

## Substituição do Controle PID por Rede Neural em Ambiente Real com Comunicação Foundation Fieldbus e OPC DA

Alberto Alvares  
Carolina Borges Miranda  
Danilo Carrasco Abrão  
Lucas De Sordi

Universidade de Brasília  
Departamento de Engenharia Mecânica  
Programa de Pós Graduação em Sistemas Mecatrônicos  
alvares@unb.br, carollbm.eng@gmail.com, danilo.eletronica@gmail.com, lucasdesordi@gmail.com

**Resumo.** Este trabalho apresenta a aplicação de uma rede neural artificial como estratégia de controle para uma planta didática baseada na arquitetura SMAR System302, conectada via protocolo Foundation Fieldbus e supervisionada por meio de comunicação OPC DA. A rede foi treinada com dados reais adquiridos da planta operando sob controle PID, sendo posteriormente utilizada para substituir o controlador convencional em tempo real. A implementação ocorreu no ambiente MATLAB, com interface gráfica desenvolvida para interação e supervisão. Os resultados experimentais demonstram que a rede neural foi capaz de manter o sistema estável e com desempenho satisfatório, validando sua aplicabilidade prática. Como proposta de continuidade, destaca-se a integração de conectividade via MQTT e visualização em tempo real utilizando Node-RED, favorecendo a escalabilidade da aplicação para contextos de Indústria 4.0. .

**Palavras chave:** Controle Inteligente, Redes Neurais, Planta Didática SMAR, OPC DA.

**Abstract.** This work presents the application of an artificial neural network as a control strategy for a didactic plant based on the SMAR System302 architecture, connected via FOUNDATION Fieldbus protocol and supervised through OPC DA communication. The network was trained with real data acquired from the plant operating under PID control, and later used to replace the conventional controller in real time. The implementation was carried out in MATLAB, with a graphical user interface developed for supervision and operator interaction. Experimental results show that the neural network was able to maintain system stability and deliver satisfactory performance, validating its practical applicability. As a proposal for future work, we highlight the integration of MQTT connectivity and real-time data visualization using Node-RED, enabling scalability for Industry 4.0 environments.

**Keywords:** Neural Networks, Intelligent Control, SMAR Didactic Plant, OPC DA.

### 1. INTRODUÇÃO

Os sistemas de controle industrial modernos dependem fortemente de controladores PID, dada sua simplicidade, robustez e fácil implementação. No entanto, em plantas com dinâmicas complexas, como a Planta Didática SMAR (System302), esse tipo de controlador pode apresentar limitações em termos de desempenho, principalmente quando há não linearidades, atrasos e perturbações inesperadas.

A justificativa deste trabalho está na crescente demanda por técnicas de controle inteligentes que possam se adaptar a variações da planta sem necessidade de reparametrizações constantes. Entre essas técnicas, as redes neurais artificiais se destacam por sua capacidade de aprender padrões dinâmicos a partir de dados e generalizar comportamentos em tempo real.

A planta utilizada neste estudo é composta por instrumentos de campo reais conectados a um sistema de controle distribuído via protocolo Foundation Fieldbus, tecnologia amplamente adotada em sistemas industriais por permitir comunicação digital, diagnósticos avançados e integração de múltiplos dispositivos em redes determinísticas.

O presente trabalho aborda a substituição do controle PID convencional por uma rede neural treinada para atuar diretamente na planta SMAR por meio de comunicação OPC DA, utilizando MATLAB como plataforma de integração. A arquitetura proposta emprega um modelo perceptron de múltiplas camadas treinado com dados reais da planta, com foco em minimizar o erro entre a variável de processo e o setpoint. O objetivo principal é avaliar a viabilidade e o desempenho da abordagem neural em comparação ao controle tradicional, explorando seus benefícios em termos de estabilidade, resposta temporal e robustez frente a perturbações.

## 2. FUNDAMENTAÇÃO TEÓRICA

### 2.1 Controle PID

O controle Proporcional-Integral-Derivativo (PID) é, segundo Ogata (2010), o método de controle mais amplamente utilizado na indústria devido à sua simplicidade de projeto e eficiência em sistemas lineares. Ele baseia sua ação no erro entre a variável de processo e o setpoint, aplicando três ações corretivas: proporcional (P), integral (I) e derivativa (D). A ação proporcional responde à magnitude do erro atual, a integral corrige desvios persistentes acumulando o erro ao longo do tempo, e a derivativa antecipa o comportamento futuro com base na taxa de variação do erro.

A equação geral do controlador PID contínuo é expressa por:

$$\begin{aligned} u(t) &= K_p \cdot e(t) + K_i \cdot \int e(t) dt + K_d \cdot \frac{de(t)}{dt} \\ &= K_{pe}(t) + K_i \cdot \int e(t) dt + K_d \cdot \frac{d}{dt} u(t) \\ &= K_p \cdot e(t) + K_i \cdot \int e(t) dt + K_d \cdot \frac{de(t)}{dt} \end{aligned}$$

onde  $u(t)$  é a saída de controle,  $e(t)$  é o erro, e os ganhos proporcional, integral e derivativo são respectivamente os ganhos  $K_p$ ,  $K_i$  e  $K_d$ .

Contudo, conforme observado por Dorf and Bishop (2022), controladores PID tradicionais têm desempenho limitado em sistemas com não linearidades, atrasos de transporte ou dinâmicas mal definidas, sendo necessário recorrer a métodos adaptativos ou inteligência computacional para garantir robustez em condições variáveis de operação.

