CENTRO UNIVERSITÁRIO FEI
GABRIEL BASTOS DE MIRANDA

COMISSIONAMENTO VIRTUAL DE CÉLULA ROBOTIZADA UTILIZANDO APRENDIZADO DE MÁQUINA

São Bernardo do Campo
2020

GABRIEL BASTOS DE MIRANDA

COMISSIONAMENTO VIRTUAL DE CÉLULA ROBOTIZADA UTILIZANDO APRENDIZADO DE MÁQUINA

Dissertação apresentada ao Centro Universitário FEI, como parte dos requisitos necessários para obtenção do título de Mestrado em Engenharia Mecânica. Orientado pelo Prof. Dr. Fabio Lima.

São Bernardo do Campo

2020

centro universitário FEI

APRESENTAÇÃO DE DISSERTAÇÃO
ATA DA BANCA EXAMINADORA

Programa de Pós-Graduação Stricto Sensu em Engenharia Mecânica

Mestrado

PGM-10

Aluno: Gabriel Bastos de Miranda **Matrícula:** 218115-4

Título do Trabalho: Comissionamento virtual de célula robotizada utilizando aprendizagem de máquina.

Área de Concentração: Produção

Orientador: Prof. Dr. Fábio Lima

Data da realização da defesa: 27/02/2020

ORIGINAL ASSINADA

Avaliação da Banca Examinadora:

_____

_____

_____

_____

_____

_____

São Bernardo do Campo, / / .

MEMBROS DA BANCA EXAMINADORA	
Prof. Dr. Fábio Lima	Ass.: _____
Prof. Dr. Fabrizio Leonardi	Ass.: _____
Prof. Dr. Marco Antonio Fumagalli	Ass.: _____

A Banca Julgadora acima-assinada atribuiu ao aluno o seguinte resultado:

APROVADO ☒ REPROVADO ☐

VERSÃO FINAL DA DISSERTAÇÃO

APROVO A VERSÃO FINAL DA DISSERTAÇÃO EM QUE FORAM INCLUÍDAS AS RECOMENDAÇÕES DA BANCA EXAMINADORA

_____

Aprovação do Coordenador do Programa de Pós-graduação

_____

Prof. Dr. Rodrigo Magnabosco

Dedico este trabalho a minha família, esposa e amigos.

AGRADECIMENTOS

Agradeço primeiramente ao Centro Universitário FEI, que acreditou em mim e me deu a oportunidade da realização dessa pesquisa.

Ao meus pais, que sempre me serviram como fonte de inspiração e referência.

A minha esposa, que esteve sempre ao meu lado e foi de fundamental importância na realização desse trabalho.

Ao Prof. Dr. Fábio Lima, por acreditar nessa pesquisa, pela orientação, ajuda e disponibilidade durante todo o trabalho.

E a todos os que contribuíram direta ou indiretamente na execução deste trabalho.

“Nós só podemos ver um pouco do futuro, mas o suficiente para perceber que há muito a fazer.”

Alan Turing

RESUMO

O mundo vivencia uma mudança sem precedentes quanto as tecnologias da informação e processos industriais. A nova era da indústria 4.0 em conjunto com novas aplicações de inteligência artificial fomentam uma série de novas áreas de interesse para a indústria, questionando-se se os métodos convencionais que compõem os processos industriais continuarão os mesmos. Visando obter respostas para alguns desses questionamentos, este trabalho investiga a viabilidade na substituição de um controlador lógico programável por um sistema de redes neurais no controle de eventos discretos de uma célula robotizada. Desta forma, a pesquisa avalia os procedimentos necessários para que essa implementação seja possível, analisando os problemas encontrados, limitações, dificuldades e vantagens comparadas aos métodos convencionais. Para tal análise, são utilizadas ferramentas de manufatura digital que permitem realizar o comissionamento virtual de uma célula robotizada para simulação e validação do modelo. O modelo proposto teve seu controle por meio de um sistema de redes neurais artificiais, do qual, os dados de treinamento foram extraídos a partir de observações de um modelo de lógica booleana. Ao término desta pesquisa, foi possível elaborar os procedimentos necessário para o método proposto, que se mostrou viável diante da aplicação realizada, e compará-la com os métodos convencionais de programação.

Palavras-chave: Comissionamento virtual, Redes neurais artificiais, Manufatura Digital, Industria 4.0.

ABSTRACT

The world is experiencing an unprecedented change in information technologies and industrial processes. The new era of industry 4.0 together with new applications of artificial intelligence foster a series of new areas of interest for the industry, questioning whether the conventional methods that make up industrial processes will remain the same. In order to obtain answers to some of these questions, this work investigates the feasibility of replacing a programmable logic controller with a system of neural networks in the control of discrete events in a robotic cell. Thus, the research evaluates the necessary procedures for this implementation to be possible, analyzing the problems found, limitations, difficulties and advantages compared to conventional methods. For this analysis, digital manufacturing tools are used that allow the virtual commissioning of a robotic cell for simulation and model validation. The proposed model had its control through a system of artificial neural networks, from which, the training data were extracted from observations of a Boolean logic model. At the end of this research, it was possible to elaborate the necessary procedures for the proposed method, which proved to be feasible in view of the application carried out, and to compare it with conventional programming methods.

Keywords: Virtual commissioning, Artificial neural networks, Digital Manufacturing, Industry 4.0.

LISTA DE ILUSTRAÇÕES

Figura 1 – Componentes da indústria 4.0 19
Figura 2 – Fábrica digital 22
Figura 3 – Controlador lógico programável moderno 24
Figura 4 – Unidade de processamento neural 26
Figura 5 – Comissionamento virtual de uma célula 28
Figura 6 – Neurônio biológico 29
Figura 7 – Rede *Perceptron* multicamadas 30
Figura 8 – Diminuição do erro durante o treinamento 32
Figura 9 – Tabela da verdade 36
Figura 10 – Portas lógicas 36
Figura 11 – Visão do software de simulação 38
Figura 12 – Imagem da célula robótica estudada 39
Figura 13 – Modelo virtual da célula 40
Figura 14 – Fluxograma de processo 41
Figura 15 – Sinais criados no modelo de simulação 42
Figura 16 – Esquema de comunicação 43
Figura 17 – Apontamento para servidor OPC 44
Figura 18 – Sistema por controle booleano 46
Figura 19 – Módulo de treinamento neural 48
Figura 20 – Erro de treinamento com 3 neurônios 49
Figura 21 – Erro de treinamento com 5 neurônios 51
Figura 22 – Erro de treinamento com 8 neurônios 53
Figura 23 – Estrutura da rede neural utilizada 55
Figura 24 – Sistema controlado por rede neural 56
Figura 25 – Taxa de atualização 61
Figura 26 – Célula sendo executada por meio de lógica booleana 62
Figura 27 – Célula sendo executada por meio de rede neural 62

LISTA DE TABELAS

Tabela 1 – Lista de sinais 42
Tabela 2 – Dados exportados 47
Tabela 3 – Rede com 3 neurônios 50
Tabela 4 – Rede com 5 neurônios 52
Tabela 5 – Rede com 8 neurônios 54
Tabela 6 – Seleção em 0,1 57
Tabela 7 – Seleção em 0,2 58
Tabela 8 – Seleção em 0,25 59
Tabela 9 – Resultados obtidos 63

LISTA DE ABREVIATURAS

CAD	*Computer-Aided Design*
CLP	Controlador Lógico Programável
CPS	Sistema Cyber-Físico
IoT	Internet Das Coisas
NPU	Unidade de Processamento Neural
OPC	*Open Platform Communications*
RNA	Rede Neural Artificial
RRS	*Real Robot Simulation*
VC	Comissionamento Virtual

LISTA DE SÍMBOLOS

x	Sinal de entrada
w	Peso sináptico
$\sum$	Combinador linear
θ	Limiar de ativação
u	Potencial de ativação
g	Função de ativação
y	Sinal de saída
E	Erro
T	Transposto
p	Amostra
N	Dimensão
k	Plano conjugado
α	Momentum
q	Quadrático

SUMÁRIO

1. INTRODUÇÃO 14
1.1. OBJETIVOS 14
1.2. JUSTIFICATIVA 15
1.3. METODOLOGIA 15
2. REVISÃO DA LITERATURA 16
2.1. INDÚSTRIA 4.0 16
2.2. MANUFATURA DIGITAL 20
2.3. INTELIGÊNCIA ARTIFICIAL 22
3. FUNDAMENTAÇÃO TEÓRICA 24
3.1. CONTROLADORES LÓGICOS PROGRAMÁVEIS 24
3.2. COMISSIONAMENTO VIRTUAL 27
3.3. REDES NEURAIS ARTIFICIAIS 29
3.4. TREINAMENTO NEURAL 31
3.5. COMUNICAÇÃO OPC 34
3.6. LÓGICA BOOLEANA 35
4. MODELAGEM 37
4.1. SIMULAÇÃO 37
4.1.1. Software de simulação 37
4.1.2. A célula simulada 38
4.1.3. Sinais 41
4.2. INTEGRAÇÃO 43
4.3. CONTROLE 45
4.3.1. Software de controle 45
4.3.2. Controle *booleano* 45
4.4. REDES NEURAIS 48
4.4.1. Treinamento 48
4.4.2. Controle por redes neurais 56
5. RESULTADOS 61

5.1. MODELO BOOLEANO 61
5.2. MODELO NEURAL 62
6. CONCLUSÕES 65
REFERÊNCIAS 67

1. INTRODUÇÃO

Desde a revolução industrial, a tecnologia voltada à manufatura vem desenvolvendo-se a passos largos. Iniciou-se com a introdução de máquinas à produção manual, caracterizando a primeira revolução, passando pela racionalização de Taylor e esteira de Henry Ford, que se caracterizam pela linha de montagem, protagonizando a segunda revolução, chegando então à terceira revolução que foi marcada pelo desenvolvimento de tecnologias da informação (WANG, 2016).

A indústria de manufatura vivencia um aumento sem precedentes de dados disponíveis. Esses dados surgem nos mais variados formatos, como em um sensor de uma linha de produção, um dado do ambiente ou parâmetros de uma ferramenta (WUEST et al., 2016). Consequentemente, o mundo industrial está passando por profundas mudanças com o desdobramento da era da informação. A vantagem competitiva na manufatura alterou-se do paradigma da produção em massa para um baseado na flexibilidade e de rápida reposta (LEE, W. B.; CHEUNG; LI, 2001).

As primeiras três revoluções industriais trouxeram a mecanização, eletricidade e a tecnologia da informação para os processos de manufatura. Entretanto, os sistemas cyber-físicos e a internet das coisas são tecnologias que obtiveram um grande avanço na última década, e com esse desenvolvimento, um novo conceito foi introduzido na Alemanha em 2011, a indústria 4.0, que simboliza o começo da quarta revolução industrial (QIN; LIU; GROSVENOR, 2016).

Visando obter respostas quanto as tecnologias que poderiam ser utilizadas nos novos modelos da indústria 4.0, este trabalho investiga a utilização e implementação de aprendizado de máquina no controle de processos industriais automáticos.

1.1. OBJETIVOS

Este trabalho se propõe a averiguar a viabilidade da substituição de controladores lógicos programáveis convencionais por um sistema de controle por aprendizado de máquina, avaliando os procedimentos necessários para que essa implementação seja possível, analisando os problemas encontrados, limitações, dificuldades e vantagens comparadas aos métodos convencionais. Dentre os objetivos específicos, podem-se destacar:

a) Criação de um procedimento para aplicação do aprendizado de máquina no controle de células automáticas.
b) Com a realização de um comissionamento virtual, verificar o correto funcionamento do sistema por redes neurais ao controlar a célula.
c) Contribuir com a literatura quanto a implementação do aprendizado de máquina nos processos industriais.

1.2. JUSTIFICATIVA

Com a introdução de modelos industriais trazidos dos conceitos da indústria 4.0 e o surgimento de aplicações de inteligência artificial no cotidiano, há uma reflexão de como essas tecnologias poderiam ser utilizadas na indústria.

Em contraste aos métodos já consolidados na utilização dos controladores lógicos programáveis, essa pesquisa propõe um modelo alternativo de programação e controle de células automáticas a partir da utilização de uma rede neural em substituição das linguagens convencionais. A utilização de modelos como esses introduzem a possibilidade de o controlador aprender qual deve ser o comportamento esperado da linha a partir de observações de dados, e após aprendido, esse comportamento poderia ser compilado em um modelo matemático de controle.

1.3. METODOLOGIA

Essa pesquisa foi realizada sob uma abordagem quantitativa quanto a análise da viabilidade na substituição de um controlador lógico programável por um sistema de aprendizado de máquina. O caráter exploratório dessa pesquisa traz como objetivos aprimorar o conhecimento sobre o conteúdo abordado, e experimentar novas possibilidades para conceitos já concretizados.

É utilizada a modelagem e simulação de uma célula robotizada automática, que contém todos elementos de recursos, processos e sinais para seu correto funcionamento, tal como a célula real. Após a realização de uma revisão na literatura, percebe-se que a adoção de um comissionamento virtual da célula modelada junto ao sistema proposto é fundamental para avaliar sua funcionalidade e validar o método proposto. Para tanto, esse trabalho apresenta os procedimentos para realização dessa integração entre os *softwares* de simulação e de controle.

2. REVISÃO DA LITERATURA

2.1. INDÚSTRIA 4.0

Tecnologias de informação e comunicação estão em rápido desenvolvimento, e muitas dessas tecnologias, como computação em nuvem, internet das coisas, *big data* e inteligência artificial, surgiram. Esse avanço tecnológico adentra a indústria manufatureira e permitem a junção do mundo real com o virtual por meio dos sistemas cyber-físicos, que marca o advento do quarto estágio da produção industrial (KHAN; TUROWSKI, 2016).

A primeira revolução industrial é o termo utilizado para se descrever as mudanças do trabalho puramente manual para a produção com máquinas, que revolucionaram os processos existentes e alteraram o curso do desenvolvimento social e existencial humano, principalmente após a máquina a vapor ser inventada por James Watt, em 1782, onde foi possível se obter energia mecânica através da transformação da energia térmica., oferecendo condições de progresso melhores à sociedade fabril do período (LIMA, 2017).

A segunda revolução industrial foi caracterizada pelo princípio da racionalização de Frederick Winslow Taylor, através deste, a metodologia de funcionamento dos processos passou a ser seriada, surgindo assim as primeiras linhas de montagem, em que cada etapa é responsável por uma parcela na formação do produto. A padronização se tornou característica marcante nesta nova metodologia abrangendo os produtos e seus respectivos processos produtivos, manufatura de precisão. Henry Ford utilizou a primeira esteira transportadora e obteve sucesso pioneiro com ela na fabricação de automóveis, no início do século XX (WANG, 2016).

A terceira revolução industrial ocorreu ainda no século XX porém na década de 60, a partir de um aquecimento do mercado americano através do plano Marshall, investimentos econômicos na área impulsionaram o desenvolvimento de novas tecnologias, também conhecida como revolução digital ou computacional possibilitou, por exemplo, o estudo e avanço de áreas correlacionadas como a tecnologia de informação (TI), estas inovações levaram às máquinas controladas numericamente, como os robôs industriais, que podem ter suas programações de aplicação modificadas com uma velocidade superior às máquinas industriais convencionais, desta forma viabilizou-se sistemas de produção flexíveis e de alta produtividade, que também são características desta revolução (WANG, 2016).

A quarta revolução industrial ou indústria 4.0, é uma iniciativa estratégica do governo alemão que, segundo Schwab (2016), é a conificação entre os mundos digital, físico, biológico e suas respectivas tecnologias, focada na criação de fábricas inteligentes, onde tecnologias são atualizadas e transformadas pelos sistemas cyber-físico (CPS), internet das coisas (IoT) e computação na nuvem. Na era da indústria 4.0, sistemas de manufatura são capazes de monitorar processos físicos por meio do que se é chamado de "gêmeo digital" do mundo físico, e tomar decisões inteligentes por meio de comunicação em tempo real e cooperação com humanos, máquinas e sensores. A indústria 4.0 combina tecnologias de sistemas de produção embarcado com processos de produção inteligentes, pavimentando o caminho para uma nova era tecnológica que transformará modelos de indústria, produção e negócios (ZHONG et al., 2017).

De uma forma geral, a indústria 4.0 é um termo coletivo para tecnologias e conceitos, entre eles, a fábrica inteligente, com um monitor cyber-físico de processos, uma cópia virtual do mundo físico, permitindo-se tomar decisões decentralizadas. Com a internet das coisas, um sistema cyber-físico pode se comunicar e cooperar uns com os outros e com humanos em tempo real. Além da mineração de dados, que adquire informações que suportam os processos de tomadas de decisões (WANG, 2016).

A manufatura inteligente (*smart manufacturing*) é um amplo conceito de manufatura que propõe a otimização da produção e produtos, fazendo-se uma completa utilização da informação avançada e das tecnologias de manufatura. Ela é considerada como um novo modelo, baseada nas tecnologias inteligentes que gradativamente melhoram o design, produção, gerenciamento e integração de todo ciclo de vida de um produto. O ciclo de vida de um produto pode ser facilitado utilizando-se sensores inteligentes, modelos de tomada de decisões adaptativos, materiais avançados, dispositivos inteligentes e análise de dados. A eficiência da produção, qualidade do produto e o nível de serviços serão, consequentemente, aperfeiçoados (ZHONG et al., 2017).

Já a internet das coisas (IoT) se refere a um mundo conectado onde vários objetos embarcados com sensores eletrônicos, atuadores ou outro dispositivo digital, podem ser conectados à rede para propósitos de coleta e troca de dados. A internet das coisas oferece uma conectividade avançada dos objetos físicos, sistemas e serviços, permitindo uma comunicação objeto a objeto (ZHONG et al., 2017). A IoT permite a coleta de dados de vários recursos de manufatura, que são convertidos em um objeto inteligente de manufatura (SMO), que pode

interagir de forma inteligente um com o outro, permitindo uma coleta de dados de produção em tempo real (ZHENG et al., 2018).

O modelo de referência de IoT é composto por um conjunto de outros modelos, cada qual atendendo necessidades especificas de implementação nas aplicações (LEITE; MARTINS, 2017). Estes submodelos estão descritos com suas respectivas responsabilidades a seguir:

- Modelo de domínio: identificação e agrupamento dos sistemas;
- Modelo de informação: transporte das informações geradas pelos sensores, ou equipamentos que estejam no sistema com esta função, por exemplo identificações por rádio frequência (RFID);
- Modelo de funcionalidade: agrupamento das funções do sistema;
- Modelo de comunicação: divisão da comunicação em 7 camadas de protocolo, seguindo o modelo de referência para interconexão de sistemas abertos (OSI), regulamentado pela organização internacional de normalização (ISO);
- Modelo de confiança, segurança e privacidade: agrupamento das funcionalidades de segurança da informação, privacidade e certificação.

Com a virtualização de recursos, a computação em nuvem proporciona a infraestrutura, plataforma e *software* como serviços. A computação em nuvem é um modelo que permite uma onipresente e conveniente gama de dados configurados à demanda do acesso à rede, que rapidamente podem abastecer um prestador de serviços com o mínimo esforço ou interação gerencial (GREEF; DUARTE FREITAS, 2015).

