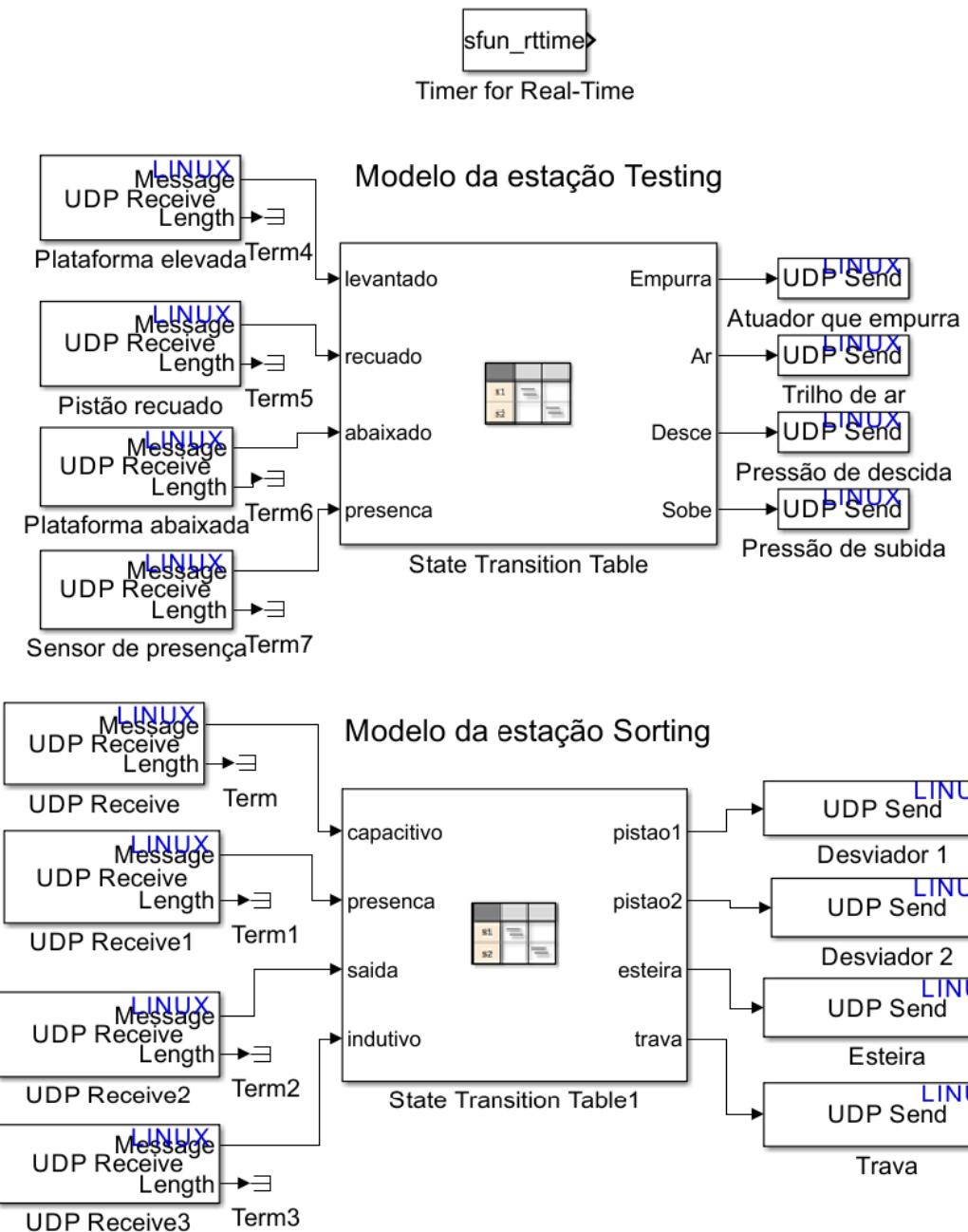


Fonte: Autoria própria.

Figura 60 – Diagrama de Estados e Transições desenvolvido para a modelagem dos estados da estação de trabalho *Sorting*.



Fonte: Autoria própria.

Figura 61 – Esquemático completo de simulação implementado no software *Simulink*.

Fonte: Autoria própria.

# APÊNDICE C – Código em C# implementado no Visual Studio

Como descrito no corpo do texto, foi necessário reescrever a classe “ClienteTCP” de forma a implementar os clientes UDP correspondentes aos dados de sensores e atuadores da planta de automação. O código que se segue foi incorporado ao *software* de Santos e Carvalho (2016).

```

using System ;
using System . Collections . Generic ;
using System . Text ;
using System . Net . Sockets ;
using System . Xml ;
using System . Threading ;
using System . Xml ;
using System . Threading ;
using System . Net ;

namespace Cliente
{
    public class ClienteTCP
    {
        private string returndata;

        public void UdpUpdate()
        {
            byte [ ] simlinktrue = new byte [2] { 1 , 0 };
            //byte para comparar com os dados que recebo via UDP

            try
            {

// Abertura das portas UDP que serao lidas
            UdpClient presenca = new UdpClient (10004);
            UdpClient capacitivo = new UdpClient (10005);
            UdpClient indutivo = new UdpClient (10006);
        }
    }
}
```

```
UdpClient fimdecurso = new UdpClient(10007);

UdpClient esteira = new UdpClient(10008);
UdpClient desv1 = new UdpClient(10009);
UdpClient desv2 = new UdpClient(10010);
UdpClient trava = new UdpClient(10011);

//Atualiza o ambiente de variaveis globais conforme as leituras
//das portas UDP

GlobalVariables.novapec aValue = UDPReader(presenca);
GlobalVariables.materialValue = UDPReader(capacitivo);
GlobalVariables.corValue = UDPReader(indutivo);
GlobalVariables.fimdecursoValue = UDPReader(fimdecurso);

GlobalVariables.esteiraValue = UDPReader(esteira);
GlobalVariables.desviador1Value = UDPReader(desv1);
GlobalVariables.desviador2Value = UDPReader(desv2);
GlobalVariables.stopperValue = UDPReader(trava);

// Nega flags que indicam acionamento pelo usuario para que seja
//iniciado um novo ciclo de leitura de inputs

GlobalVariables.sensoresFlag = "0";
GlobalVariables.esteiraFlag = "0";
GlobalVariables.desviador1Flag = "0";
GlobalVariables.desviador2Flag = "0";
GlobalVariables.stopperFlag = "0";
GlobalVariables.painelFlag = "0";
GlobalVariables.ledsFlag = "0";

}

catch (Exception ex)
{
    returndata = ex.Message;
}
```

```
//funcao de chamada UDP – recebe a porta udp e diz se o
//valor recebido por ela e' true ou false

string UDPReader( UdpClient p )
{
    IPEndPoint RemoteIpEndPoint = new IPPEndPoint( IPAddress.Any, 0 );
    byte[] receiveBytes = p.Receive( ref RemoteIpEndPoint );

    if ( receiveBytes[0] == simlinktrue[0] )
    {
        return ( "1" );
    }
    else
    {
        return ( "0" );
    }

}
}
}
}
```