### 2.2 Redes Neurais Artificiais

As redes neurais artificiais (RNAs) são descritas por Haykin (2001) como sistemas de processamento paralelo distribuído, compostos por unidades simples de processamento (neurônios) que operam em conjunto para resolver tarefas específicas. A capacidade de aprender padrões não lineares a partir de exemplos torna as RNAs particularmente úteis em controle de processos com comportamento dinâmico complexo ou difícil modelagem analítica.

Cada neurônio realiza uma combinação linear ponderada das entradas seguida por uma função de ativação não linear. O processo de treinamento ajusta os pesos sinápticos por algoritmos como o backpropagation, com o objetivo de minimizar o erro entre a saída desejada e a resposta da rede.

No contexto de controle, uma RNA pode atuar como um controlador de substituição, aprendendo o mapeamento entre erro e ação de controle a partir de dados históricos do sistema, mesmo sem conhecimento explícito do modelo da planta.

### 2.3 Controle Inteligente com Redes Neurais

Segundo Ogata (2010), métodos inteligentes como redes neurais são recomendados quando o sistema apresenta características não lineares, incertezas estruturais ou perturbações não modeladas. O uso de RNAs como controladores permite que o sistema aprenda com o histórico do processo, mesmo em malha fechada, e gere comandos de controle com base em padrões temporais de erro.

Uma abordagem comum é a implementação de uma rede perceptron multicamada (MLP), com neurônios ocultos que capturam relações não lineares entre entradas (como erro e variáveis passadas) e a variável manipulada. A RNA pode ser executada em tempo real, substituindo ou complementando o controle PID tradicional, com vantagem adicional de adaptabilidade sem reparametrização manual Haykin (2001).

### 2.4 Comunicação OPC (OLE for Process Control)

O padrão OPC DA (Data Access), conforme especificado pela OPC Foundation (2014), define uma interface padronizada para troca de dados em tempo real entre sistemas de automação e softwares de supervisão, promovendo interoperabilidade entre dispositivos de diferentes fabricantes.

O OPC DA opera segundo uma arquitetura cliente-servidor, onde os servidores disponibilizam variáveis (tags) de entrada e saída de dispositivos industriais. No presente trabalho, o MATLAB atuou como cliente OPC, realizando leitura e escrita de dados com a planta SMAR via os servidores Smar.DfiOleServer.0 e Smar.DF65Server.1. Essa comunicação permitiu integrar a lógica de controle neural ao ambiente real da planta, com atualização periódica das variáveis envolvidas.

### 2.5 Características da Planta SMAR (System302)

A planta utilizada no experimento é baseada no sistema System302, da empresa SMAR, uma arquitetura de controle distribuído (DCS) que utiliza o protocolo digital Foundation Fieldbus H1 como padrão de comunicação entre dispositivos de campo Industriais (2018). A plataforma é amplamente utilizada para treinamento em universidades e aplicações

industriais de pequeno porte.

A planta é composta por transmissores digitais, válvulas com posicionadores, controladores configuráveis e servidores OPC, possibilitando a simulação realista de malhas de controle de nível, temperatura e vazão. Seu ambiente modular e programável é ideal para testes de algoritmos avançados de controle, como redes neurais, integrando dispositivos de campo reais com sistemas computacionais externos via protocolos industriais.

### 3. METODOLOGIA

#### 3.1 Arquitetura do Sistema

O sistema proposto foi desenvolvido com o objetivo de controlar uma variável de processo da planta SMAR utilizando uma rede neural artificial implementada no ambiente MATLAB. A comunicação entre o MATLAB e a planta foi estabelecida via protocolo OPC DA, garantindo a leitura e escrita das variáveis de processo em tempo real. A arquitetura geral inclui:

- A planta SMAR (System302), operando com dispositivos Foundation Fieldbus;
- O servidor OPC DA (Smar DFI Foundation (2023)), disponibilizando as variáveis em rede;
- O ambiente MathWorks (2024), 2008a v7.6, responsável por executar o algoritmo de controle neural e supervisionar o processo.

Essa estrutura permite o controle direto da planta com base nas decisões tomadas pela rede neural, sem intervenção do controle PID interno da planta.

#### 3.2 Coleta e Pré-processamento dos Dados

Para o treinamento da rede, foi realizada uma coleta de dados reais da planta operando sob controle PID convencional. As variáveis adquiridas incluem: setpoint (SP), variável de processo (PV) e variável manipulada (VM). Os dados foram amostrados a cada dois segundos e armazenados em arquivos .csv para posterior uso no treinamento.

Antes do treinamento, os dados passaram por etapas de normalização e análise de consistência. Foram utilizados três lags temporais das variáveis SP, PV e erro ( $e = SP - PV$ ), compondo o vetor de entrada da rede neural.

#### 3.3 Controle de Temperatura com Rede Neural

##### 3.3.1 Aquisição e Construção do Dataset

A coleta de dados foi realizada com a planta SMAR operando sob controle PID previamente sintonizado para estabilizar a variável de temperatura. O procedimento consistiu na aplicação incremental de setpoints e monitoramento do comportamento dinâmico da planta até que dois ciclos de sobre-sinal fossem completados, garantindo que a planta atingisse estabilidade antes de alterar o setpoint novamente. O objetivo foi capturar a resposta transiente completa para cada degrau, a fim de mapear com fidelidade a dinâmica térmica do sistema.