A manufatura em nuvem se refere a um modelo avançado de manufatura sobre o suporte da computação em nuvem, internet das coisas, virtualização, e tecnologia de serviços orientados, que transformam os recursos de manufatura em serviços que podem ser compartilhados de forma compreensível. (ZHONG et al., 2017)

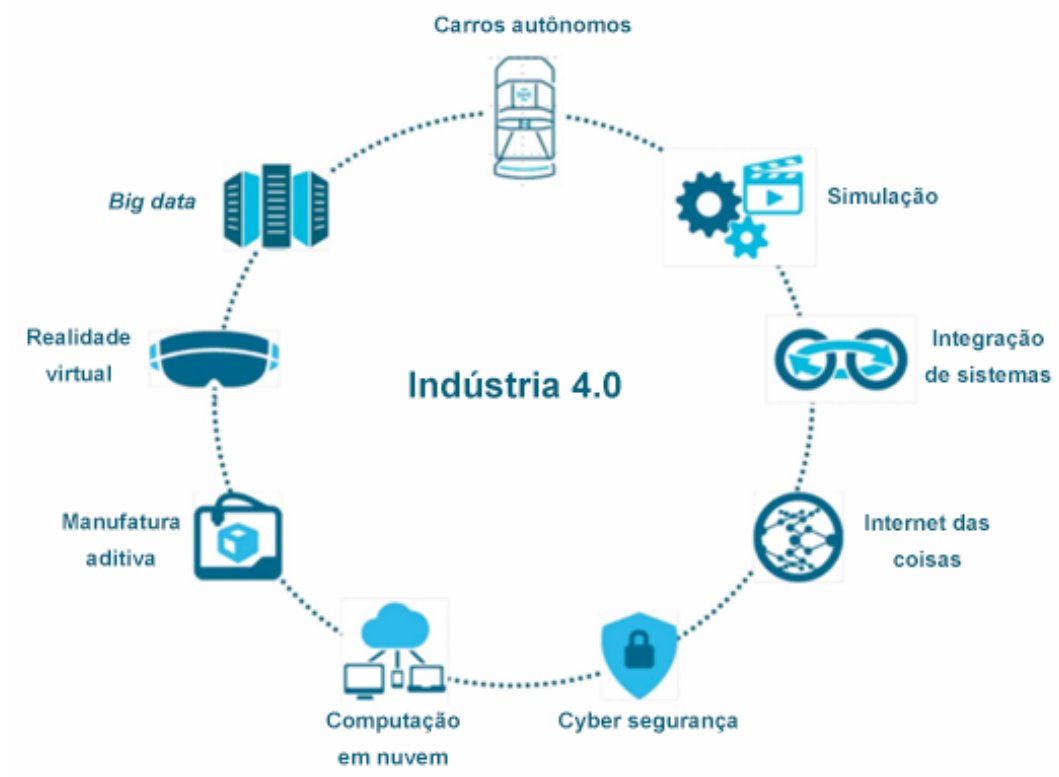
Já um sistema cyber-físico (CPS) colabora com as entidades computacionais que estão em conexão com seu mundo e processos físicos, proporcionando e utilizando serviços de processamento e acesso de dados disponíveis na internet. Um sistema de produção CPS depende dos mais recentes desenvolvimentos em informática e ciência da manufatura (GREEF; DUARTE FREITAS, 2015). Um sistema cyber-físico é definido como uma tecnologia de transformação para gerenciar sistemas interconectados entre seus recursos físicos e capacidades computacionais, e com seu recente desenvolvimento, tem-se como resultado uma maior

disponibilidade e acessibilidade dos sensores, sistemas de aquisição de dados e redes de computadores (LEE, Jay; BAGHERI; KAO, 2015).

A natureza competitiva das indústrias atuais, forçam as fábricas a irem em meio a implementação de metodologias de alta tecnologia. Consequentemente, o crescimento da utilização de sensores e máquinas conectadas resultaram em uma geração continua de um grande volume de dados, conhecido como *Big Data* (LEE, Jay; BAGHERI; KAO, 2015).

O Ponto chave na quarta revolução industrial é criar valor a enorme quantidade de informação gerada por esses vários objetos (ARAS et al., 2018). Para isso, uma análise desses dados é crucial para o desenvolvimento e operação dos sistemas de manufatura, como por exemplo, uma estrutura para avaliação de riscos baseada nos dados do sistema de produção em tempo real. Esse tópico tem sido amplamente discutido no que tange ao suporte para otimização da produção e visualização dos sistemas cyber-físicos de manufatura (ZHENG et al., 2018).

Figura 1 – Componentes da indústria 4.0



Fonte: Adaptado de ARAS et al., 2018

2.2. MANUFATURA DIGITAL

A necessidade na redução do tempo de desenvolvimento junto ao crescimento da demanda para mais produtos orientados ao consumidor tem levado à próxima geração dos sistemas de tecnologia da informação na manufatura. As organizações têm se esforçado em integrar seus negócios e departamentos com os novos sistemas baseados nos conceitos da fábrica/manufatura digital, de acordo com as tecnologias de simulação e gerenciamento de produção utilizadas para otimizar seus processos (CHRYSSOLOURIS et al., 2009).

O suporte da tecnologia é imprescindível para a integração entre três fatores: recurso, processo e produto, a capacidade de análise das informações, e parâmetros relacionados a estes fatores isoladamente e integrados possibilitam além da concepção do próprio conceito de manufatura digital, sua implementação e principalmente soluções capazes de atender as demandas do mercado (ZHOU; XIE; CHEN, 2012).

Manufatura virtual é uma tecnologia de sistema computacional que integra atividades de manufatura com modelos e simulações, ao invés de objetos e suas operações do mundo real. Isso proporciona uma ferramenta para otimização da eficiência da produção por meio, principalmente, de simulações para se começar a produção efetiva (LEE, W. B.; CHEUNG; LI, 2001).

O conceito de fábrica digital inclui uma rede de modelos digitais, métodos e ferramentas, como a simulação e a visualização 3D, que estão integrados a dados compreensíveis. A visão do conceito da fábrica digital foca na integração dos métodos e ferramentas disponíveis em diferentes níveis para planejar e testar o produto e seu processo, ainda na fase de desenvolvimento. Uma fábrica digital integra: desenvolvimento de produto, desenvolvimento dos processos de produção, desenho da planta e planejamento de produção (KHAN; TUROWSKI, 2016).

KA (2013) listou as classes de um modelo de fabrica digital em:

- Produtos: Modelam os dados relacionados ao produto desejado.
- Recursos: Modelam os dados dos recursos de produção utilizados no sistema com objetivo de transformar o produto.
- Processos: Modelam os dados referentes aos processos adotados no sistema para a transformação do produto.

- Sistemas: Modelam os dados de um sistema de transformação que afetam o produto com os recursos do processo.
- Construções: Modelam os dados relativos à estrutura física da fábrica e layout.

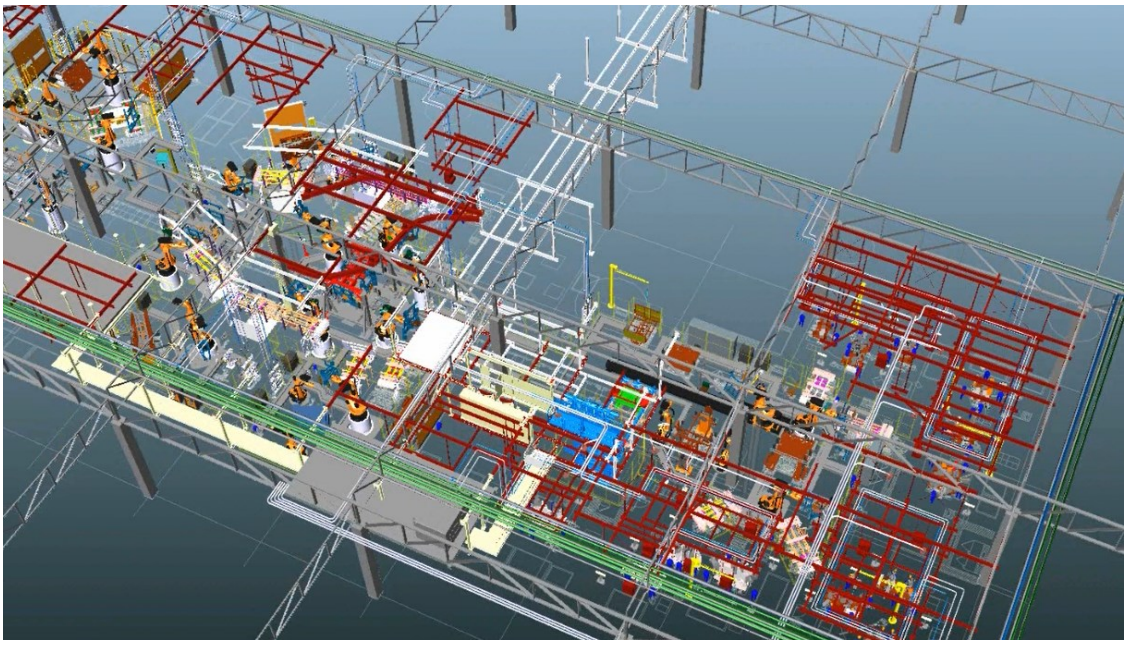
Os sistemas CAD (*computer-aided design*) são considerados, entre as tecnologias, a que impulsionou a produtividade, reduzindo drasticamente o tempo do produto até o mercado. As interfaces disponíveis no mercado atualmente apresentam funcionalidades de análises de elementos finitos, cinemática, dinâmica e simulação completa das propriedades geométricas dos materiais. Com isso, os sistemas CAD se tornaram indispensáveis nas companhias de manufatura atuais. Juntamente com o CAD, a simulação computacional se tornou uma técnica amplamente utilizada no desenvolvimento dos sistemas de manufatura, permitindo ao engenheiros e tomadores de decisões investigarem a complexidade de seus sistemas e a forma como as alterações nas configurações podem afetar seu desempenho (CHRYSSOLOURIS et al., 2009).

Ferramentas de simulação de eventos discretos podem simular a planta e seus processos, permitindo ao sistema aperfeiçoar o fluxo de material, utilização de recursos e logística em todos os níveis do planejamento da planta (KHAN; TUROWSKI, 2016).

A fim de se aperfeiçoar o modelamento, simulação e programação *offline* de uma aplicação de um robô em respeito à sua dinâmica e da acurácia na sua posição absoluta, um controlador de robô virtual permite integrar o software original do controlador industrial do robô dentro da interface do sistema de simulação (KUEHN, 2006). A simulação realística de robô (RRS) é uma interface padrão entre ferramenta de simulação de robô e o controlador do sistema do robô, possibilitando a utilização de programas *offline* no robô real sem nenhuma correção. Com a utilização do RRS podemos encontrar um mecanismo de sincronização que certifica todos os movimentos e tempos simulados pelo robô (CARLSSON et al., 2012).

Há também a possibilidade de simulação de recursos humanos, que permite o aperfeiçoamento da ergonomia do espaço de trabalho, templo de ciclo de montagens e comunicação dos resultados à planta, aumentando produtividade da produção, gerando uma documentação compreensível de operações humanas e promovendo a utilização das melhores práticas ergonômicas (KUEHN, 2006).

Figura 2 – Fábrica digital



Fonte: Autor

2.3. INTELIGÊNCIA ARTIFICIAL

Com a atual popularização da internet, uma existência universal de sensores, o surgimento do *Big Data*, desenvolvimento do e-commerce e a fusão dos conhecimentos com a sociedade física e cyber-física, o ambiente da informação para desenvolvimento de inteligência artificial se transformou profundamente, a guiando para uma nova fase evolucionária, a AI 2.0. Um sistema de manufatura inteligente pode ser caracterizado pela detecção inteligente autônoma, interconexões, colaboração, aprendizagem, análise, cognição, tomada de decisões, e a execução de materiais, máquinas, ambiente, informação e humanos em todo o sistema e ciclo de vida (LI et al., 2017).

A inteligência artificial será um importante recurso de *design* no futuro da automação, desempenhando um papel fundamental na redução dos esforços de programação e engenharia necessários para criar soluções de automação, tornando a lógica de controle mais ágil e os processos de produção mais flexíveis e precisos, sendo possível executar tarefas de automação que levam as soluções tradicionais a seus limites, se estabelecendo gradualmente em todos os níveis da automação totalmente integrada, do campo à nuvem. (SIEMENS, 2019).

Na área de inteligência artificial, o aprendizado de máquinas (*machine learning*) tem surgido como um método de escolha para o desenvolvimento prático de *softwares* para visão computacional, reconhecimento de fala, controle robótico e outras aplicações. Desenvolvedores reconhecem que, para muitas aplicações, pode ser mais fácil treinar sistemas demostrando-se exemplos de entradas e saídas desejadas do que programá-las manualmente antecipando-se a respostas para todas as entradas possíveis. (JORDAN; MITCHELL, 2015)

O aprendizado de máquinas encontra-se situado nas áreas de ciência computacional estatística, e várias outras disciplinas interessadas no aperfeiçoamento automático com o tempo e tomadas de decisões sobre incertezas. Os métodos mais utilizados em aprendizado de máquinas são os treinamentos supervisionados, onde os dados de treinamento são apresentados na forma de uma coleção de pares (x, y), e a saída produz um y* inferido a um x* questionado (JORDAN; MITCHELL, 2015).

Há várias formas de se mapear uma função em um treinamento supervisionado, como árvores de decisão, regressão logística, vetor de suporte, redes neurais, etc. Uma das áreas com maior impacto no treinamento supervisionado nos últimos anos envolve as redes profundas (*Deep Learning)*, que são redes multicamadas, onde, cada uma calcula um simples parâmetro de sua entrada. Os sistemas de *Deep Learning* fazem uso de algoritmos de otimização baseados em gradiente para se ajustar todos os seus parâmetros em função do erro em suas saídas (JORDAN; MITCHELL, 2015).

Tradicionalmente, técnicas de modelagens analíticas nem sempre são adequadas para se modelar sistemas de manufatura complexos, principalmente, devido ao número de variáveis de processo e a natureza não linear dos problemas. As redes neurais se mostraram uma boa alternativa para esses problemas, e uma série de implementações de sucesso vêm sendo reportadas. Normalmente, ao se utilizar essa abordagem, a rede neural é treinada, por meio das entradas e saídas do processo, para que por exemplo, possa reconhecer padrões específicos de um sistema e associá-los com uma determinada condição. A rede treinada é então um monitor das saídas do processo, do qual responderá com uma saída indicando um tipo de causa quando um padrão similar reaparecer no processo (SUKTHOMYA; TANNOCK, 2005).

Monsef e Areed (2006) fizeram um sistema CLP com redes neurais, onde o CLP media, operava e controlava a planta, e as redes neurais recebia todos os dados da operação para checar e supervisionar seu comportamento. O trabalho foi realizado visando regular os erros entre entradas e saídas utilizando redes neurais, que aprende o processo industrial o monitorando continuamente.

3. FUNDAMENTAÇÃO TEÓRICA

3.1. CONTROLADORES LÓGICOS PROGRAMÁVEIS

Os controladores lógicos programáveis (CLP) são os controladores industriais mais utilizados, especialmente por controles de supervisão, sequenciamento e lógicas de segurança. O desenvolvimento de novas estratégias de controle e expansão das funções de cálculo existentes são assuntos de interesses dos fabricantes de CLP. O sistema de uma CLP pode ser normalmente programado nas linguagens Ladder (LAD), diagrama de bloco de funções (FBD) e lista de declarações (STL) (ABDI; SALAMI; AHMADI, 2007).

O computador lógico programável emula o comportamento de um diagrama elétrico em escada, onde o CLP utiliza uma tabela de símbolos de entradas/saídas e um ciclo de varredura, como são máquinas sequenciais, para emular o funcionamento de circuitos paralelos que respondem instantaneamente. A varredura do programa resolve as lógicas booleanas relacionadas a informação de entrada com as tabelas de relés internos de saída, utilizadas para comandar dispositivos (KO; AHN; PARK, 2013).

O hardware da versão moderna do CLP é composto, basicamente, pelos mesmos componentes básicos, que detectam dados recebidos, processam e controlam várias saídas. Os blocos básicos são: Rack de montagem; Modulo de entrada e saída; Dispositivo de programação; Fonte de energia; Unidade de processamento central (CPU) (ALPHONSUS; ABDULLAH, 2016).

Figura 3 – Controlador lógico programável moderno

Fonte: SIEMENS, 2019

Um CLP industrial não manipula somente eventos discretos, ele também pode controlar sinais analógicos, controle de movimento, controle de posição e outras funções de tempos críticos. Uma característica importante do CLP é a função de controle reprogramável, que é descrita por seu código de controle (CARLSSON et al., 2012). O CLP, porém, não controla uma planta continuamente, ele trabalha com uma "foto" de um estado das entradas, e assim, atualiza as saídas de acordo com a lógica programada. Esse ciclo de escaneamento demanda certo tempo até que o CLP possa notar o valor de uma entrada da planta. Portanto, o CLP somente olha paras as entradas durante o ciclo de escaneamento (MONSEF; AREED, 2006).

Alphonsus e Abdullah (2016) citaram em seu trabalho as vantagens e desvantagens da utilização do CLP. Quando as vantagens, podemos citar:

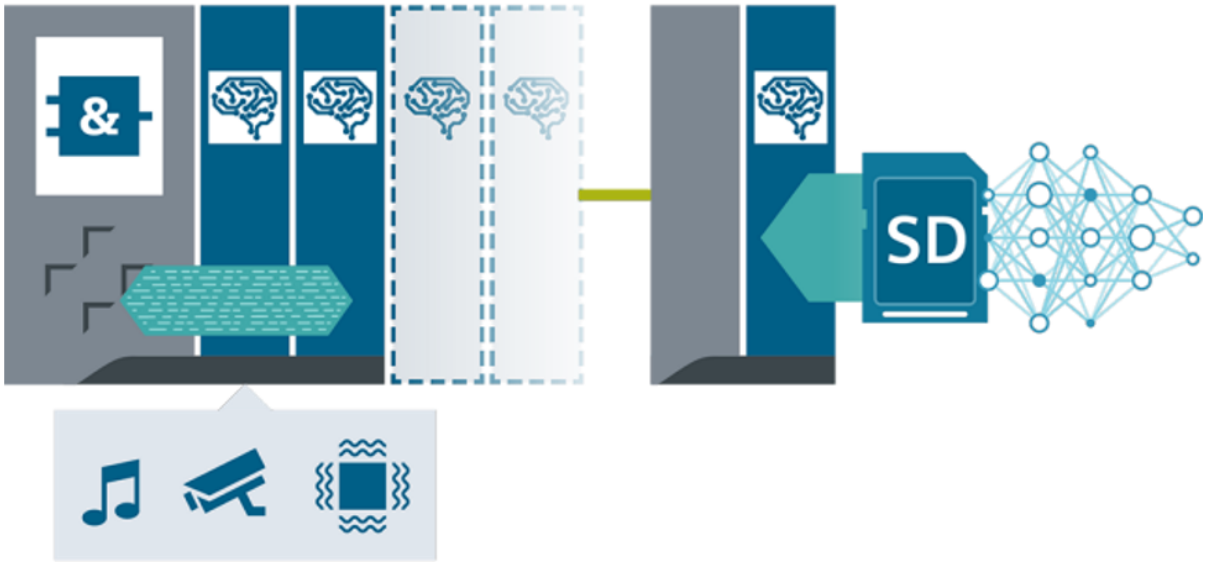
- É mais robusto e imune ao ruído.
- Abordagem modular na construção, sendo facilmente substituível ou modificado.
- Conexões de entradas e saídas padrões.
- Mais confiável comparado a relés, os dispositivos eletromecânicos sofrem desgaste físico toda vez que os dispositivos são ligados.
- Fácil de entender, programar, reprogramar e com programação off-line.
- Ampla base de aplicativos, suportando uma ampla gama de aplicações discretas e analógicas em vários setores.
- Custo e o tamanho dos CLPs diminuíram significativamente.