A coleta foi realizada utilizando um script MATLAB conectado via OPC DA ao sistema SMAR. Foram registradas as seguintes variáveis:

- Setpoint da malha de temperatura (Ty31Sp);
- Variável de processo (Tit31);
- Variável manipulada (Ty31Ao);
- Carimbo temporal (time).

A frequência de amostragem foi de 1 Hz, totalizando aproximadamente 2.700 amostras. Os dados foram salvos em formato .mat para posterior processamento.

##### 3.3.2 Pós-processamento e Preparação dos Dados

Com o objetivo de preparar os dados para o treinamento da rede neural, foi desenvolvido um script de pós-processamento no ambiente MATLAB, compatível com a versão 2008a. O script realiza as seguintes etapas:

- Conversão das variáveis brutas em vetores coluna;
- Cálculo de variáveis derivadas, incluindo:

- Erro ( $e = SP - PV$ );
- Derivada do erro, obtida por diferenças finitas centradas;
- Integral do erro, obtida por soma acumulada ponderada pelo tempo;
- DeltaMV, correspondente à variação da saída do controlador entre amostras consecutivas.

Essas variáveis refletem não apenas o estado atual da planta, mas também seu comportamento histórico, permitindo à rede neural aprender a evolução temporal do processo.

### 3.3.3 Inserção de Histórico Temporal (3 Lags)

Para incorporar memória temporal ao modelo, foram utilizados três lags de histórico (valores passados) de cada variável derivada. Isso resultou em um vetor de entrada com 16 características por amostra, compostas por:

- $Erro_{(t-0)}$ ,  $Erro_{(t-1)}$ ,  $Erro_{(t-2)}$ ,  $Erro_{(t-3)}$  ;
- Variável manipulada (Ty31Ao);
- Carimbo temporal (time).

O vetor de saída da rede foi definido como a variável de processo (PV) ajustada para o mesmo alinhamento temporal, garantindo coerência entre entradas e saída.

A base final foi exportada como arquivo .csv, contendo as variáveis de entrada (com os lags) e a saída esperada da rede (TIT31).

datasetposprocessadocom3lagsTemp															
Erro_10	DerivadaEr...	IntegralEr...	DeltaMV_10	Erro_11	DerivadaEr...	IntegralEr...	DeltaMV_11	Erro_12	DerivadaEr...	IntegralEr...	DeltaMV_12	Erro_13	DerivadaEr...	IntegralEr...	DeltaMV_13
Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number
Erro_1-0	DerivadaEr...	IntegralEr...	DeltaMV_1-0	Erro_1-1	DerivadaEr...	IntegralEr...	DeltaMV_1-1	Erro_1-2	DerivadaEr...	IntegralEr...	DeltaMV_1-2	Erro_1-3	DerivadaEr...	IntegralEr...	DeltaMV_1-3
Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number
1	-0.8170490...	0.00305203...	-3.2952451...	0	-0.8201007...	0.00598198...	-2.4782692...	0	-0.8260822...	0.00622614...	-1.6582417...	0	-0.8323078...	0	-0.8322333...
2	-0.8156127...	0.00143636...	-4.1107850...	0	-0.8170490...	0.00305203...	-3.2952451...	0	-0.8201007...	0.00598198...	-2.4782692...	0	-0.8260822...	0.00622614...	-1.6582417...
3	-0.8103332...	0.00528001...	-4.9210458...	0	-0.8156127...	0.00143636...	-4.1107850...	0	-0.8170490...	0.00305203...	-3.2952451...	0	-0.8201007...	0.00598198...	-2.4782692...
4	-0.8044414...	0.00589232...	-5.7254153...	0	-0.8103332...	0.00528001...	-4.9210458...	0	-0.8156127...	0.00143636...	-4.1107850...	0	-0.8170490...	0.00305203...	-3.2952451...
5	-0.8017272...	0.00271439...	-6.5270709...	0	-0.8044414...	0.00589232...	-5.7254153...	0	-0.8103332...	0.00528001...	-4.9210458...	0	-0.8156127...	0.00143636...	-4.1107850...
6	-0.7968463...	0.00488134...	-7.3238461...	0	-0.8017272...	0.00271439...	-6.5270709...	0	-0.8044414...	0.00589232...	-5.7254153...	0	-0.8103332...	0.00528001...	-4.9210458...
7	-0.7968463...	0.00488134...	-7.3238461...	0	-0.7968463...	0.00488134...	-7.3238461...	0	-0.8017272...	0.00271439...	-6.5270709...	0	-0.8044414...	0.00589232...	-5.7254153...
8	-0.7862243...	0.01062297...	-8.1100001...	0	-0.7862243...	0.01062297...	-8.1100001...	0	-0.7968463...	0.00488134...	-7.3238461...	0	-0.8017272...	0.00271439...	-6.5270709...
9	-0.7788391...	0.00738591...	-8.8887696...	0	-0.7788391...	0.00738591...	-8.8887696...	0	-0.7862243...	0.01062297...	-8.1100001...	0	-0.7968463...	0.00488134...	-7.3238461...
10	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7788391...	0.00738591...	-8.8887696...	0	-0.7862243...	0.01062297...	-8.1100001...
11	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7788391...	0.00738591...	-8.8887696...
12	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...
13	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...
14	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...
15	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...
16	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...
17	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...
18	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...
19	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...
20	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...
21	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...
22	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...
23	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...
24	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...
25	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...	0	-0.7749023...	0.00393711...	-9.6636027...