Já as limitações do CLP, principalmente comparado aos computadores convencionais (PC), podem ser listadas em:

- Os PCs melhoram em um ritmo muito rápido, são mais baratos e são mais potentes que um CLP.
- As memórias dos PCs excederam 1 Terabyte, enquanto os PLCs ainda estão dentro de Megabyte ou até mesmo de kilobyte.
- O PC está disponível em todo o mundo, a curto prazo, e de muitos fornecedores.
- O PC pode fornecer uma solução totalmente integrada que incorpora as funções do CLP, da Interface Homem-Máquina (IHM) e do terminal de programação.

No contexto da automação totalmente integrada (TIA), a *Siemens* avança na integração da inteligência artificial no ambiente de automação com o novo módulo NPU (unidade de processamento neural) para o controlador *SIMATIC*. O módulo realiza uma combinação simples de algoritmos de IA e lógica de CLP, permitindo que aplicativos possam ser implementados na máquina com base na inteligência artificial. O NPU facilita o manuseio de objetos desconhecidos, sem a necessidade de uma programação específica, podendo ser aplicados na detecção visual confiável de defeitos de qualidade, além de reconhecer padrões complexos nos dados e fazer com que um produto relevante seja rejeitado. O módulo opera usando um sistema neural treinado em um cartão SD (Figura 3), onde os usuários podem conectar sensores compatíveis, como câmeras e microfones, às interfaces integradas do módulo e ter os resultados do processamento avaliados no programa da CPU (SIEMENS, 2019).

Figura 4 – Unidade de processamento neural



Fonte: SIEMENS, 2019

3.2. COMISSIONAMENTO VIRTUAL

Para obterem sucesso em uma competição global, construtores estão sobre constante pressão para entregar produtos de baixo custo e alta qualidade com o menor tempo ao mercado. Não somente a necessidade de um sistema de produção eficiente e flexível, mas também o tempo de desenvolvimento desses sistemas devem ser considerados (DAHL et al., 2016), (KO; AHN; PARK, 2013).

Em um comissionamento tradicional, um programa de automação é testado nas máquinas reais, na linha de produção. O comissionamento virtual (VC) surgiu como uma resposta para uma necessidade de uma técnica que traga eficiência de tempo e custo ao processo de comissionamento (ARAS et al., 2018).

O comissionamento virtual para linhas de produção avançou como uma inovação trazida pela quarta revolução industrial. Nas abordagens tradicionais, o programa somente era testado nos componentes da fábrica, e só então, os erros de projetos e de automação eram detectados e corrigidos, período onde se utilizava cerca de 25% de todo o tempo da fase de construção (ARAS et al., 2018). Com a aplicação do comissionamento virtual, é esperado uma redução entre 10 e 30% do tempo de comissionamento do sistema, além de aumentar a qualidade das soluções em engenharia (DRATH; WEBER; MAUSER, 2008). Uma das maiores motivações de se utilizar o comissionamento virtual é em se reduzir o tempo de teste e integração durante a fase de desenvolvimento, antes do sistema físico de produção estar completamente instalado (DAHL et al., 2016).

O comissionamento virtual de um sistema de produção pode ser considerado como uma simulação envolvendo um CLP real interagindo com uma planta virtual. Enquanto nos métodos convencionais as fases de desenvolvimentos mecânicos e elétricos ocorrem sequencialmente, a abordagem com VC pode executá-las simultaneamente (KO; AHN; PARK, 2013).

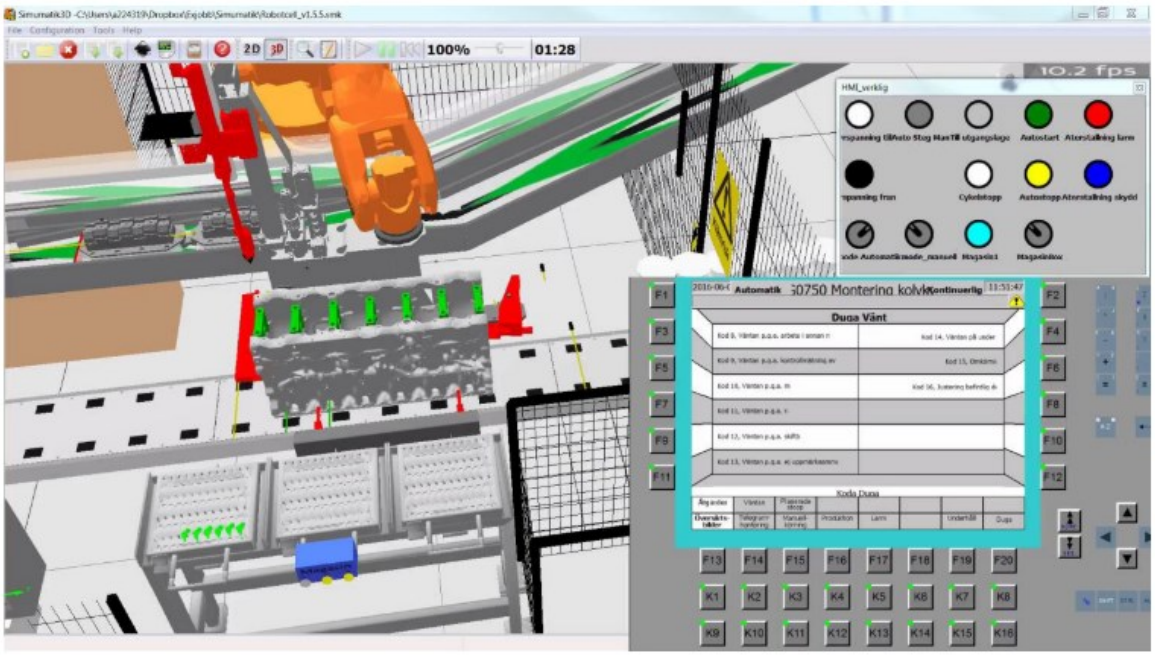
No conceito de comissionamento virtual, o processo é executado em um ambiente computacional conectado a um CLP, com o propósito de detectar e corrigir os erros ainda durante a fase de desenvolvimento da automação. Para isso, o modelo preparado no software de simulação deve conter as mesmas configurações de entradas e saídas, e comportamento lógico do sistema físico pretendido. Portanto, os modelos de comissionamento virtual contêm tanto o comportamento físico, quanto o lógico do sistema em desenvolvimento (DAHL et al., 2016), (ARAS et al., 2018).

O controle do sistema é implementado na ferramenta de programação original do CLP e do controlador do robô, o que reflete a realidade dos projetos de engenharia. O foco do modelo de simulação é modelar a planta física em respeito ao seu comportamento mecatrônico, que pode ser abstraído em seus atuadores e sensores (DRATH; WEBER; MAUSER, 2008).

Para um comissionamento virtual de um CLP, a planta virtual deve interagir com os seus I/Os. O comportamento do modelo da planta deve ser o mesmo que o sistema de produção real e seus dispositivos, incluindo robôs, transportadores, solenoides e sensores. Pode-se considerar como o modelo da planta sendo um conjunto de modelos de dispositivos (KO; AHN; PARK, 2013).

A execução somente em ambiente de software requer a disponibilidade de softwares controladores e um controlador virtual de robô. A comunicação entre o CLP e o software de simulação é feita normalmente via padrão de comunicação OPC, onde é possível compartilhar dados entre as famílias de controladores e outros componentes de acordo com a demanda do projeto (DRATH; WEBER; MAUSER, 2008). Desenvolvido pela OPC Foundation, o OPC consiste em uma série de especificações que contém informações de como servidor e cliente devem trocas dados, definindo a interface entre CLP e aplicação do computador. A implementação de um cliente OPC permite ao CLP controlar o modelo da simulação (CARLSSON et al., 2012).

Figura 5 – Comissionamento virtual de uma célula

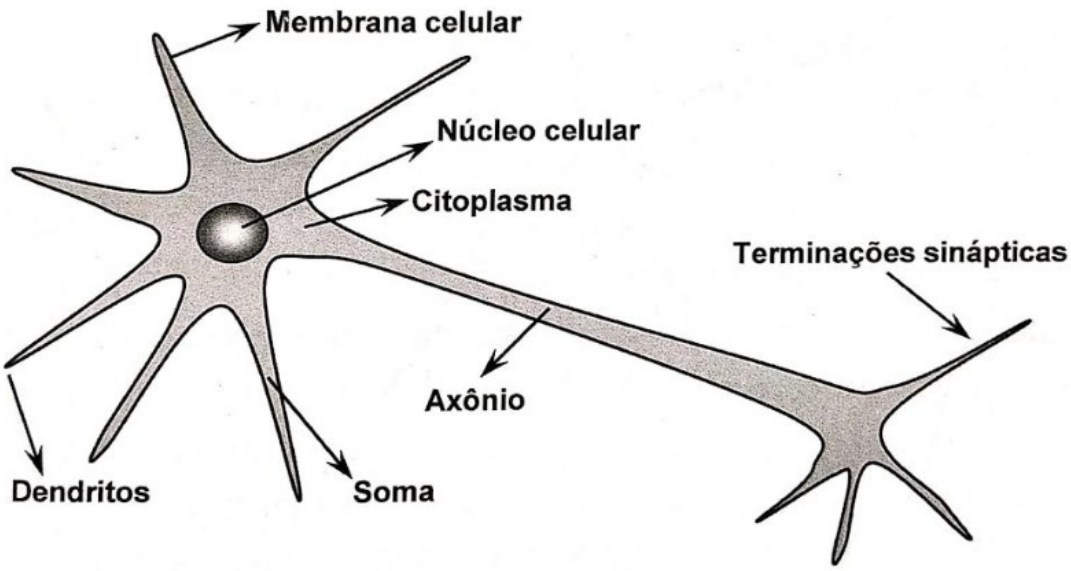


Fonte: BINNBERG et al., 2016

3.3. REDES NEURAIS ARTIFICIAIS

As redes neurais artificiais são uma técnica de processamento de informação de inteligência artificial, inspirado no modelo do cérebro biológico, que contém numerosos neurônios interconectados. sua estrutura busca representar de forma matemática o modelo de cognição humana, como a capacidade de aprender e armazenar informações a partir de experiências anteriores. Cada processador de uma rede neural recebe informação de um nível superior e transfere a saída para um nível inferior. Tal como um neurônio biológico, o buffer de entrada de uma rede neural artificial funciona como os dendritos, recebendo informações de outros neurônios, os neurônios da camada oculta são como o corpo celular, processando informação para outros neurônios, e a camada de saída como o axônio, transmitindo a informação processada (SILVA; SPATTI; FLAUZINO, 2016, p. 29).

Figura 6 – Neurônio biológico



Fonte: SILVA; SPATTI; FLAUZINO, 2016, p. 29

Em um neurônio artificial, as informações de entrada excitam o canal de entrada de um modelo matemático, assim como um sinal elétrico, de um neurônio real. Essas informações são multiplicadas pelos pesos sinápticos, e após somadas, excitam a saída do neurônio (ABDI; SALAMI; AHMADI, 2007). Um neurônio artificial pode ser descrito a partir dos seguintes elementos básicos:

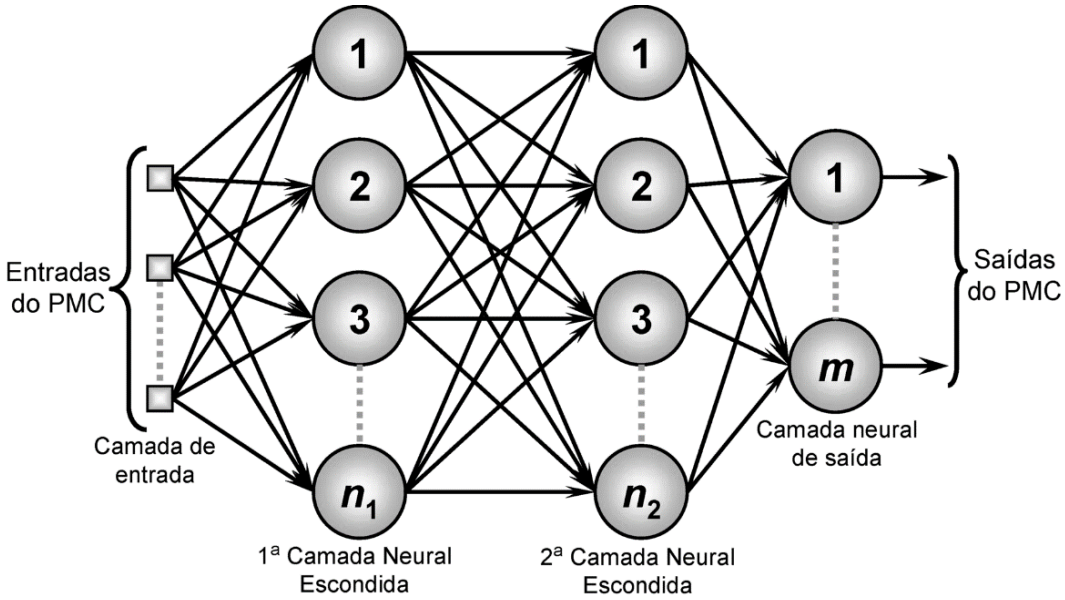
- Sinais de entrada $\{X1, X2, \ldots, Xn\}$
- Pesos sinápticos $\{W1, W2, \ldots, Wn\}$
- Combinador linear $\{\sum\}$
- Limiar de ativação $\{\theta\}$
- Potencial de ativação {u}
- Função de ativação {g}
- Sinal de saída {y}

Portanto, de seus elementos básicos, podemos definir as expressões de um neurônio artificial em (SILVA; SPATTI; FLAUZINO, 2016, p. 35):

$$(1) \quad u = \sum_{i=1}^{n} Wi \cdot Xi - \theta \qquad\qquad (2) \quad y = g(u)$$

Basicamente, as redes neurais artificiais podem ser dividias em três partes: Camada de entrada, Camadas escondidas e Camada de saída. Uma arquitetura *feedforward* indica que as informações processadas seguem somente uma direção, da camada de entrada para a camada de saída. As redes *perceptron* multicamadas são redes *feedforward* com uma série de aplicações em problemas complexos e não lineares. Essas redes contam com as camadas de entradas e saídas como uma parte da estrutura, e entre elas, são utilizadas camadas escondidas de neurônios (ABDI; SALAMI; AHMADI, 2007).

Figura 7 – Rede *Perceptron* multicamadas



Fonte: SILVA; SPATTI; FLAUZINO, 2016, p. 92

Sistemas industriais complexos são considerados aqueles com múltiplos parâmetros de processo e, ou, condições do ambiente que alterem o comportamento do processo. Podem haver também uma série de saídas no sistema, o que demanda um completo entendimento teórico das operações do processo (SUKTHOMYA; TANNOCK, 2005). As redes neurais podem ser utilizadas em controle industrial tanto para apoiar o controlador quanto para serem o controlador do processo. As redes neurais conseguem simplificar as leis de controles, classificando certos parâmetros do controlador, resolvendo uma lei de controle implícita, executando diagnósticos e recomendando ações de supervisão para o controlador (MONSEF; AREED, 2006).

3.4. TREINAMENTO NEURAL

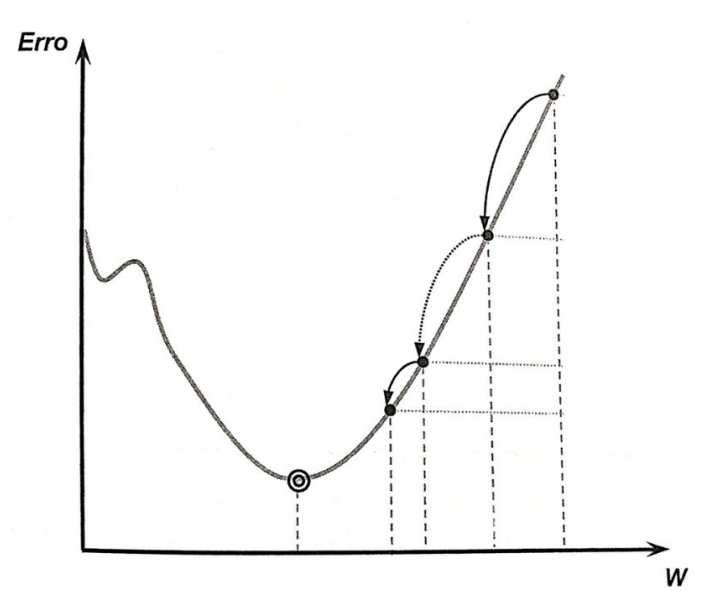
Um dos aspectos mais importantes das redes neurais artificiais é a capacidade de aprender a partir de uma certa amostra, do qual, represente a relação de entradas e saídas de um sistema. Após aprendido está relação, a rede é capaz de generalizar e produzir uma saída a partir de qualquer entrada inserida. Esse processo de treinamento pode ser realizado, basicamente, de forma supervisionada ou não-supervisionada (SILVA; SPATTI; FLAUZINO, 2016, p. 51). O aprendizado supervisionado é realizado por meio de pares de entradas e saídas. Com isso, um conjunto de valores reais são aplicados à entrada da rede e sua saída é comparada com o valor esperado. Já no aprendizado não-supervisionado, temos durante o treinamento da rede apenas os valores das variáveis de entrada disponíveis, portanto, a rede é responsável por identificar padrões e características a partir dos dados de treinamento, realizando o agrupamento de informações (clusterização).

O treinamento é o processo de determinar os pesos ideais e os pontos de polarização da rede neural artificial. Isso é feito definindo a função de erro total entre a saída da rede e o destino desejado e, em seguida, minimizando-a em relação aos pesos. O processo de treinamento de uma rede *Perceptron* multicamadas é dado por meio de uma técnica supervisionada chamada de *Backpropagation*, do qual o nome refere-se à propagação retroativa do erro durante o treinamento da rede, de tal forma que os algoritmos dessa família usam a regra da cadeia várias vezes para o cálculo da derivada parcial da função de erro total da rede (BAGHIRLI, 2015). Esse processo pode ser descrito nos passos abaixo:

- Na primeira fase, chamada de *forward*, os sinais de uma amostra do conjunto de treinamento são inseridos nas entradas da rede e são propagadas camada por camada até sua saída.
- As respostas produzidas pelas saídas, para um determinado conjunto de entradas, são comparadas com as respostas desejadas dos dados de treinamento.
- Os erros provenientes das diferenças entre as repostas desejadas e as saídas obtidas são calculados para serem utilizados nos ajustes dos parâmetros da rede.
- Na segunda fase do método *Backpropagation*, chamada de *backward*, os ajustes dos pesos sinápticos e limiares de todos os neurônios da rede são executados.

Portanto, as sucessivas aplicações dessas fazes fazem com que os pesos sinápticos e limiares dos neurônios se ajustem automaticamente em cada interação, diminuído gradativamente os erros entre saídas obtidas e saídas desejadas (SILVA; SPATTI; FLAUZINO, 2016, p. 94).

Figura 8 – Diminuição do erro durante o treinamento



Fonte: Adaptado de SILVA; SPATTI; FLAUZINO, 2016, p. 113

O algoritmo básico de *Backpropagation* ajusta os pesos na direção de descida mais íngreme, ou seja, a mais negativa do gradiente (figura 8). Esta é a direção na qual a função de desempenho está diminuindo mais rapidamente. Acontece que, embora a função diminua mais rapidamente ao longo do negativo do gradiente, isso não produz necessariamente a convergência mais rápida.

Nos algoritmos de gradiente conjugado, uma busca é realizada ao longo de uma direção que produz uma convergência geralmente mais rápida do que a direção de descida mais íngreme, preservando a minimização de erros alcançada em todas as etapas anteriores. Essa direção é chamada de direção conjugada. Na maioria dos algoritmos, o tamanho do passo é ajustado em cada iteração. Uma pesquisa é feita ao longo da direção do gradiente conjugado para determinar o tamanho do passo, o que minimizará a função de desempenho ao longo dessa linha (BAGHIRLI, 2015).