Figure 1. Arquivo exportado após o pós-processamento do dataset.

## 3.4 Controle de Nível com Rede Neural

### 3.4.1 Aquisição e Construção do Dataset

A coleta de dados da variável de nível foi realizada com a planta SMAR operando sob controle PID sintonizado, de forma semelhante ao experimento de temperatura. Durante o procedimento, o setpoint da malha de nível (Fy32Sp) foi alterado manualmente em degraus sucessivos, respeitando um tempo de estabilização suficiente para garantir a resposta completa do sistema. A variável de processo (Tit32) foi monitorada continuamente, permitindo o mapeamento da resposta dinâmica da planta.

Os dados foram adquiridos via protocolo OPC DA e registrados em arquivos .mat, contendo as variáveis de setpoint, variável de processo e carimbo de tempo. A frequência de amostragem foi de 1 Hz, totalizando aproximadamente 2.700 amostras.

### 3.4.2 Pós-processamento e Preparação dos Dados

O tratamento dos dados foi realizado em MATLAB com foco na simplicidade e compatibilidade com o ambiente operacional da planta. O vetor de saída da rede foi definido como a variável de processo ( $T_{t+32}$ ), previamente normalizada para o intervalo  $[0,1]$ , compatível com a função de ativação logística (logsig) utilizada na camada de saída da rede.

O conjunto de entradas foi composto por:

- $y_k$ : Valor da variável de processo no instante anterior;
- $y_k$ : Valor da variável de processo dois instantes anteriores ;
- Erro atual: Diferença entre o setpoint (SP) e a variável de processo (PV) no instante atual .

Essa estrutura resulta em três entradas por amostra, refletindo o histórico da planta e a diferença em relação ao valor desejado. A estrutura simples foi adotada propositalmente para facilitar o treinamento e execução da rede em ambiente MATLAB 2008a, com menor demanda computacional.

### 3.4.3 Montagem do Dataset Final

Após o cálculo das entradas, as linhas iniciais foram descartadas para eliminar valores inválidos resultantes da falta de histórico. A matriz final, composta por  $[y_k, y_k, \text{erro}, y_k]$ , foi salva em arquivo .mat para posterior uso no processo de treinamento supervisionado. Os limites máximos e mínimos da variável original foram armazenados para possibilitar a reversão da normalização durante os testes em tempo real.

## 3.5 Estrutura e Treinamento da Rede Neural

O modelo de controle neural utilizado neste trabalho foi baseado em uma rede neural artificial do tipo perceptron multicamada (MLP), implementada no ambiente MATLAB R2008a de forma totalmente manual, utilizando apenas operações matriciais e funções básicas de ativação. Essa abordagem garante compatibilidade com ambientes industriais legados e permite controle total sobre o processo de aprendizado.

A rede foi estruturada com:

- 3 neurônios de entrada, correspondentes a:
  - A variável de processo no instante anterior ( $y_k$ )
  - A variável de processo dois instantes anteriores ( $y_k$ )
  - O erro atual ( $SP - PV$ )
- 1 camada oculta com 10 neurônios e função de ativação tangente hiperbólica (tansig);
- 1 neurônio de saída, com função de ativação sigmoide (logsig), cuja saída representa a próxima estimativa da variável de processo, já normalizada entre 0 e 1.

Os pesos foram inicializados com valores aleatórios pequenos e controlados por semente fixa ( $\text{rand('seed',1)}$ ) para garantir reprodutibilidade. O treinamento foi conduzido pelo algoritmo de retropropagação do erro (backpropagation) com taxa de aprendizado constante de 0,01, durante 3.000 épocas. O erro utilizado como critério de otimização foi o erro quadrático médio (MSE) entre a saída desejada (PV normalizada) e a saída prevista pela rede.

A cada época, a rede realiza:

- Propagação direta (forward pass) das entradas;
- Cálculo do erro e derivadas parciais nas camadas de saída e oculta;
- Atualização dos pesos e bias com base nos gradientes obtidos.

Ao final do treinamento, os pesos finais ( $V, V_o, W, W_o$ ) e os parâmetros de normalização da variável de processo ( $PV_{\min}, PV_{\max}$ ) foram salvos para uso em tempo real no sistema de controle e são mostrados na Figura 2.

pesos_rede_simplificada.mat (MAT-file)		
	Name	Value
	PV_max	49.6507
	PV_min	25.4529
	V	10x3 double
	Vo	[1.5075;0.9909;3.0044;-4....
	W	[-0.3060;-0.2510;-0.5383;...
	Wo	0.8013

Figure 2. Pesos e bias resultantes.

A Figura 3 apresenta a curva de evolução do erro MSE ao longo das épocas de treinamento, demonstrando a convergência da rede e a redução progressiva do erro. A Figura 4 mostra a predição da saída após o treinamento.