Os Métodos de Gradiente Conjugado são baseados em estratégia geral de otimização, mas escolhe a direção da pesquisa e o tamanho da etapa com mais cuidado usando informações da aproximação de segunda ordem:

$$(3) \quad E(w+y) \approx E(w) + E'(w)^T y + ½ y^T E''(w) y$$

Funções quadráticas têm algumas boas propriedades que funções gerais não necessariamente possuem. Denunciam a aproximação quadrática de E em uma vizinhança de um ponto w por $Eqw(y)$, de modo que $Eqw(y)$ seja dado por:

$$(4) \quad Eqw(y) = E(w) + E'(w)^T y + ½ y^T E''(w) y$$

Para determinar os mínimos para $Eqw(y)$, os pontos críticos para $Eqw(y)$ devem ser encontrados, ou seja, os pontos em que:

$$(5) \quad Eqw(y) = E''(w)\, y + E'(w) = 0$$

Considerando $p1, \dots, pN$ um sistema conjugado e y1 é um ponto no espaço de peso. Seja os pontos $y2, \dots, yN+1$ definidos recursivamente por $(6) \quad yk+1 = yk + \alpha kpk$, então $yk+1$ minimiza Eqw restrito ao plano $k\ \pi k$ dado por y1 e $p1, \dots, pk$.

O algoritmo, portanto, é dado pelos seguintes passos: Selecione um vetor de peso inicial $y1$ e um sistema conjugado $p1, \dots, pN$. Encontre mínimos sucessivos para Eqw nos planos $\pi 1, \dots, \pi N$, onde $\pi k, 1 \leq k \leq N$, é dado por:

$$(7) \quad y = y1 + \alpha 1p1 + \cdots + \alpha kpk, \alpha i \in \Re.$$

O algoritmo garante que o mínimo global para uma função quadrática seja detectado na maioria das N iterações. Se todos os valores de $E''(w)$ se dividirem em vários grupos com valores do mesmo tamanho, existe uma grande possibilidade de que o algoritmo termine em muito menos que N iterações.

Não é necessário conhecer os vetores de peso conjugados $p1, \dots, pN$ antecipadamente, eles podem ser determinados recursivamente. Inicialmente, $p1$ é definido como o vetor de descida mais íngreme $-Eqw(y1)$. $pk+1$ é então determinado recursivamente como uma

combinação linear da descida mais íngreme atual do vetor $-Eqw\ (yk+1)$ e a direção anterior pk.

Para cada iteração, o algoritmo descrito acima é aplicado à aproximação quadrática Eqw da função de erro global E no ponto atual w no espaço de peso. Como a função de erro $E(w)$ é não quadrática, o algoritmo não converge necessariamente em N etapas. Se o algoritmo não convergir após N etapas, o algoritmo será reiniciado, ou seja, inicializando $pk+1$ na direção de descida mais íngreme atual $rk+1$ (MØLLER, 1990).

A capacidade da rede neural artificial representar o sistema dependerá do nível de flexibilidade empregado na construção da rede, de tal forma que ela não seja superdimensionada a ponto de resultar em uma memorização excessiva, o *overfitting*, que decora as respostas para determinadas entradas, tornando o erro baixo para estas e alto para entradas que não foram utilizadas durante o treinamento. Quando ocorre uma situação de subdimensionamento e a rede não foi capaz de extrair um resultado que forneça um baixo erro e boa generalização, podemos dizer que há o *underfitting* (SILVA; SPATTI; FLAUZINO, 2016, p. 153).

3.5. COMUNICAÇÃO OPC

Desenvolvido pela *OPC Foundation*, o OPC (*Open Platform Communications*) é o padrão de interoperabilidade para a troca segura e confiável de dados. Ele é independente de plataforma e garante o fluxo contínuo de informações entre dispositivos de vários fornecedores. O padrão OPC é uma série de especificações desenvolvidas por fornecedores do setor industrial, usuários finais e desenvolvedores de software. Essas especificações definem a interface entre Clientes e Servidores (OPC FOUNDATION, 2019).

As especificações clássicas do OPC são baseadas na tecnologia *Microsoft Windows* para a troca de dados entre componentes de *software*. As especificações fornecem definições separadas para acessar dados de processo, alarmes e dados históricos. Dentre os tipos de comunicação disponíveis, temos:

- Acesso a Dados OPC (OPC DA): A especificação OPC DA define a troca de dados, incluindo valores, tempo e informações de qualidade.
- Alarmes e Eventos OPC (OPC AE): A especificação OPC A & E define a troca de informações de mensagens de alarme e tipo de evento, bem como estados de variáveis e gerenciamento de estado.

- Acesso a Dados Históricos OPC (OPC HDA): A especificação OPC HDA define métodos de consulta e análise que podem ser aplicados a dados históricos, com registro de data e hora.

As especificações clássicas do OPC serviram bem à comunidade, no entanto, com a introdução de arquiteturas orientadas a serviços em sistemas de manufatura, surgiram novos desafios em segurança e modelagem de dados. Logo, em 2008, a Fundação OPC lançou a Arquitetura Unificada OPC (OPC UA), uma arquitetura aberta orientada a serviços independente de plataforma que integra todas as funcionalidades das especificações clássicas existentes do OPC (OPC FOUNDATION, 2019).

Essa abordagem em várias camadas atinge os objetivos de especificação de design original de:

- Equivalência funcional: todas as especificações do COM OPC Classic são mapeadas para UA.
- Independência da plataforma: de um microcontrolador incorporado à infraestrutura baseada na nuvem.
- Seguro: criptografia, autenticação e auditoria.
- Extensível: capacidade de adicionar novos recursos sem afetar os aplicativos existentes.
- Modelagem abrangente de informações: para definir informações complexas.

Com base no sucesso do OPC Classic, o OPC UA foi projetado para aprimorar e superar os recursos das especificações do OPC Classic, fato esse que confirma o OPC-UA como padrão na camada de comunicação nas plataformas de indústria 4.0. (OPC FOUNDATION, 2019).

3.6. LÓGICA BOOLEANA

Na álgebra booleana, uma variável só pode ter dois valores: 1 ou 0, verdadeiro ou falso, alto ou baixo etc. Uma função booleana consiste em uma ou mais operações booleanas que resultam e um único valor booleano (saída) a partir de um conjunto de outros valores booleanos (entradas). Essas entradas e saídas podem ser apenas 1 ou 0 (CREATIONS, 2017).

A operação OR é a primeira das três operações booleanas básicas. A expressão $X = A + B$ significa que X será 1 quando A ou B for 1. Em circuitos digitais, uma porta OR tem

duas ou mais entradas cuja saída é igual a combinação das entradas por meio da operação *OR* (TOCCI; WIDMER; MOSS, 2011, p. 51).

A operação *AND* é a segunda operação básica. A expressão $X = A \cdot B$ significa que X será 1 somente quando A e B forem, ambas, nível 1. A porta lógica *AND* tem sua saída igual seu produto lógico *AND* das entradas, portanto, a porta *AND* é um circuito que opera de modo que sua saída seja nível alto somente quando toas as entradas também o forem. Para todos os outros casos, sua saída é nível baixo (TOCCI; WIDMER; MOSS, 2011, p. 55).

A operação *NOT*, também denominada inversão ou complemento, é diferente das operações *OR* e *AND* por ser realizada em uma única variável de entrada, podendo ser expresso por $X = \bar{A}$. O circuito *NOT* apresenta sempre uma entrada, e seu nível lógico de saída é o oposto do nível lógico de entrada, portanto, tendo entrada igual a 0, temos a saída igual a 1, e vice-versa (TOCCI; WIDMER; MOSS, 2011, p. 57).

A tabela da verdade da figura 9 resume os valores resultantes das operações lógicas básicas para uma entrada X e Y:

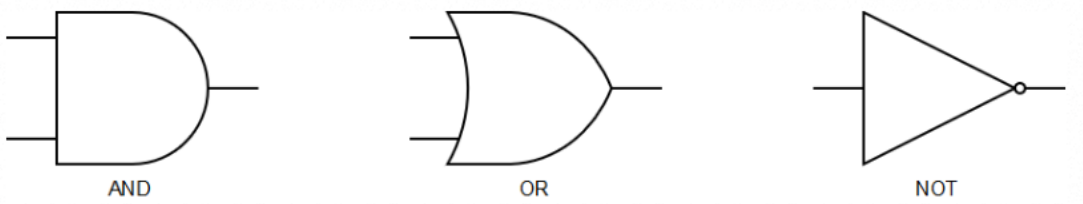
Figura 9 – Tabela da verdade

x	y	x AND y	x OR y	NOT x	NOT y
0	**0**	0	0	1	1
0	**1**	0	1	1	0
1	**0**	0	1	0	1
1	**1**	1	1	0	0

Fonte: CREATIONS, 2017

Em direção à aplicação prática (hardware) da lógica booleana, se é comummente visto diagramas lógicos-eletrônicos que se utilizem das portas lógicas como forma de representação. Portanto, as operações booleanas básicas podem ser representadas em forma de diagrama com os símbolos representados na figura 10:

Figura 10 – Portas lógicas



Fonte: CREATIONS, 2017

4. MODELAGEM

Os procedimentos para a implementação do controle por aprendizado de máquina proposto neste projeto podem ser conferidos nos tópicos a seguir. A modelagem utilizada na pesquisa será abordada nos itens que devem ser seguidos sequencialmente, organizados em simulação, integração, controle e redes neurais.

4.1. SIMULAÇÃO

4.1.1. Software de simulação

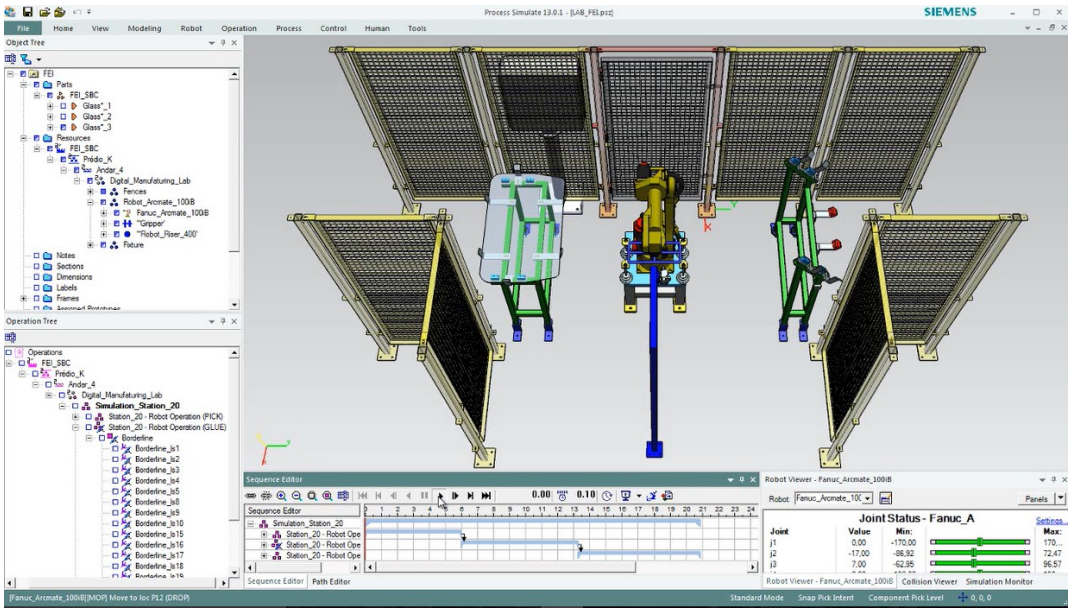
Para a elaboração do modelo de simulação da célula robotizada, foi utilizado o *software Process Simulate*, que faz parte do pacote *Tecnomatix* da empresa *Siemens,* consistindo em uma série de soluções de manufatura digital, engenharia de produto, engenharia de produção e operações, focando em maximizar a eficiência desses processos.

O *software Process Simulate* é uma solução de manufatura digital para verificação do processo de fabricação em um ambiente 3D, sendo um importante facilitador em velocidade e eficiência, permitindo que as organizações validem virtualmente os principais conceitos de uma manufatura. O *software Process Simulate* permite a verificação de diferentes segmentos do processo de fabricação, como montagens, operações humanas, soldagem, processos contínuos como solda a laser além de outros processos robóticos que podem ser simulados em um mesmo ambiente, emulando o comportamento humano realista, controladores robóticos e lógica do CLP (SIEMENS PLM, 2019).

O *Process Simulate Commissioning* permite que os usuários simplifiquem os dados existentes de fabricação e engenharia desde o projeto conceitual até o chão de fábrica, fornecendo uma plataforma de integração comum para as diversas disciplinas que participam de um comissionamento real de uma célula de produção. Com o comissionamento virtual, os usuários podem simular o código real do CLP com o hardware que será utilizado por meio de uma comunicação OPC, além da utilização dos programas reais do robô, permitindo assim um ambiente de comissionamento virtual mais realista (SIEMENS PLM, 2019).

A figura 11 ilustra a célula robotizada que foi utilizada nesta pesquisa construída no ambiente virtual do *software Process Simulate.*

Figura 11 – Visão do software de simulação



Fonte: Autor

4.1.2. A célula simulada

Para essa pesquisa, o laboratório de manufatura digital do centro universitário FEI foi utilizado, a partir das ferramentas de manufatura digital, efetuando a simulação e realizando o comissionamento virtual da célula robotizada. A célula simulada possui os seguintes recursos de *hardware*:

1. Um robô *Fanuc Arcmate 100iB*
2. Um dispositivo de abastecimento
3. Um dispositivo de retirada
4. Um pedestal de aplicação estacionário
5. Uma garra de ventosa à vácuo
6. Fechamento da célula com porta, botoeiras, barreiras de luz e indicadores luminosos

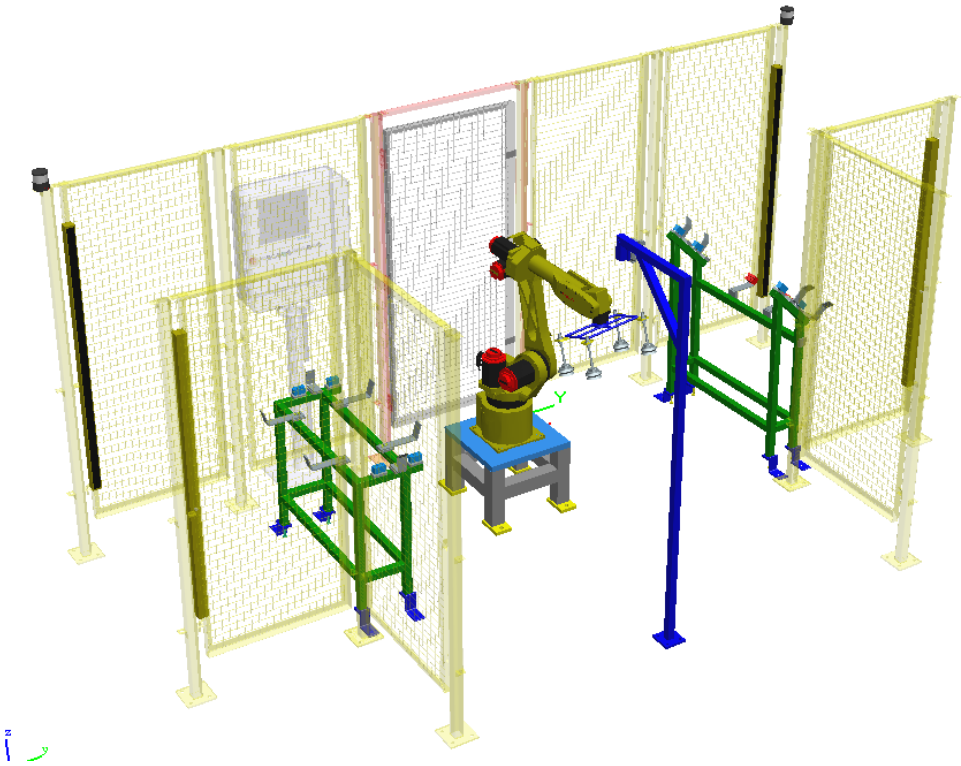
A figura 12 apresenta uma imagem do laboratório de manufatura digital estudado nessa pesquisa.

Figura 12 – Imagem da célula robótica estudada

Fonte: Autor

Para que haja um comissionamento virtual, é necessária a criação do modelo idêntico ao real nos aspectos de recursos, operações e comportamento no software de simulação, modelo esse que pode ser conferido na Figura 13.

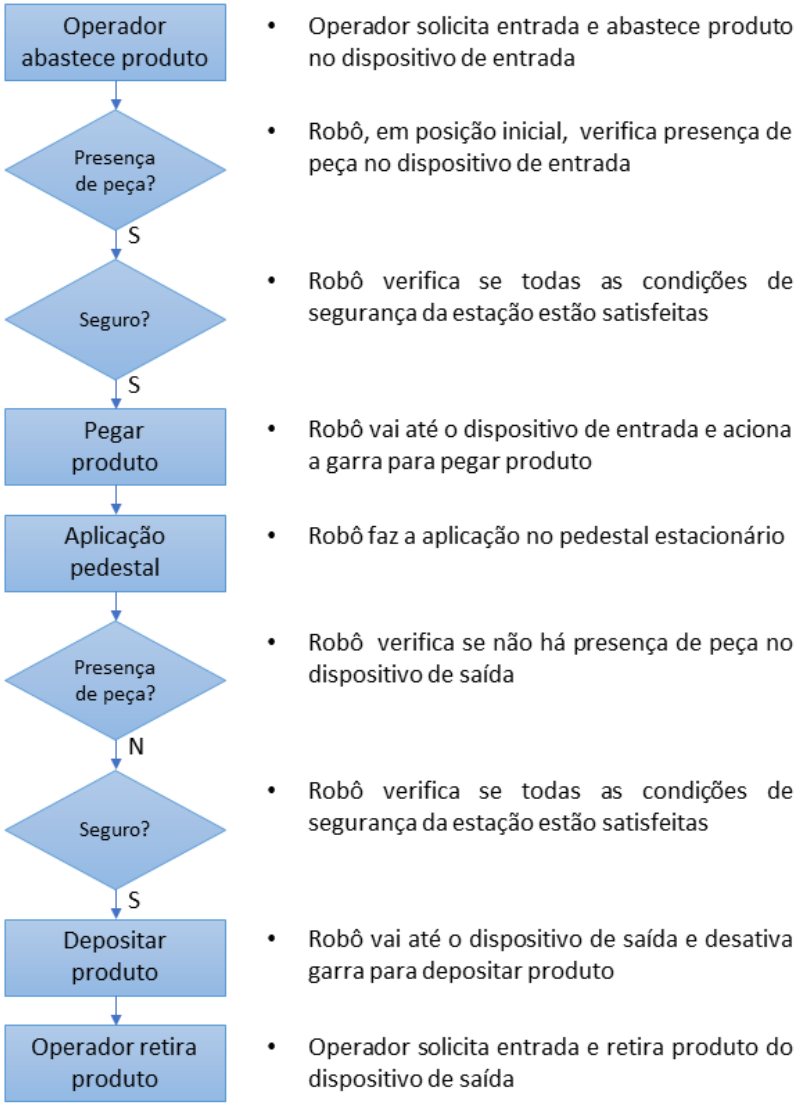
Figura 13 – Modelo virtual da célula



Fonte: Autor

A célula estudada apresenta um fluxo de operação simples, onde há a utilização do operador para abastecer a linha, aguardar a operação automática realizada pelo robô, e retirar a peça pronta da linha. As sequências de operações e condições para seu correto funcionamento podem ser observadas no fluxograma da Figura 14.

Figura 14 – Fluxograma de processo



Fonte: Autor

4.1.3. Sinais

Para uma correta aplicação de um comissionamento virtual, é necessário que o modelo simulado contenha todos os sinais utilizados pelo processo, e que seu comportamento corresponda com exatidão à realidade, garantindo assim, seu correto funcionamento lógico e operacional. Portanto, foi necessário fazer um levantamento de todos os sinais existentes na célula, sinais estes que se encontram listados na tabela 1.