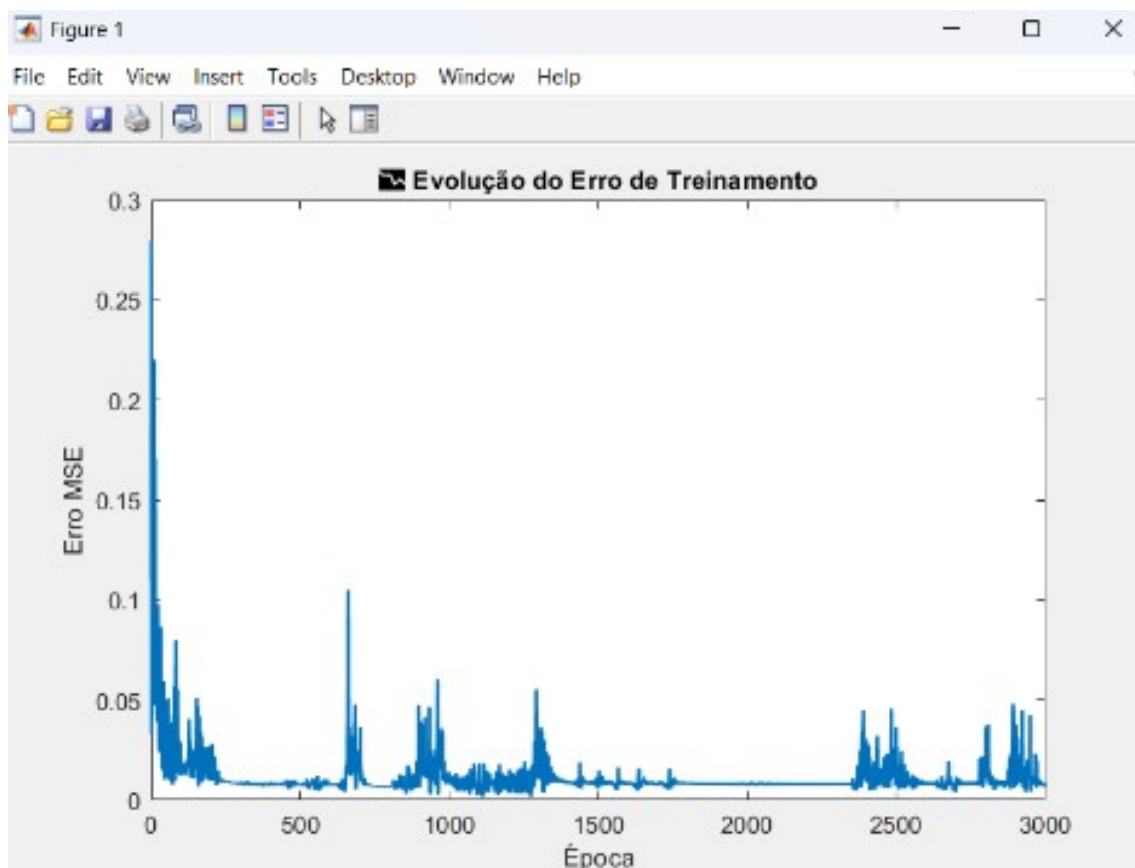


Figure 3. Evolução do erro de treinamento.

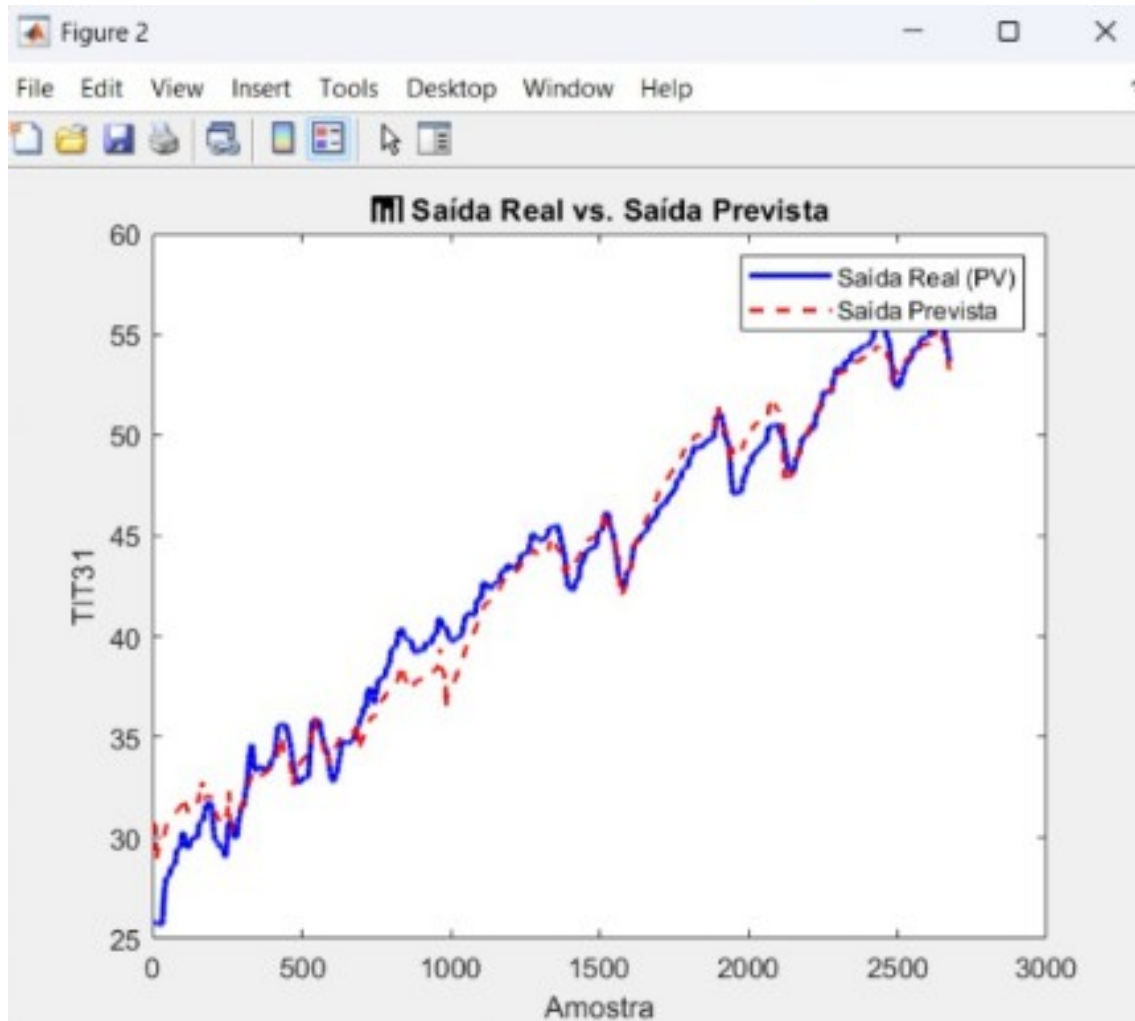


Figure 4. Predição da resposta da rede neural comparada ao sinal coletado do PID.

### 3.6 Implementação do Controle em Tempo Real

Com a rede treinada, o modelo foi integrado ao loop de controle em tempo real. A cada ciclo de controle (a cada 2 segundos), o MATLAB lê as variáveis atuais da planta via OPC, processa os dados com base nos lags históricos e fornece uma nova ação de controle calculada pela rede neural.

Essa ação de controle é então escrita de volta na variável manipulada (VM) da planta, substituindo completamente a ação do PID tradicional. O processo de leitura, inferência e escrita é realizado de forma contínua por meio de uma interface gráfica desenvolvida em GUIDE (MATLAB), garantindo supervisão e comandos manuais, quando necessário.

### 3.7 Estrutura da Interface Gráfica

A interface gráfica foi implementada com o objetivo de facilitar a interação entre o operador e o sistema de controle neural. Ela permite:

- Conexão com o servidor OPC;
- Visualização em tempo real de SP, PV e VM;
- Ativação/desativação do controle neural;
- Registro dos dados do processo para futuras análises.

Essa interface serviu também como ferramenta para validação prática da estabilidade e resposta do sistema em diferentes condições operacionais.



## 4. RESULTADOS

A validação prática do sistema de controle neural foi realizada em duas malhas distintas da planta SMAR System302: controle de temperatura e controle de nível. Em ambas as aplicações, o controlador PID tradicional foi utilizado como referência inicial para estabilizar a planta e gerar a base de dados para o treinamento da rede neural.

Após o treinamento, a rede foi inserida no loop de controle em tempo real, substituindo o PID e sendo responsável pelo envio direto da variável manipulada à planta via OPC DA.