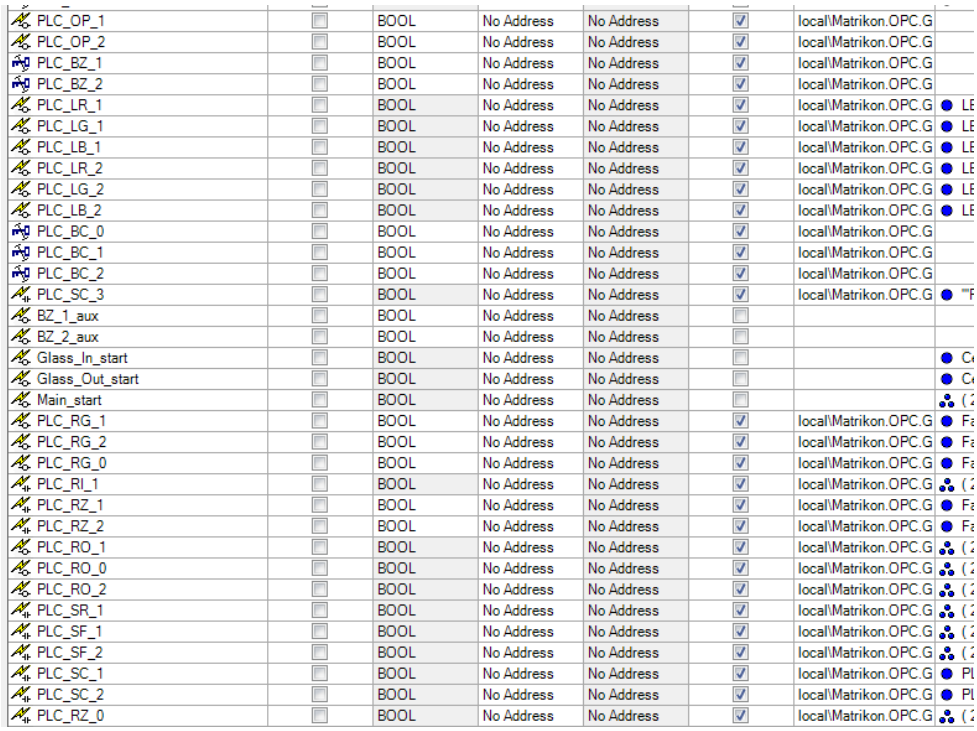
Tabela 1 – Lista de sinais

Entradas	Descrição	Saídas	Descrição
PLC_SR_1	Sensor de peça da garra	PLC_OP_1	Operador na zona 1 (Entrada)
PLC_SF_1	Sensor de peça dispositivo de entrada	PLC_OP_2	Operador na zona 1 (Saída)
PLC_SF_2	Sensor de peça dispositivo de saída	PLC_RG_0	Enviar robô para Home
PLC_RZ_0	Robô na posição de Home	PLC_RG_1	Liberação do robô para zona 1
PLC_RZ_1	Robô na zona 1 (Entrada)	PLC_RG_2	Liberação do robô para zona 2
PLC_RZ_2	Robô na zona 2 (Saída)	PLC_LR_1	Sinal luminoso zona 1 - Vermelho
PLC_RI_1	Sinal de vácuo do robô	PLC_LG_1	Sinal luminoso zona 1 - Verde
PLC_BZ_1	Botão de solicitação de entrada zona 1	PLC_LB_1	Sinal luminoso zona 1 - Azul
PLC_BZ_2	Botão de solicitação de entrada zona 2	PLC_LR_2	Sinal luminoso zona 2 - Vermelho
PLC_BC_0	Botão de emergência	PLC_LG_2	Sinal luminoso zona 2 - Verde
PLC_BC_1	Botão de início de processo	PLC_LB_2	Sinal luminoso zona 2 - Azul
PLC_BC_2	Botão de fim de processo	PLC_RO_0	Sinal de emergência
PLC_SC_1	Sinal de segurança barreira de luz 1	PLC_RO_1	Sinal de início para o robô
PLC_SC_2	Sinal de segurança barreira de luz 2	PLC_RO_2	Sinal de fim para o robô
PLC_SC_3	Sinal de segurança da porta	PLC_VO_1	Sinal de acionamento de vácuo

Fonte: Autor

Tendo em mãos os sinais necessários para o funcionamento da célula, estes devem ser devidamente criados no software de simulação, atentando-se à classificação correta de entradas e saídas, e habilitando a conexão externa para o servidor utilizado, conforme a figura 15:

Figura 15 – Sinais criados no modelo de simulação



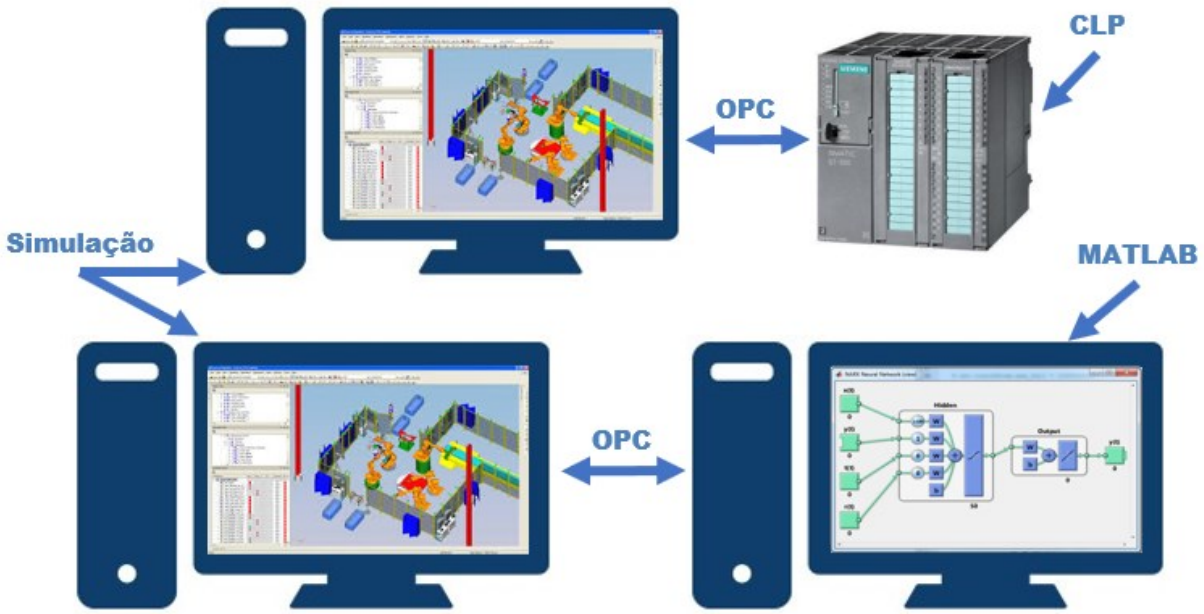
PLC_OP_1		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	
PLC_OP_2		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	
PLC_BZ_1		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	
PLC_BZ_2		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	
PLC_LR_1		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	LE
PLC_LG_1		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	LE
PLC_LB_1		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	LE
PLC_LR_2		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	LE
PLC_LG_2		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	LE
PLC_LB_2		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	LE
PLC_BC_0		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	
PLC_BC_1		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	
PLC_BC_2		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	
PLC_SC_3		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	"F
BZ_1_aux		BOOL	No Address	No Address			
BZ_2_aux		BOOL	No Address	No Address			
Glass_In_start		BOOL	No Address	No Address			C
Glass_Out_start		BOOL	No Address	No Address			C
Main_start		BOOL	No Address	No Address			(
PLC_RG_1		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	Fa
PLC_RG_2		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	Fa
PLC_RG_0		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	Fa
PLC_RI_1		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	(
PLC_RZ_1		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	Fa
PLC_RZ_2		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	Fa
PLC_RO_1		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	(
PLC_RO_0		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	(
PLC_RO_2		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	(
PLC_SR_1		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	(
PLC_SF_1		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	(
PLC_SF_2		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	(
PLC_SC_1		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	Pl
PLC_SC_2		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	Pl
PLC_RZ_0		BOOL	No Address	No Address	☑	local\Matrikon.OPC.G	(

Fonte: Autor

4.2. INTEGRAÇÃO

Para possibilitar a integração entre o software de simulação (*Process Simulate*) e o software de controle (*Matlab)*, esta pesquisa utiliza um protocolo de comunicação OPC. Com a figura 16 é possível verificar que, em um comissionamento virtual convencional, o *software Process Simulate* se comunica a um CLP por meio de comunicação OPC, e assim, há a certificação da lógica programada na simulação. Essa pesquisa traz a substituição desse CLP pelo *software Matlab* que, utilizando a integração com protocolo OPC, também é capaz da certificação da lógica na simulação, porém nesse caso, com o recurso matemático de uma rede neural artificial.

Figura 16 – Esquema de comunicação



Fonte: Autor

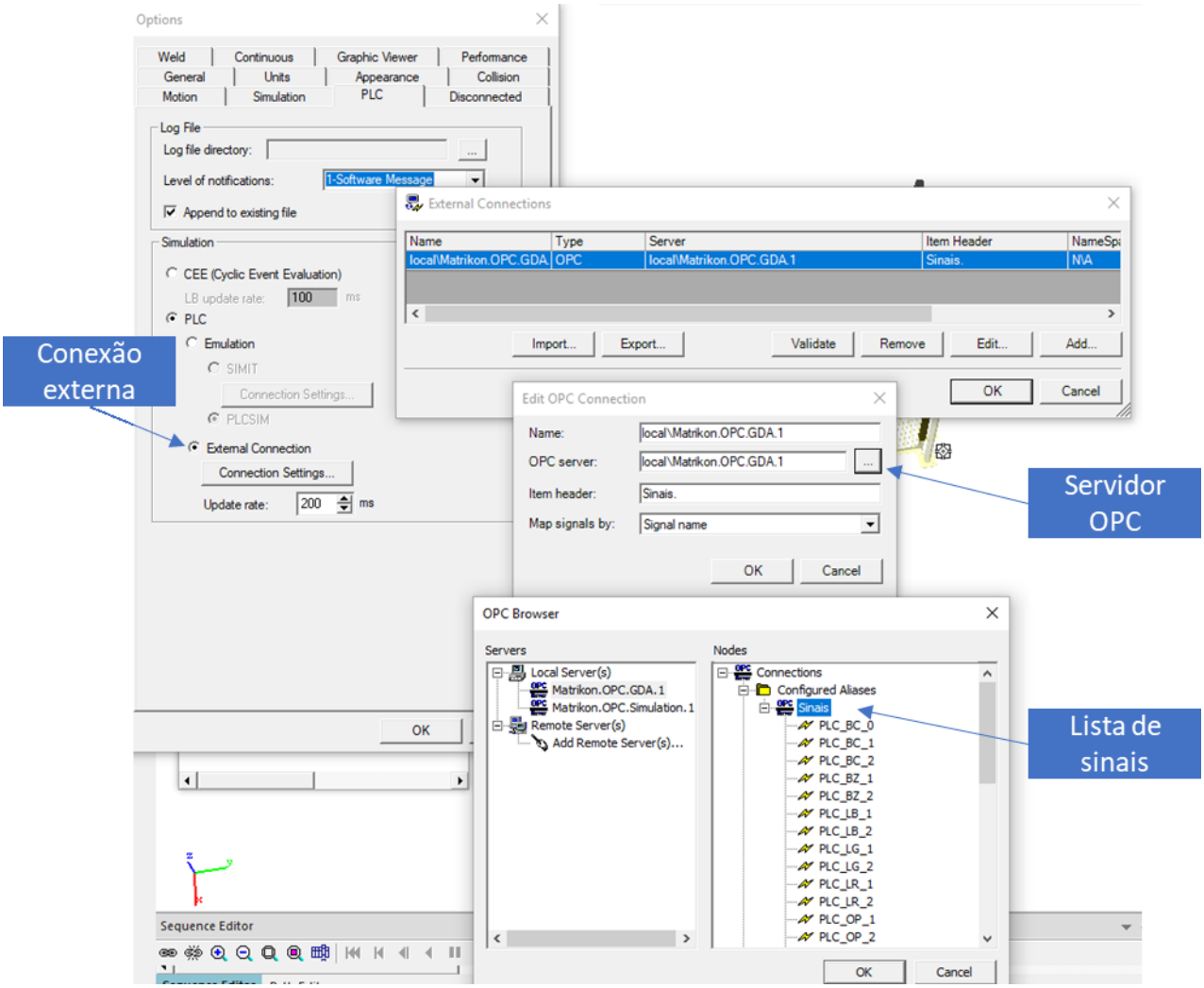
Para estabelecer uma rede OPC, é necessária a utilização de um software que trabalha como servidor, provendo assim, o compartilhamento dos sinais nele criado com os softwares integrados. Existem vários servidores OPC disponíveis, e para essa pesquisa, foi utilizado o *software Matrikon OPC Server GDA*, que apresenta uma interface de fácil utilização e de completa integração em aplicações para propósitos genéricos, como essa.

É utilizado a comunicação OPC de acesso de dados genérica (GDA), pois, devido à baixa complexidade dos dados utilizados nessa pesquisa e a fácil integração entre os softwares,

não é necessário uma plataforma com histórico de dados (OPC-HDA) ou de arquitetura unificada (OPC-UA), embora também seja possível serem utilizadas para esse propósito.

Após configurado o servidor e criados os sinais de processo, é necessário acessar os *softwares Process Simulate* e *Matlab* e realizar o apontamento do servidor, no qual o *software* recebe e atualiza as variáveis. Por meio das configurações de conexões externas, deve-se selecionar na lista de servidores disponíveis o que foi criado para a aplicação, e por meio dos nomes dos sinais criados, os *softwares Process Simulate* e *Matlab* conseguem trocar dados com o servidor, conforme ilustra a figura 17.

Figura 17 – Apontamento para servidor OPC



Fonte: Autor

4.3. CONTROLE

4.3.1. Software de controle

Após a elaboração do modelo de simulação, é necessário a modelagem do sistema que irá controlar a célula em substituição aos controladores lógicos programáveis convencionais. Para esse propósito, essa pesquisa utiliza o software *Matlab* como interface de controle, tendo assim, recursos para elaborar a lógica de funcionamento de processo, extrair os dados de treinamento, treinar uma rede neural artificial e controlar o modelo de simulação com o resultante do treinamento, comunicando-se com o software de simulação por meio da interface OPC disponível.

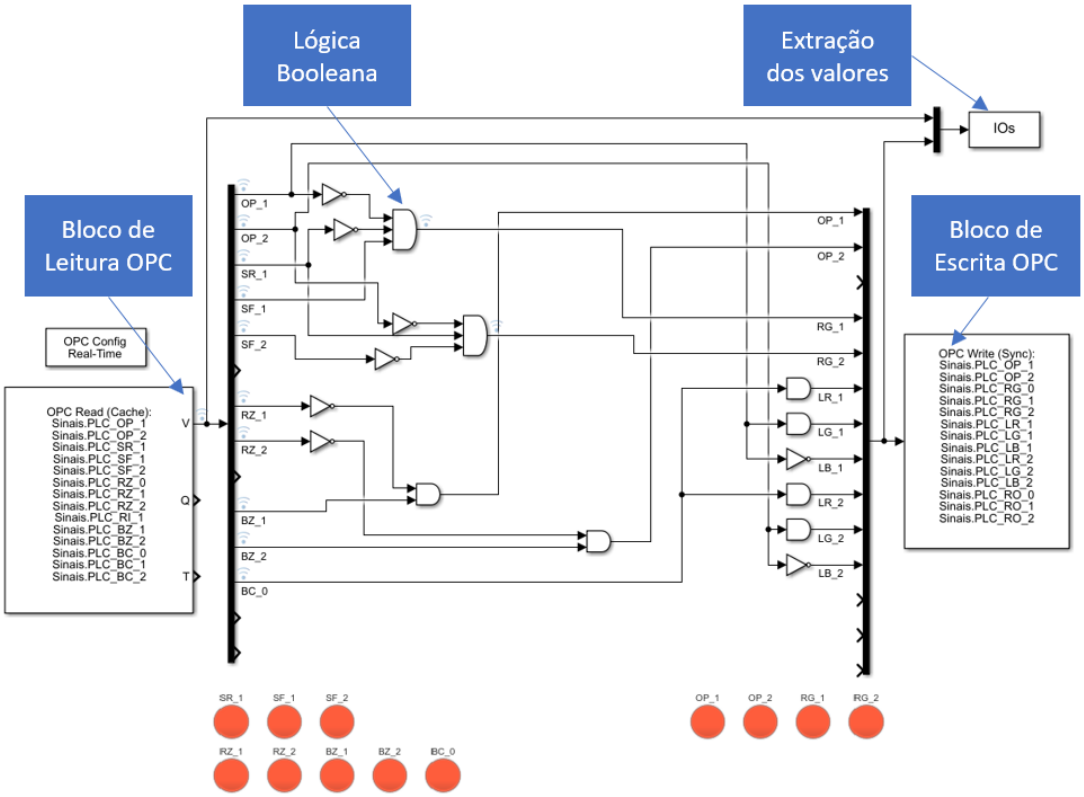
O *software Matlab* é uma ferramenta matemática e um ambiente de desenvolvimento computacional. Sua interface trabalha com programação de alto nível e por linguagem de blocos através do pacote denominado *Simulink*. Devido a sua flexibilidade e capacidade de expansão, o *software Matlab* pode ser utilizado em diversas áreas de conhecimento, como análise de elementos finitos, inteligência artificial, análise de sinais etc. A extensão *Simulink* é uma ferramenta para modelagem, simulação e análise de sistemas dinâmicos com interface de diagramação gráfica por blocos, oferecendo integração com o resto do ambiente *Matlab*. O pacote *Simulink* é amplamente usado em teoria de controle e processamento digital de sinais (MATHWORKS, 2019).

4.3.2. Controle *booleano*

Visando a obtenção dos dados para o treinamento da rede neural, a primeira modelagem de controle elaborada foi representada por uma lógica booleana de acordo com o correto funcionamento dos sinais da célula. Portanto, em um modelo *Simulink*, são adicionados os blocos de conexão OPC configurados de acordo com o servidor utilizado, do qual recebe e atualiza as informações dos estados dos sinais da célula. De um bloco de leitura OPC, é feita uma expansão do vetor dos sinais de entrada, e desses sinais, são inseridos os blocos lógicos de forma a atender a lógica de processo da célula. Após passar por esses blocos lógicos, os valores dos sinais são atualizados e arranjados em forma de vetor novamente, e assim, transmitido para o bloco de escrita OPC, que irá atualizar os valores das variáveis no servidor.

O sistema controlado por blocos lógicos é representado figura 18. É possível observar que há um fluxo de informações da esquerda (iniciando-se no bloco de leitura do servidor OCP) para a direita (bloco de escrita no servidor OPC). Entre os blocos de comunicação há a presença das portas lógicas arranjadas de forma a representar corretamente a lógica de operação da célula transformando os valores recebidos e os transmitindo para saída.

Figura 18 – Sistema por controle booleano



Fonte: Autor

Para conseguir extrair os dados necessários para o treinamento, foi adicionado ao sistema de controle um bloco (denominado como “IOs”) que transmite os valores do vetor de sinais de entrada e do vetor de sinais de saída para uma variável do *Matlab*. Em cada instante de processamento, esses valores são adicionados e armazenados a essa variável, resultando em uma matriz contendo os valores de cada sinal do processo, registrando assim, todas as variações lógicas existentes.