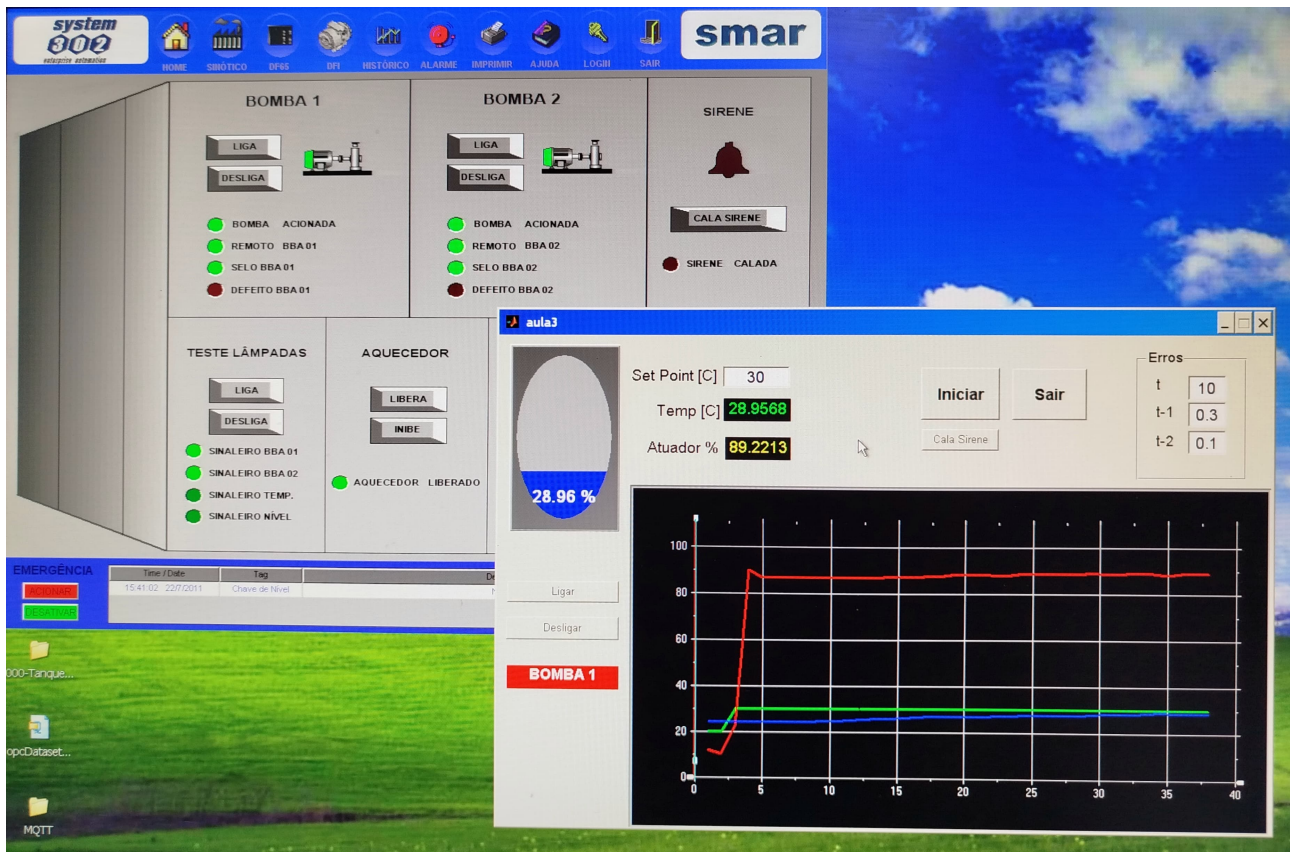


Figure 5. Implementação do controle neural.

### 4.1 Resultados do Controle de Temperatura

O controle de temperatura foi conduzido aplicando degraus manuais de setpoint em diversos níveis, aguardando o regime permanente para cada patamar.

A rede neural, treinada previamente com dados normalizados, mostrou-se capaz de reproduzir a ação de controle com comportamento semelhante ao PID.

Durante os testes, observou-se que:

- A rede foi capaz de acompanhar os degraus de referência com estabilidade, sem oscilações significativas;
- O tempo de resposta foi ligeiramente maior que o do PID, mas com menor sobre-sinal;
- O controle neural apresentou robustez frente a pequenas perturbações, como variações de carga simuladas na planta.



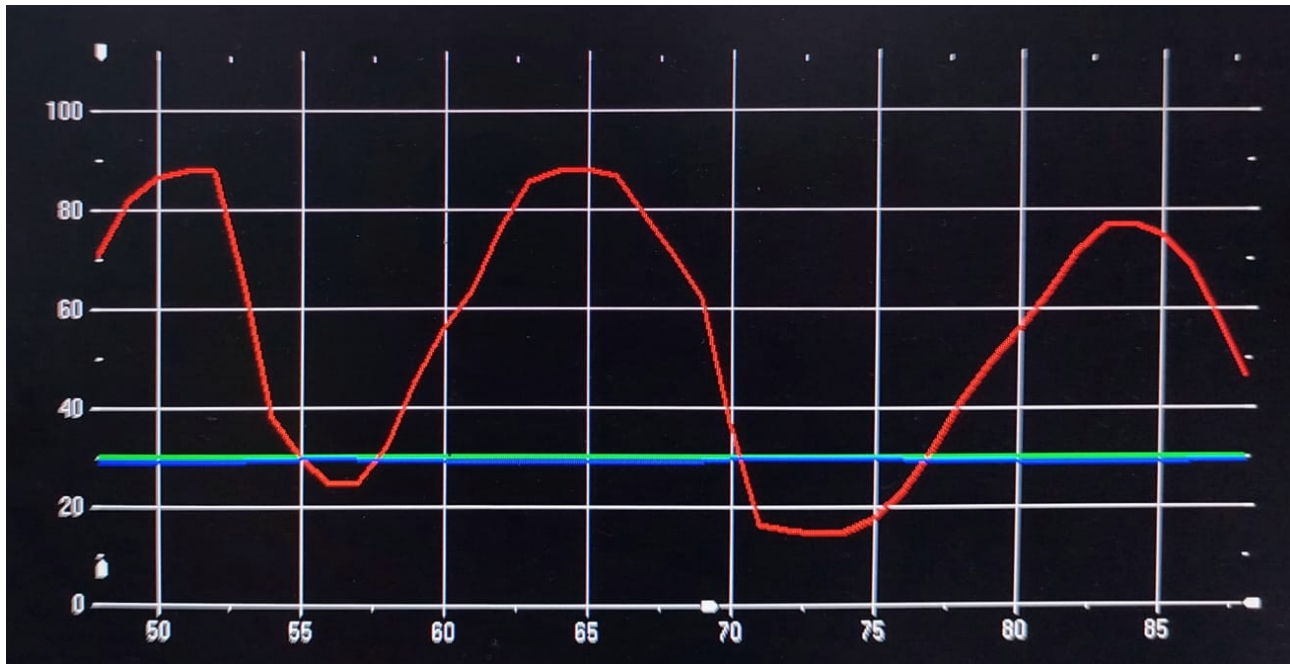


Figure 6. Resposta do controle neural.

#### 4.2 Resultados do Controle de Nível

Para a malha de nível, os testes foram conduzidos de forma semelhante. O setpoint foi alterado em degraus discretos, respeitando o tempo de estabilização entre as etapas. A rede neural utilizada nesta aplicação foi mais simples em termos de entrada (3 variáveis apenas), mas ainda assim obteve resultados relevantes.

Os principais resultados observados foram:

- A rede acompanhou o comportamento da planta com estabilidade, mesmo utilizando apenas dois valores passados e o erro como entradas;
- Houve menor precisão nos níveis de regime permanente em comparação ao PID, mas o comportamento geral foi estável;
- A rede respondeu corretamente a variações rápidas, mostrando capacidade de generalização.

#### 4.3 Discussão Comparativa

Os testes indicam que redes neurais treinadas com dados operacionais reais podem substituir o controle PID com desempenho comparável em plantas didáticas reais. Mesmo com estruturas simples e dados limitados, a rede mostrou-se capaz de generalizar a dinâmica das malhas de temperatura e nível. A diferença de desempenho entre as duas malhas está associada à complexidade da dinâmica térmica (mais lenta e previsível) em relação à dinâmica hidráulica (mais sensível e ruidosa).

#### 4.4 Limitações e Potenciais Melhorias

Embora os resultados obtidos com as redes neurais tenham sido promissores, algumas limitações técnicas foram observadas durante o desenvolvimento e testes do sistema, principalmente relacionadas à estrutura computacional e à integração em tempo real com a planta SMAR.

Uma das limitações está no uso de uma rede neural estática (do tipo MLP), que não incorpora estados internos ou memória dinâmica real, como ocorreria em arquiteturas recorrentes. Isso restringe a capacidade da rede de representar adequadamente fenômenos com dependência temporal mais prolongada, especialmente em malhas com retardo de transporte ou histerese.

Outro ponto de limitação está no processo de normalização e na dependência do intervalo de operação utilizado durante o treinamento. A rede foi treinada com um conjunto específico de dados normalizados, o que implica que variações fora desse intervalo podem comprometer a resposta da rede. Esse fator reduz a capacidade de generalização para faixas não observadas e exige atenção ao reuso da rede em novos contextos.

No aspecto da implementação prática, a execução contínua no MATLAB 2008a impõe restrições quanto ao uso de estruturas mais modernas, como treinamento online ou ajustes adaptativos durante a operação. Além disso, a ausência

de mecanismos de validação cruzada ou regularização pode favorecer o sobreajuste em situações de coleta com baixa variabilidade.

Como potenciais melhorias, destaca-se:

- A substituição da MLP por uma arquitetura recorrente (RNN) ou LSTM, que pode capturar dependências temporais mais profundas;
- A implementação de técnicas de validação cruzada durante o treinamento para melhorar a capacidade de generalização;
- A introdução de métodos de atualização online, permitindo que a rede aprenda continuamente com novos dados da planta;
- A adoção de uma camada de segurança ou supervisão fuzzy, para detectar desvios de operação e transições seguras entre controle neural e PID convencional;
- Por fim a migração do ambiente MATLAB para plataformas mais modernas, como Python com TensorFlow ou PyTorch, ou ainda implementações embarcadas com TinyML, visando melhor desempenho e portabilidade industrial.

## 5. CONCLUSÃO

Este trabalho apresentou o desenvolvimento e a aplicação de um sistema de controle baseado em rede neural artificial para uma planta didática SMAR, utilizando comunicação OPC DA com integração ao ambiente MATLAB. A proposta consistiu em substituir o controle PID tradicional por um controlador inteligente, treinado com dados reais da planta e executado em tempo real por meio de uma interface gráfica interativa.

Os testes realizados demonstraram que a rede neural, mesmo com arquitetura simples, foi capaz de controlar eficientemente a variável de processo, mantendo o sistema estável e com desempenho comparável ao PID. A resposta ao degrau e os indicadores de erro mostraram que a abordagem neural pode atuar de forma confiável mesmo em planta real, sem necessidade de modelo matemático explícito.

Dentre os principais benefícios observados destacam-se: facilidade de adaptação do controle, robustez frente a pequenas variações operacionais e potencial de aplicação em ambientes industriais com plantas complexas ou mal modeladas.

Como continuidade, sugere-se o aprimoramento da arquitetura da rede, o uso de técnicas de atualização online e a aplicação em sistemas multivariáveis. Além disso, destaca-se a possibilidade de incorporar uma camada de conectividade via protocolo MQTT (Message Queuing Telemetry Transport), permitindo o envio das variáveis de processo utilizando o modelo publish/subscribe em um broker central, o que viabilizaria a criação de dashboards interativos com dados em tempo real no ambiente Node-RED, favorecendo o monitoramento remoto, a análise histórica e a escalabilidade da solução para aplicações reais em Indústria 4.0.

## 6. AGRADECIMENTOS

Os autores agradecem ao Departamento de Engenharia Mecânica da Universidade de Brasília (UnB) pelo suporte estrutural ao desenvolvimento do projeto, bem como ao Programa de Pós-Graduação em Sistemas Mecatrônicos pela formação técnica e científica proporcionada ao longo do curso.

Agradecimentos especiais ao Prof. Dr. Alberto Alvares, pela orientação e incentivo à pesquisa aplicada em controle inteligente, e ao Laboratório de Automação e Controle (GRACO), onde se encontra instalada a planta SMAR utilizada neste trabalho, pelas condições experimentais e infraestrutura disponibilizadas durante as aulas práticas.

## 7. REFERÊNCIAS

- Dorf, R.C. and Bishop, R.H., 2022. *Modern Control Systems*. Pearson, 14th edition. ISBN 9780137307240.
- Foundation, O., 2023. "Opc unified architecture specification".
- Haykin, S., 2001. *Redes Neurais: Princípios e Prática*. Bookman Editora, Porto Alegre.
- Industriais, S.E., 2018. "System302 - manual técnico". <https://www.smar.com/br/system302-manual>.
- MathWorks, 2024. "Matlab documentation". <https://www.mathworks.com/help/matlab/>.
- Ogata, K., 2010. *Modern Control Engineering*. Prentice Hall, 5th edition. ISBN 9780136156733.

## 8. AVISO DE RESPONSABILIDADE

Os autores são os únicos responsáveis pelo material impresso incluso nesse artigo.