Estando estabelecida a integração entre o software de simulação e o software de controle, é necessário, portanto, executar o modelo de simulação integrado ao controle booleano a fim de se explorar o máximo de situações e condições de processo, abrangendo a maior gama

de combinações de sinais possível. Tendo registrado a matriz contendo os valores dos sinais durante a simulação, esses valores são exportados para um arquivo do tipo "*.csv*", que será utilizado para alimentar o processo de treinamento posteriormente.

Foram extraídas 44 combinações diferentes de entradas e saídas para a realização do treinamento neural, conforme apresentado na tabela 2.

Tabela 2 – Dados exportados

	Entradas	**Saídas**
1	00000100000000	00010001001000
2	00000100010000	00000000001000
3	10000100010000	10000010001000
4	10010100010000	10000010001000
5	10010100000000	10000010001000
6	00010100000000	00010001001000
7	00010010000000	00000001001000
8	00110010100000	00001001001000
9	00100010100000	00001001001000
10	00100000100000	00001001001000
11	00100000110000	00001000001000
12	10100000110000	10000010001000
13	10110000110000	10000010001000
14	10110000100000	10000000001000
15	00110000100000	00001001001000
16	00110001100000	00001001001000
17	00011001000000	00010001001000
18	00011110000000	00010001001000
19	00011010001000	00010001000000
20	01011010001000	01000001000000
21	01111010101000	01000001010000
22	01101010101000	01000001010000
23	01100010101000	01000001010000
24	01100000101000	01000001010000
25	01100000111000	01000000010000
26	11100000111000	10000010010000
27	11110000111000	10000010010000
28	11110000110000	10000010000000
29	10110001110000	10000010001000
30	10111001010000	10000010001000
31	10011001010000	10000010001000
32	10011100010000	10000010001000
33	10011100000000	00000010001000
34	00011100000000	00010000001000
35	00011010000000	00010001001000
36	00111010100000	00000001001000
37	00101010100000	00000001001000
38	00101000100000	00000001001000
39	00101000101000	00000001000000
40	01101000101000	01000001010000
41	01100000100000	01000001010000
42	00100001100000	00001001001000
43	00101001100000	00000001001000
44	00001001000000	00010001001000

Fonte: Autor

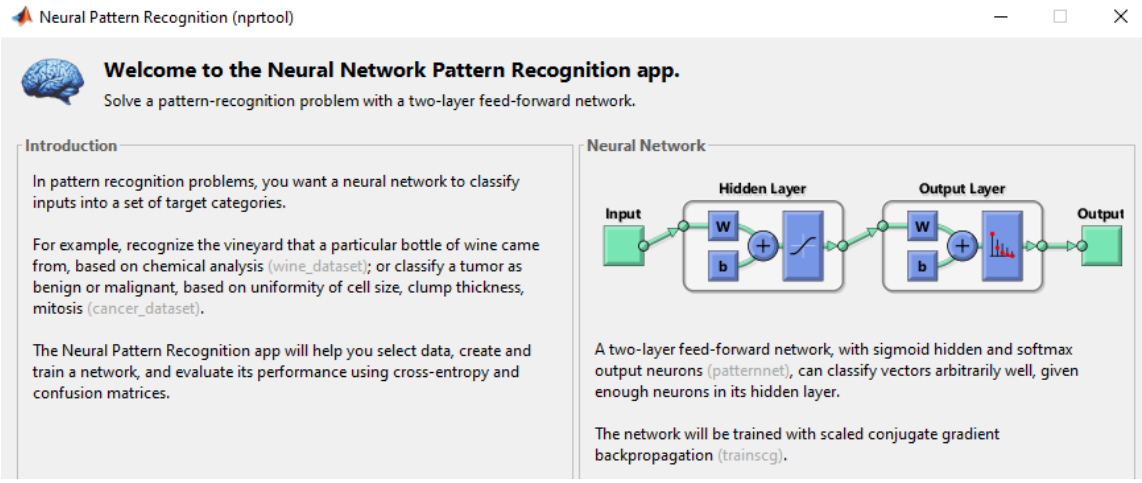
4.4. REDES NEURAIS

4.4.1. Treinamento

Com os dados de treinamento exportados, é utilizado o módulo de reconhecimento de padrão de rede neural do *software Matlab*, que permite resolver problemas de classificação de dados usando uma rede *feed-forward* do tipo *Perceptron* multicamadas. O algoritmo de aprendizado supervisionado utilizado pelo módulo neural é o de retropropagação por gradiente conjugado, que garante uma convergência mais rápida.

A figura 19 apresenta o módulo de reconhecimento de padrão de rede neural do *software Matlab*, que dispõe de uma interface de utilização e parametrização. Ao se utilizar o módulo, deve-se importar os dados extraídos da execução de controle booleano, separando os sinais de entrada dos sinais de saída.

Figura 19 – Módulo de treinamento neural

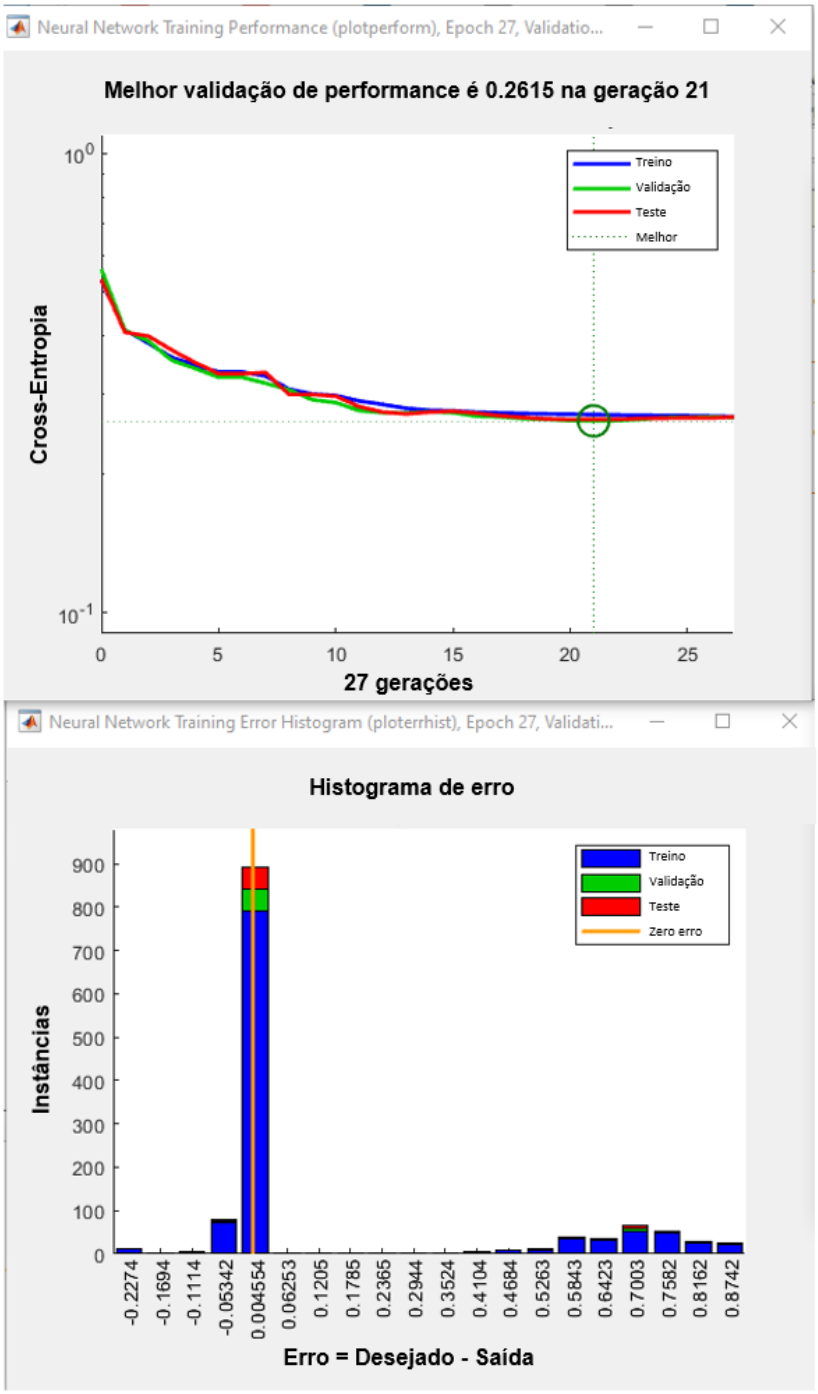


Fonte: Autor

Após importar os dados de entrada e saída, deve-se configurar a topologia da rede neural utilizada. A rede neural do módulo de reconhecimento de padrões é constituída de duas camadas, a camada escondida, com o número de neurônios configuráveis e função sigmoide de saída, e a camada de saída, que processa e classifica os valores da camada anterior de acordo com a quantidade de saídas do sistema.

Para determinar a quantidade de neurônios a ser utilizada no modelo proposto, foram realizados testes com algumas topologias. O primeiro teste utilizou-se de 3 neurônios na camada escondida, o que apresentou, de forma geral, um erro que se encontra plotado nos gráficos da figura 20.

Figura 20 – Erro de treinamento com 3 neurônios



Fonte: Autor

Para verificar o quanto a rede conseguiu aprender da lógica demonstrada, foi elaborada uma tabela com as saídas desejadas comparadas as saídas obtidas do modelo, calculando a quantidade de acertos entre as combinações. A tabela 3 apresenta que para 3 neurônios, de maneira geral, é obtida uma taxa de aprendizagem de 56,8% das combinações ensinadas.

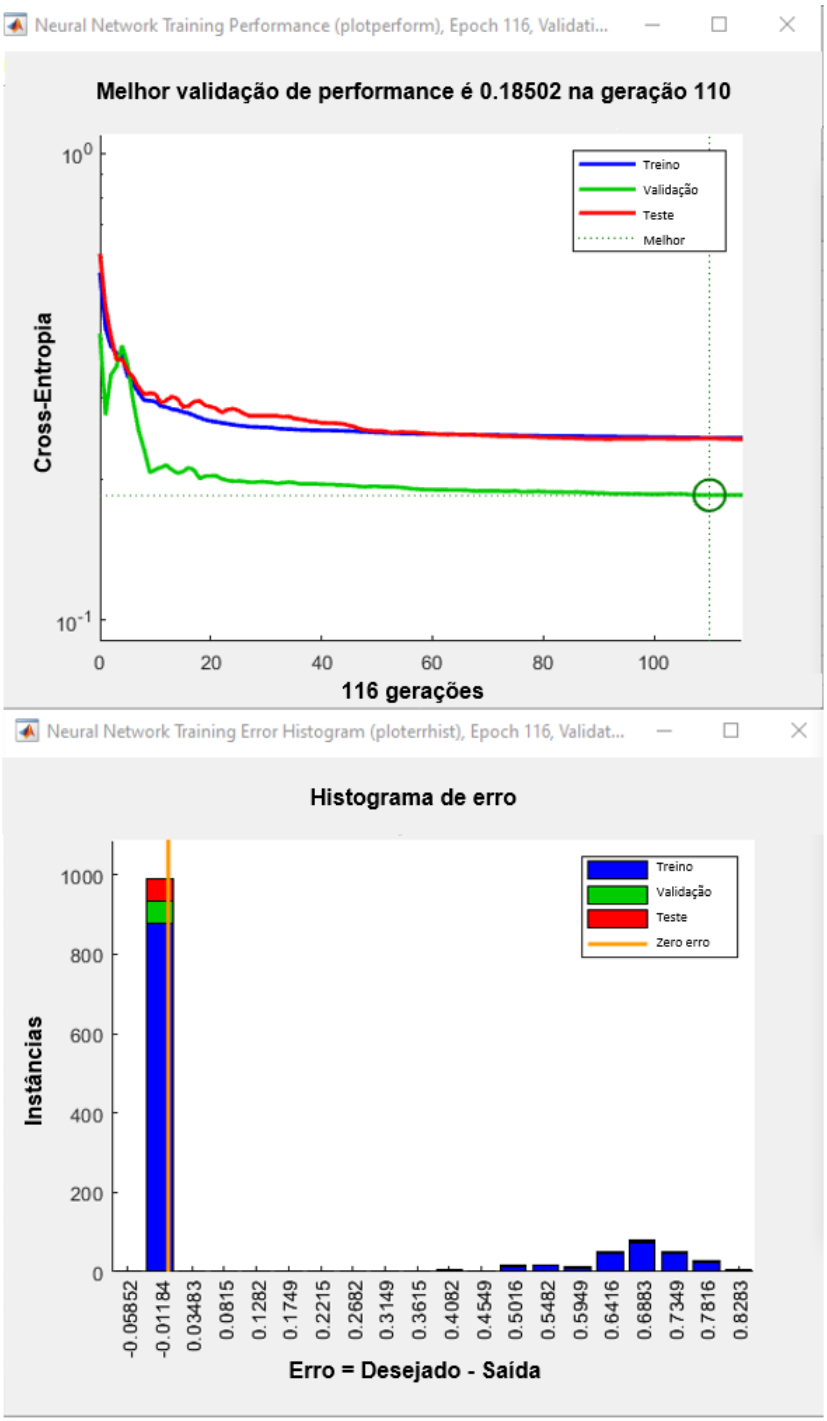
Tabela 3 – Rede com 3 neurônios

	Demonstrado	Aprendido	Acertos
1	00000001001000	00010001001000	0
2	10000001001000	00000000001000	0
3	10000010001000	10000010001000	1
4	10000010001000	10000010001000	1
5	00000010001000	10000010001000	0
6	00010001001000	00010001001000	1
7	00010001001000	00000001001000	0
8	00001001001000	00001001001000	1
9	00001001001000	00001001001000	1
10	00001001001000	00001001001000	1
11	10001001001000	00001000001000	0
12	10001010001000	10000010001000	0
13	10001010001000	10000010001000	0
14	00001010001000	10000000001000	0
15	00001001001000	00001001001000	1
16	00001001001000	00001001001000	1
17	00010001001000	00010001001000	1
18	00010001001000	00010001001000	1
19	01010001001000	00010001000000	0
20	01010001010000	01000001000000	0
21	01000001010000	01000001010000	1
22	01000001010000	01000001010000	1
23	01000001010000	01000001010000	1
24	01000001010000	01000001010000	1
25	11000001010000	01000000010000	0
26	11000010010000	10000010010000	0
27	11000010010000	10000010010000	0
28	10000010010000	10000010000000	0
29	10001010001000	10000010001000	0
30	10000010001000	10000010001000	1
31	10000010001000	10000010001000	1
32	10000010001000	10000010001000	1
33	00000010001000	00000010001000	1
34	00010001001000	00010000001000	0
35	00010001001000	00010001001000	1
36	00000001001000	00000001001000	1
37	00000001001000	00000001001000	1
38	00000001001000	00000001001000	1
39	01000001001000	00000001000000	0
40	01000001010000	01000001010000	1
41	00000001010000	01000001010000	0
42	00001001001000	00001001001000	1
43	00000001001000	00000001001000	1
44	00000001001000	00010001001000	0
			56,80%

Fonte: Autor

Como verificado que 3 neurônios na camada escondida não é o suficiente para a representação da lógica ensinada, foi realizado um treinamento com 5 neurônios, o que apresentou, de maneira geral, o erro apresentado na figura 21.

Figura 21 – Erro de treinamento com 5 neurônios



Fonte: Autor

É possível perceber que a topologia com 5 neurônios apresentou, de forma geral, uma taxa de aprendizagem de 93,18% das combinações ensinadas, conforme a tabela 4, embora tenha se verificado uma única vez um treinamento que conseguiu aprender 100% das combinações.

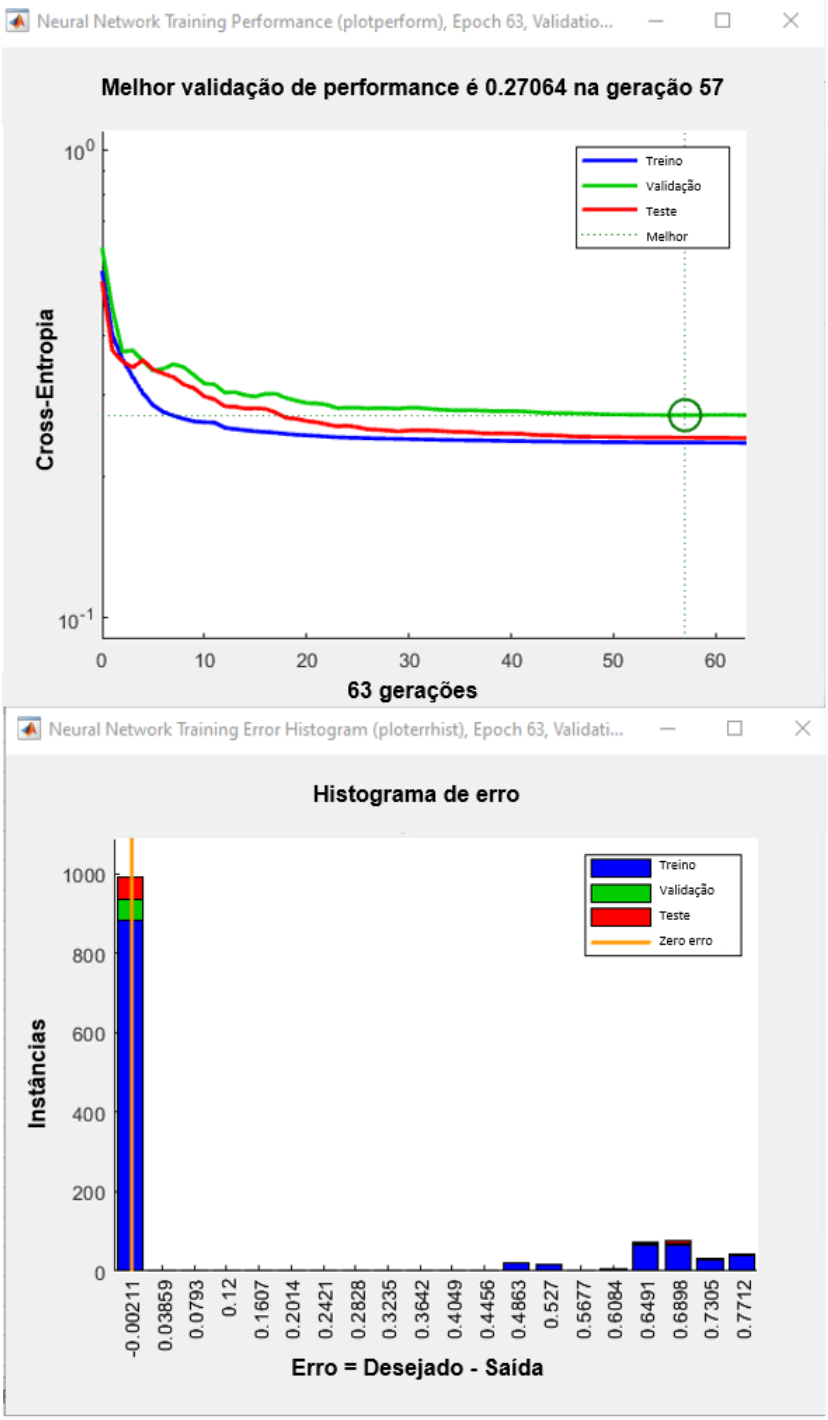
Tabela 4 – Rede com 5 neurônios

	Demonstrado	Aprendido	Acertos
1	00000001001000	00000001001000	1
2	10000001001000	10000001001000	1
3	10000010001000	10000010001000	1
4	10000010001000	10000010001000	1
5	00000010001000	00000010001000	1
6	00010001001000	00010001001000	1
7	00010001001000	00010001001000	1
8	00001001001000	00001001001000	1
9	00001001001000	00001001001000	1
10	00001001001000	00001001001000	1
11	10001001001000	10001001001000	1
12	10001010001000	00001010001000	0
13	10001010001000	10001010001000	1
14	00001010001000	00001010001000	1
15	00001001001000	00001001001000	1
16	00001001001000	00001001001000	1
17	00010001001000	00010001001000	1
18	00010001001000	00010001001000	1
19	01010001001000	00010001001000	0
20	01010001010000	01010001010000	1
21	01000001010000	01000001010000	1
22	01000001010000	01000001010000	1
23	01000001010000	01000001010000	1
24	01000001010000	01000001010000	1
25	11000001010000	11000001010000	1
26	11000010010000	11000010000000	0
27	11000010010000	11000010010000	1
28	10000010010000	10000010010000	1
29	10001010001000	10001010001000	1
30	10000010001000	10000010001000	1
31	10000010001000	10000010001000	1
32	10000010001000	10000010001000	1
33	00000010001000	00000010001000	1
34	00010001001000	00010001001000	1
35	00010001001000	00010001001000	1
36	00000001001000	00000001001000	1
37	00000001001000	00000001001000	1
38	00000001001000	00000001001000	1
39	01000001001000	01000001001000	1
40	01000001010000	01000001010000	1
41	00000001010000	00000001010000	1
42	00001001001000	00001001001000	1
43	00000001001000	00000001001000	1
44	00000001001000	00000001001000	1
			93,18%

Fonte: Autor

Por fim, verificando-se que com 5 neurônios nem sempre é obtido 100% de aprendizagem, foi realizado um treinamento com 8 neurônios, o que apresentou, de maneira geral, o erro apresentado na figura 22.

Figura 22 – Erro de treinamento com 8 neurônios



Fonte: Autor

Verifica-se uma diminuição do erro de treinamento e, portanto, é constatado que 8 neurônios na camada escondida conseguiram representar, de maneira geral, 100% das combinações ensinadas, conforme a tabela 5.

Tabela 5 – Rede com 8 neurônios

	Demonstrado	Aprendido	Acertos
1	00000001001000	00000001001000	1
2	10000001001000	10000001001000	1
3	10000010001000	10000010001000	1
4	10000010001000	10000010001000	1
5	00000010001000	00000010001000	1
6	00010001001000	00010001001000	1
7	00010001001000	00010001001000	1
8	00001001001000	00001001001000	1
9	00001001001000	00001001001000	1
10	00001001001000	00001001001000	1
11	10001001001000	10001001001000	1
12	10001010001000	10001010001000	1
13	10001010001000	10001010001000	1
14	00001010001000	00001010001000	1
15	00001001001000	00001001001000	1
16	00001001001000	00001001001000	1
17	00010001001000	00010001001000	1
18	00010001001000	00010001001000	1
19	01010001001000	01010001001000	1
20	01010001010000	01010001010000	1
21	01000001010000	01000001010000	1
22	01000001010000	01000001010000	1
23	01000001010000	01000001010000	1
24	01000001010000	01000001010000	1
25	11000001010000	11000001010000	1
26	11000010010000	11000010010000	1
27	11000010010000	11000010010000	1
28	10000010010000	10000010010000	1
29	10001010001000	10001010001000	1
30	10000010001000	10000010001000	1
31	10000010001000	10000010001000	1
32	10000010001000	10000010001000	1
33	00000010001000	00000010001000	1
34	00010001001000	00010001001000	1
35	00010001001000	00010001001000	1
36	00000001001000	00000001001000	1
37	00000001001000	00000001001000	1
38	00000001001000	00000001001000	1
39	01000001001000	01000001001000	1
40	01000001010000	01000001010000	1
41	00000001010000	00000001010000	1
42	00001001001000	00001001001000	1
43	00000001001000	00000001001000	1
44	00000001001000	00000001001000	1
			100,00%

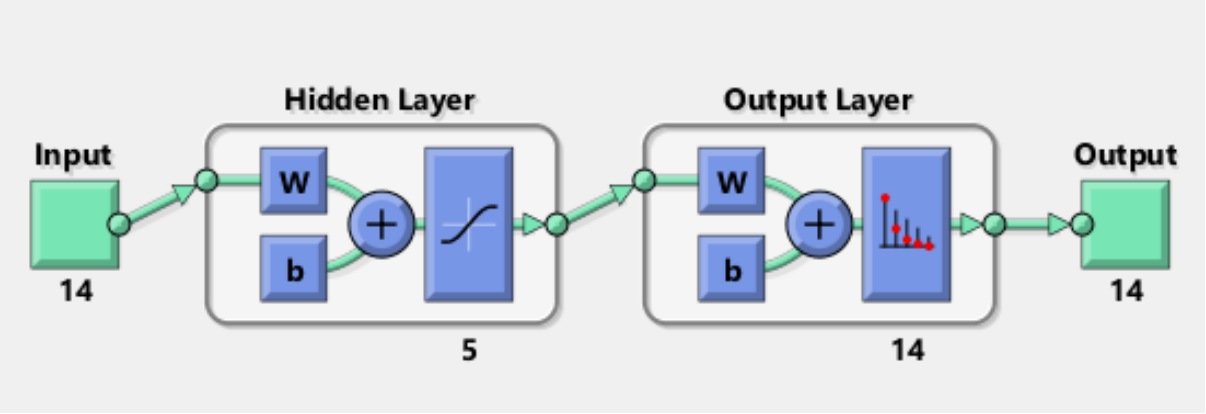
Fonte: Autor

Após essa análise, constatou-se que, para as 44 combinações lógicas ensinadas, a topologia da rede deve conter um número de 5 a 8 neurônios em sua camada escondida de forma a garantir uma correta operação da célula estudada. Como o objetivo dessa rede é adquirir a lógica a ela apresentada e não a realização de uma generalização, todo o conjunto de dados obtidos devem ser usados para o processo de treinamento, sendo utilizado pelo módulo de treinamento neural, aleatoriamente e automaticamente, 5% desse mesmo conjunto para teste e validação.

O processo de treinamento é finalizado automaticamente quando o erro do conjunto de validação para de diminuir, apresentando a geração de melhor desempenho. Como os pesos sinápticos iniciais apresentam valores aleatórios, o treinamento deve ser executado algumas vezes até que se verifique um resultado de treinamento melhor e satisfatório.

Portanto, o modelo neural escolhido, ilustrado pela figura 23, tem 14 entradas e 14 saídas, constituído de 5 neurônios na camada escondida, do qual, seu treinamento obteve 100% de acertos das combinações apresentadas.

Figura 23 – Estrutura da rede neural utilizada

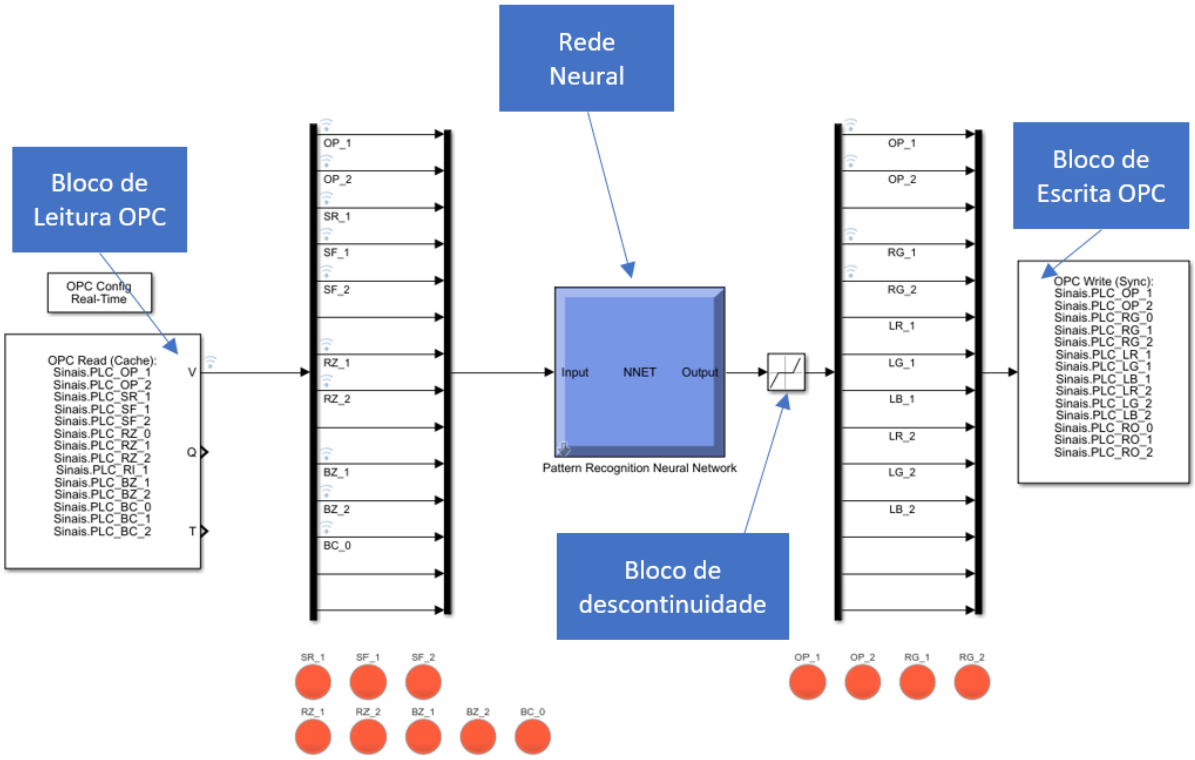


Fonte: Autor

4.4.2. Controle por redes neurais

Estando tudo definido e com um resultado satisfatório de aprendizado, a rede é compilada em um único bloco do pacote *Simulink*, que irá substituir toda a lógica booleana do sistema de controle. Portanto, o sistema de controle final pode ser conferido na figura 24:

Figura 24 – Sistema controlado por rede neural



Fonte: Autor

Semelhantemente ao sistema *booleano* anterior, o sistema atualizará os valores dos sinais do servidor OPC de acordo com os sinais emitidos pelo software de simulação. É importante ressaltar, porém, a necessidade de se utilizar um bloco para acrescentar uma descontinuidade na saída da rede neural, pois, o valor resultante da saída do bloco neural não é booleano, e por não se tratar mais de um sinal booleano, quaisquer valores diferentes de zero seriam interpretados como verdadeiro. Portanto, é necessária a inserção de um selecionador de valores verdadeiros ou falsos, como por exemplo um bloco de zona morta, fazendo com que somente sinais maiores do que determinado valor sejam considerados diferentes de zero, resolvendo assim o problema na definição de valores verdadeiros ou falsos.

Para seleção do melhor valor para esse bloco de descontinuidade foram realizados testes alterando-se o valor do limiar de seleção entre verdadeiro ou falso e comparando esses resultados com os valores desejados. O primeiro teste iniciou-se com o valor de 0,1, conforme a tabela 6.

Tabela 6 – Seleção em 0,1

	Desejado	Obtido	Acertos
1	00000001001000	00000001001000	1
2	10000001001000	10000001001000	1
3	10000010001000	10000010001000	1
4	10000010001000	10000010001000	1
5	00000010001000	00000010001000	1
6	00010001001000	00010001001000	1
7	00010001001000	00010001001000	1
8	00001001001000	00001001001000	1
9	00001001001000	00001001001000	1
10	00001001001000	00001001001000	1
11	10001001001000	10001001001000	1
12	10001010001000	10001010001000	1
13	10001010001000	10001010001000	1
14	00001010001000	00001010001000	1
15	00001001001000	00001001001000	1
16	00001001001000	00001001001000	1
17	00010001001000	00010001001000	1
18	00010001001000	00010001001000	1
19	01010001001000	01010001001000	1
20	01010001010000	01010001010000	1
21	01000001010000	01000001010000	1
22	01000001010000	01000001010000	1
23	01000001010000	01000001010000	1
24	01000001010000	01000001010000	1
25	11000001010000	11000001010000	1
26	11000010010000	11000010010000	1
27	11000010010000	11000010010000	1
28	10000010010000	10000010010000	1
29	10001010001000	10001010001000	1
30	10000010001000	10000010001000	1
31	10000010001000	10000010001000	1
32	10000010001000	10000010001000	1
33	00000010001000	10000010001000	0
34	00010001001000	00010001001000	1
35	00010001001000	00010001001000	1
36	00000001001000	00000001001000	1
37	00000001001000	00000001001000	1
38	00000001001000	00000001001000	1
39	01000001001000	01000001001000	1
40	01000001010000	01000001010000	1
41	00000001010000	00000001010000	1
42	00001001001000	00001001001000	1
43	00000001001000	00000001001000	1
44	00000001001000	00000001001000	1
			97,72%

Fonte: Autor

É possível perceber uma taxa de acerto de 97,72% para a correta distinção entre valores verdadeiros ou falsos. O próximo valor de limiar testado foi o de 0,2, o que apresentou os resultados verificados na tabela 7.

Tabela 7 – Seleção em 0,2

	Desejado	Obtido	Acertos
1	00000001001000	00000001001000	1
2	10000001001000	10000001001000	1
3	10000010001000	10000010001000	1
4	10000010001000	10000010001000	1
5	00000010001000	00000010001000	1
6	00010001001000	00010001001000	1
7	00010001001000	00010001001000	1
8	00001001001000	00001001001000	1
9	00001001001000	00001001001000	1
10	00001001001000	00001001001000	1
11	10001001001000	10001001001000	1
12	10001010001000	10001010001000	1
13	10001010001000	10001010001000	1
14	00001010001000	00001010001000	1
15	00001001001000	00001001001000	1
16	00001001001000	00001001001000	1
17	00010001001000	00010001001000	1
18	00010001001000	00010001001000	1
19	01010001001000	01010001001000	1
20	01010001010000	01010001010000	1
21	01000001010000	01000001010000	1
22	01000001010000	01000001010000	1
23	01000001010000	01000001010000	1
24	01000001010000	01000001010000	1
25	11000001010000	11000001010000	1
26	11000010010000	11000010010000	1
27	11000010010000	11000010010000	1
28	10000010010000	10000010010000	1
29	10001010001000	10001010001000	1
30	10000010001000	10000010001000	1
31	10000010001000	10000010001000	1
32	10000010001000	10000010001000	1
33	00000010001000	00000010001000	1
34	00010001001000	00010001001000	1
35	00010001001000	00010001001000	1
36	00000001001000	00000001001000	1
37	00000001001000	00000001001000	1
38	00000001001000	00000001001000	1
39	01000001001000	01000001001000	1
40	01000001010000	01000001010000	1
41	00000001010000	00000001010000	1
42	00001001001000	00001001001000	1
43	00000001001000	00000001001000	1
44	00000001001000	00000001001000	1
			100,00%

Fonte: Autor

É possível perceber que houve uma taxa de acerto de 100% na distinção entre os valores verdadeiros ou falsos. O teste com o valor de limiar em 0,25 foi então realizado e apresentado na tabela 8.

Tabela 8 – Seleção em 0,25

	Desejado	Obtido	Acertos
1	00000001001000	00000001001000	1
2	10000001001000	10000001001000	1
3	10000010001000	10000010001000	1
4	10000010001000	10000010001000	1
5	00000010001000	00000010001000	1
6	00010001001000	00010001001000	1
7	00010001001000	00010001001000	1
8	00001001001000	00001001001000	1
9	00001001001000	00001001001000	1
10	00001001001000	00001001001000	1
11	10001001001000	00001001000000	0
12	10001010001000	10001000001000	0
13	10001010001000	10001000000000	0
14	00001010001000	00001010001000	1
15	00001001001000	00001001001000	1
16	00001001001000	00001001001000	1
17	00010001001000	00010001001000	1
18	00010001001000	00010001001000	1
19	01010001001000	00010001000000	0
20	01010001010000	00010000000000	0
21	01000001010000	01000001010000	1
22	01000001010000	01000001010000	1
23	01000001010000	01000001010000	1
24	01000001010000	01000001010000	1
25	11000001010000	10000000000000	0
26	11000010010000	01000000010000	0
27	11000010010000	01000000010000	0
28	10000010010000	10000010010000	1
29	10001010001000	00000010001000	0
30	10000010001000	10000010001000	1
31	10000010001000	10000010001000	1
32	10000010001000	10000010001000	1
33	00000010001000	00000010001000	1
34	00010001001000	00010001001000	1
35	00010001001000	00010001001000	1
36	00000001001000	00000001001000	1
37	00000001001000	00000001001000	1
38	00000001001000	00000001001000	1
39	01000001001000	01000001001000	1
40	01000001010000	01000001010000	1
41	00000001010000	00000001010000	1
42	00001001001000	00001001001000	1
43	00000001001000	00000001001000	1
44	00000001001000	00000001001000	1
			79,54%

Fonte: Autor

É constatado uma significativa diminuição na taxa de acerto entre valores verdadeiros ou falsos, concluindo-se, portanto, que o valor mais indicado para separação de variáveis verdadeira ou falsa deve estrar entre 0,1 e 0,2. Para o modelo estudado, foi adotado o valor de 0,2, garantindo 100% de acertos para as combinações ensinadas.

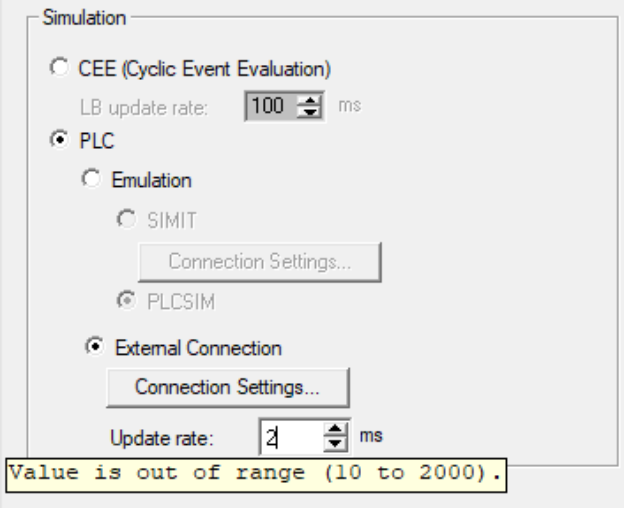
É possível concluir, portanto, que para saídas do sistema neural, que não mais encontram-se no universo discreto de 0 ou 1 e sim qualquer valor entre 0 e 1, sendo 0 saída totalmente desativada e 1 saída totalmente ativada, algumas combinações não atingiram um nível maior do que 0,25 para uma assumir um valor verdadeiro. Ao utilizar-se de mais neurônios, porém, verifica-se uma melhor distinção entre saídas verdadeiras e falsas, caracterizando que a rede está mais próxima de um *overfitting*.

5. RESULTADOS

5.1. MODELO BOOLEANO

A modelagem do sistema de controle de lógica booleana necessária para extração dos dados de treinamento da rede neural apresentou uma correta funcionalidade, comportando-se como o desejado e de fácil alteração quando havia algum item da lógica que não estava de acordo. Houve uma boa resposta para integração dos softwares de simulação e controle, porém, a velocidade de resposta se dá em função das taxas de atualizações dos sinais desses softwares junto ao servidor OPC. Taxa essa que é configurável e assíncrona entre os softwares, com tempo de ciclo na escala de milissegundos (figura 25). Já alguns computadores lógicos programáveis modernos utilizam tempo de ciclos de poucos microssegundos.

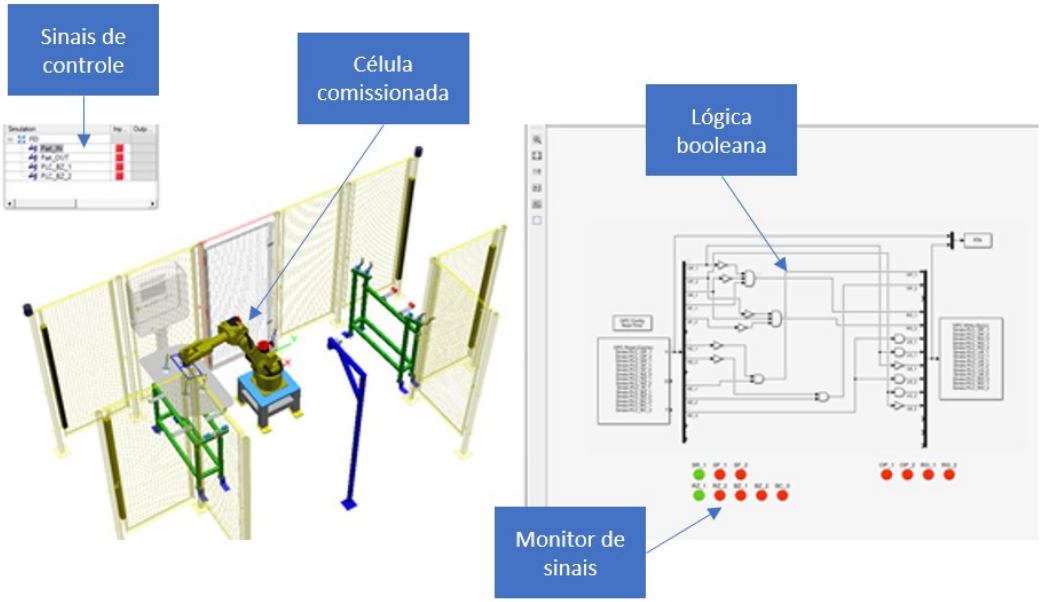
Figura 25 – Taxa de atualização



Fonte: Autor

A Figura 26 demonstra o modelo de simulação sendo executado ao lado do modelo de controle por lógica booleana. Os sinais da célula gerados dentro do modelo de simulação foram transmitidos para o servidor OPC, que por sua vez, eram lidos pelo modelo de controle booleano, e assim, atualizados e enviados novamente para o modelo de simulação. A execução desse modelo gerou os dados de treinamento (Tabela 2) que permitiu o aprendizado do modelo neural, o que demonstra que a lógica aqui empregada será “copiada” para a rede neural e, portanto, a execução e testes das mais variadas condições e situações do processo são imprescindíveis.

Figura 26 – Célula sendo executada por meio de lógica booleana

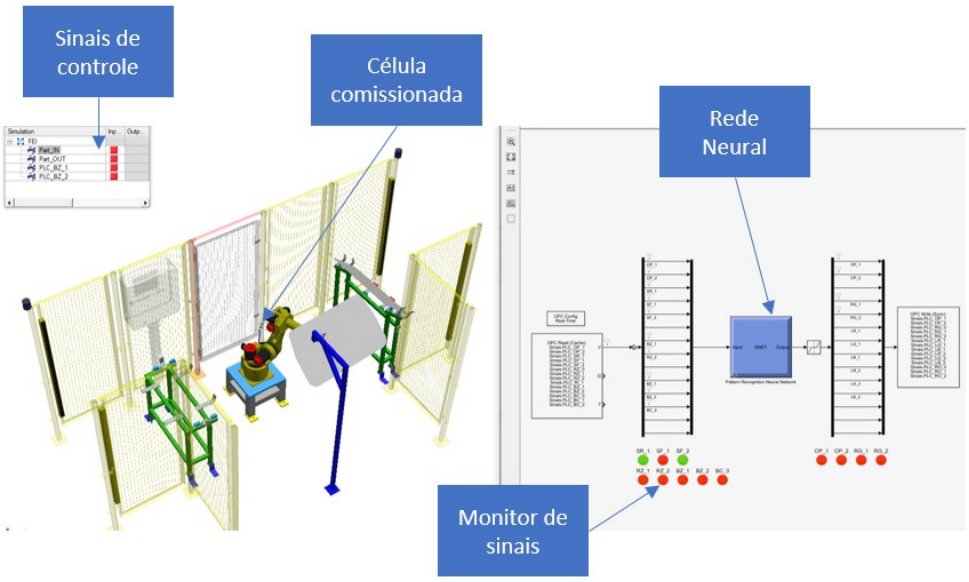


Fonte: Autor

5.2. MODELO NEURAL

Após o treinamento da rede neural por meio dos dados extraídos do modelo anterior, e da troca das portas lógicas pelo bloco neural resultante, observou-se uma execução funcional do sistema, do qual a célula se comportou, de maneira geral, de acordo com a lógica observada.

Figura 27 – Célula sendo executada por meio de rede neural



Fonte: Autor

Conforme indicado pela Figura 27, onde há a execução de um novo comissionamento virtual do modelo de simulação, sendo dessa vez utilizando-se a rede neural resultante do processo de treinamento. Não houve diferenças ao se comparar a execução da célula no modelo neural com o modelo booleano. A tabela 9 demonstra todas as combinações lógicas extraídas do modelo neural funcionaram corretamente.

Tabela 9 – Resultados obtidos

	Desejado	Obtido	Acertos
1	00000001001000	00000001001000	1
2	10000001001000	10000001001000	1
3	10000010001000	10000010001000	1
4	10000010001000	10000010001000	1
5	00000010001000	00000010001000	1
6	00010001001000	00010001001000	1
7	00010001001000	00010001001000	1
8	00001001001000	00001001001000	1
9	00001001001000	00001001001000	1
10	00001001001000	00001001001000	1
11	10001001001000	10001001001000	1
12	10001010001000	10001010001000	1
13	10001010001000	10001010001000	1
14	00001010001000	00001010001000	1
15	00001001001000	00001001001000	1
16	00001001001000	00001001001000	1
17	00010001001000	00010001001000	1
18	00010001001000	00010001001000	1
19	01010001001000	01010001001000	1
20	01010001010000	01010001010000	1
21	01000001010000	01000001010000	1
22	01000001010000	01000001010000	1
23	01000001010000	01000001010000	1
24	01000001010000	01000001010000	1
25	11000001010000	11000001010000	1
26	11000010010000	11000010010000	1
27	11000010010000	11000010010000	1
28	10000010010000	10000010010000	1
29	10001010001000	10001010001000	1
30	10000010001000	10000010001000	1
31	10000010001000	10000010001000	1
32	10000010001000	10000010001000	1
33	00000010001000	00000010001000	1
34	00010001001000	00010001001000	1
35	00010001001000	00010001001000	1
36	00000001001000	00000001001000	1
37	00000001001000	00000001001000	1
38	00000001001000	00000001001000	1
39	01000001001000	01000001001000	1
40	01000001010000	01000001010000	1
41	00000001010000	00000001010000	1
42	00001001001000	00001001001000	1
43	00000001001000	00000001001000	1
44	00000001001000	00000001001000	1
			100,00%

Fonte: Autor

Verifica-se, porém, que antes de se chegar a uma condição satisfatória, ocorreram algumas condições inesperadas em algumas execuções do controle neural, como por exemplo o sinalizador luminoso que estava acesso com determinada cor e, para uma determinada combinação de sinais, veio a se apagar e acender novamente quando mudou-se aquela combinação. Esse tipo de comportamento de indecisão indica que houve algumas condições de processo que não foram bem amostradas nos dados de treinamento, levando uma determinada situação do qual a rede neural não havia aprendido. Portanto, foi necessária uma nova extração de dados que ensinassem essa condição.

Sabendo-se que a rede tem por objetivo replicar um conjunto de combinações lógicas apresentado e não a generalização para entradas desconhecidas, essa pesquisa vai contra uma das maiores vantagens na utilização de redes neurais artificiais, além de, ao contrário de outros modelos, o *overfitting* não será um malefício, já que todas as amostras devem ser utilizadas para o treinamento. Entretanto, uma única forma de se garantir que toda lógica tenha sido adequadamente demonstrada seria por meio de uma tabela da verdade com todas as combinações possíveis de entradas e saídas, o que, em alguns casos, pode se tornar algo inviável, pois seria necessária 2^i (sendo "i" a quantidade de entradas do sistema) combinações diferentes para serem utilizadas no conjunto de treinamento. Para essa pesquisa, por exemplo, uma tabela da verdade seria constituída de $2^{14} = 16.384$ combinações lógicas diferentes para garantir que todas condições possíveis foram amostradas.

Portanto, devido ao seu propósito de não ser uma rede de generalização, combinações lógicas não utilizadas no processo de treinamento podem ou não apresentar uma saída correta, mas, por não se tratar de uma relação linear entre entradas e saídas, sempre estarão no campo da incerteza. O que indica a importância de uma boa amostragem neste caso, e que a quantidade de combinações lógicas disponíveis, ou possíveis de serem amostradas, já poderiam inviabilizar esta abordagem em uma determinada aplicação.

6. CONCLUSÕES

Este trabalho apresenta a implementação de um comissionamento virtual de uma célula robotizada por meio de redes neurais artificiais em substituição dos controladores lógicos programáveis convencionais. O modelo de controle proposto nesta pesquisa visou ser uma alternativa dos métodos de controle já concretizados na indústria, em uma abordagem integrada com a indústria 4.0 no que tange a utilização de métodos de inteligência artificial nos processos industriais.

A modelagem utilizada trata de uma célula virtualizada integrada com um controlador neural, cuja lógica de operação foi extraída de um modelo booleano. Portanto, é necessário desenvolver inicialmente um modelo funcional de controle do processo, para que dele sejam extraídos os dados para o treinamento da rede neural. Logo, conclui-se que essa abordagem não elimina a programação detalhada para o funcionamento do processo, pois, o modelo neural por si só não saberia a lógica que deveria ser executada.

Ao se extrair as informações necessárias para o treinamento, são apresentados os pares de entradas e saídas à rede neural que, a partir desses dados, aprende o comportamento esperado da linha e o compila em um modelo matemático contido em seus neurônios artificiais. Porém, para isso, é fundamental que seja extraído o máximo de situações e combinações diferentes do processo. A rede não tem como objetivo a generalização de saídas desconhecidas, mas sim a memorização do comportamento lógico de um processo.

Quando foi apresentado um conjunto de entradas que não havia sido aprendido, a rede apresentou um valor de saída diferente do que se era esperado. Ao se verificar que a lógica ainda não estava adequada, houve a necessidade de se realizar uma nova extração de dados que contemplem as condições lógicas faltantes, já que, não é possível alterar a lógica contida dentro do modelo neural, sendo necessário refazer todo o processo de treinamento. A única forma de se garantir que todas as condições lógicas possíveis foram corretamente amostradas, seria por meio de uma tabela da verdade, o que, dependendo da quantidade de entradas, pode ser algo inviável.

É possível concluir, portanto, que se tornou possível a implementação do modelo proposto, porém, este não substitui as etapas de programação convencionais, uma vez que é necessário treinar a rede com dados funcionais do processo, além de acrescentar as etapas de extração de dados e de treinamento ao se programar uma célula robotizada, o que torna essa abordagem, a princípio, não tão vantajosa comparada aos métodos convencionais.

Em contrapartida, a compilação da lógica de um processo em um modelo matemático, pode trazer ganhos de execução quando se comparada a um CLP convencional, possibilitando novos tipos de aplicações com a utilização de inteligência artificial e suas ferramentas, como por exemplo um monitor neural, complementando um CLP e analisando se seu comportamento lógico está adequado, ou a possibilidade de agregar um comportamento lógico booleano a uma NPU (unidade de processamento neural), e assim, não deixar a rede neural somente para o tratamento de sinais e as decisões lógicas para o CLP. Há também a possibilidade implementar o comportamento adquirido com o método proposto em um *hardware* de baixo custo, e desta forma, haveria a lógica neural compilada em forma de *hardware* para aplicações onde não haja necessidade de reprogramação.

Portanto, em uma situação convencional, a utilização do CLP ainda se mantém mais usual, tendo mais controle sobre a lógica aplicada e sendo ela facilmente reprogramada. Esta pesquisa propõe para trabalhos futuros a utilização de uma rede de controle que tenha seu treinamento de forma *on-line*, e não *off-line* como foi realizado, para que dessa forma, a rede neural consiga aprender o comportamento do processo de forma adaptativa, sem precisar refaze-lo completamente, e desta forma, tornar mais interessante e compensatório a substituição do controlador lógico convencional para um modelo de aprendizado de máquina.

REFERÊNCIAS

ABDI, Hamid; SALAMI, Abolfazl; AHMADI, Abolfazl. Implementation of a New Neural Network Function Block to Programmable Logic Controllers Library Function. **International Journal of Computer and Systems Engineering**, p. 147–150, 2007.

ALPHONSUS, Ephrem Ryan; ABDULLAH, Mohammad Omar. A review on the applications of programmable logic controllers (PLCs). **Renewable and Sustainable Energy Reviews**, v. 60, n. July 2017, p. 1185–1205, 2016. Disponível em: <http://dx.doi.org/10.1016/j.rser.2016.01.025>.

ARAS, Akın et al. Investigation on Industry 4 . 0 and Virtual Commissioning. v. 4, n. 2, p. 107–113, 2018.

BAGHIRLI, O. Comparison of Lavenberg-Marquardt, Scaled Conjugate Gradient And Bayesian Regularization Backpropagation Algorithms for Multistep Ahead Wind Speed Forecasting Using Multilayer Perceptron Feedforward Neural Network. **Dissertation**, n. June, p. Uppsala University, 2015.

BINNBERG, Dennis et al. VIRTUAL COMMISSIONING Emulation of a production cell. 2016.

CARLSSON, Henrik et al. Methods for reliable simulation-based PLC code verification. **IEEE Transactions on Industrial Informatics**, v. 8, n. 2, p. 267–278, 2012.

CHRYSSOLOURIS, G. et al. Digital manufacturing: History, perspectives, and outlook. 2009, [S.l: s.n.], 2009. p. 451–462.

CREATIONS, Norwegian. **An Intro to Boolean Algebra and Logic Gates**. Disponível em: <https://www.norwegiancreations.com/2017/10/an-intro-to-boolean-algebra-and-logic-gates-part-1/>. Acesso em: 11 jul. 2019.

DAHL, M. et al. Integrated Virtual Preparation and Commissioning: supporting formal methods during automation systems development. **IFAC-PapersOnLine**, v. 49, n. 12, p. 1939–1944, 2016.

DRATH, Rainer; WEBER, Peter; MAUSER, Nicolas. An evolutionary approach for the industrial introduction of virtual commissioning. 2008, [S.l: s.n.], 2008. p. 5–8.

GREEF, Ana Carolina; DUARTE FREITAS, Maria do Carmo. Industry 4.0 with Cyber-Physical Integration: A Design and Manufacture Perspective. 2015, [S.l: s.n.], 2015. p. 37–55.

JORDAN, M. I.; MITCHELL, T. M. **Machine learning: Trends, perspectives, and prospects**. **Science**. [S.l: s.n.]. , 2015

KA, Botond. CIRP Annals - Manufacturing Technology Semantic Virtual Factory supporting interoperable modelling and evaluation of production systems. v. 62, p. 443–446, 2013.

KHAN, Ateeq; TUROWSKI, Klaus. A Perspective on Industry 4.0: From Challenges to Opportunities in Production Systems. 2016, [S.l: s.n.], 2016. p. 441–448. Disponível em: <http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0005929704410448>.

KO, Minsuk; AHN, Euikoog; PARK, Sang C. A concurrent design methodology of a production system for virtual commissioning. **Concurrent Engineering Research and Applications**, v. 21, n. 2, p. 129–140, 2013.

KUEHN, Wolfgang. Digital factory-integration of simulation enhancing the product and production process towards operative control and Optimisation. **International Journal of Simulation: Systems, Science and Technology**, v. 7, n. 7, p. 27–39, 2006.

LEE, Jay; BAGHERI, Behrad; KAO, Hung An. A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. **Manufacturing Letters**, v. 3, n. October 2017, p. 18–23, 2015. Disponível em: <http://dx.doi.org/10.1016/j.mfglet.2014.12.001>.

LEE, W. B.; CHEUNG, C. F.; LI, J. G. Applications of virtual manufacturing in materials processing. **Journal of Materials Processing Technology**, v. 113, n. 1–3, p. 416–423, 2001.

LEITE, J R Emiliano; MARTINS, Paulo S. A Internet das Coisas (IoT) : Tecnologias e Aplicações. n. December, 2017.

LI, Bo-hu et al. Applications of artificial intelligence in intelligent manufacturing: a review. **Frontiers of Information Technology & Electronic Engineering**, v. 18, n. 1, p. 86–96, 2017. Disponível em: <http://link.springer.com/10.1631/FITEE.1601885>.

LIMA, Elaine Carvalho D E. Revolução Industrial: considerações sobre o pioneirismo industrial inglês. **Revista Espaço Acadêmico**, v. 17, n. 194, p. 102–113, 2017.

MATHWORKS. **No Title**. Disponível em: <https://www.mathworks.com/products/matlab.html>. Acesso em: 20 jun. 2019.

MØLLER, Martin F. PREPRINT A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning Supervised Learning. **Science**, 1990.

MONSEF, Wael A.; AREED, Fayez FG. Design of a Neural – PLC controller FOR Industrial Plant. **Proceedings of the 2006 International Conference on Machine Learning**, 2006.

OPC FOUNDATION. **No Title**. Disponível em: <https://opcfoundation.org/>. Acesso em: 20 jun. 2019.

QIN, Jian; LIU, Ying; GROSVENOR, Roger. A Categorical Framework of Manufacturing for Industry 4.0 and beyond. 2016, [S.l: s.n.], 2016. p. 173–178.

SCHWAB, Klaus. A Quarta Revolução Industrial. **World Economic Forum**, 2016.

SIEMENS. **Artificial intelligence in SIMATIC**. Disponível em: <https://new.siemens.com/global/en/products/automation/systems/industrial/io-systems/artificial-intelligence.html>. Acesso em: 20 jun. 2019.

SIEMENS PLM. **No Title**. Disponível em: <https://www.plm.automation.siemens.com/global/pt/products/tecnomatix/>. Acesso em: 20 jun. 2019.

SILVA, I. N.; SPATTI, D. H.; FLAUZINO, R. A. **Redes Neurais Artificiais Para Engenharia e Ciências Aplicadas**. [S.l: s.n.], 2016.

SUKTHOMYA, W; TANNOCK, J. The training of neural networks to model manufacturing processes. **Journal of Intelligent Manufacturing**, n. 1995, p. 39–51, 2005. Disponível em: <http://link.springer.com/article/10.1007/s10845-005-4823-7>.

TOCCI, Ronald J.; WIDMER, Neal S.; MOSS, Gregory L. **Sistemas Digitais - Princípios e Aplicações**. [S.l: s.n.], 2011.

WANG, Keqin. Intelligent Predictive Maintenance (IPdM) system – Industry 4.0 scenario. **WIT Transactions on Engineering Sciences**, v. 113, p. 259–268, 2016. Disponível em: <https://www.witpress.com/Secure/elibrary/papers/IWAMA15/IWAMA15030FU1.pdf>.

WUEST, Thorsten et al. Machine learning in manufacturing: Advantages, challenges, and applications. **Production and Manufacturing Research**, v. 4, n. 1, p. 23–45, 2016. Disponível em: <http://dx.doi.org/10.1080/21693277.2016.1192517>.

ZHENG, Pai et al. **Smart manufacturing systems for Industry 4.0: Conceptual framework, scenarios, and future perspectives**. **Frontiers of Mechanical Engineering**. [S.l: s.n.]. , 2018

ZHONG, Ray Y. et al. Intelligent Manufacturing in the Context of Industry 4.0: A Review. **Engineering**, v. 3, n. 5, p. 616–630, 2017. Disponível em: <http://dx.doi.org/10.1016/J.ENG.2017.05.015>.

ZHOU, Zude; XIE, Shane; CHEN, Dejun. **Fundamentals of Digital Manufacturing Science**. [S.l.]: Springer, 2012.