

Sistema de temperatura com lâmpada incandescente e sensor: projeto de implementação em Arduino IDE e Python

Temperature control system with incandescent lamp and sensor: design and implementation in Arduino IDE and Python

João Vitor Hoffmann Spode¹, Nicolas Araujo Paladino do Campo¹,
André Francisco Caldeira¹, Mathias Verдум de Almeida¹,
Arthur Sandri Lunkes¹, Charles Rech¹, Simone Ferigolo Venturini¹,
Cristiano Frandalozo Maidana¹

¹Universidade Federal de Santa Maria, Cachoeira do Sul, RS, Brasil

RESUMO

Este trabalho apresenta o projeto e a implementação de um sistema de controle de temperatura utilizando uma lâmpada incandescente de 12 V utilizada como atuador térmico e um sensor LM35 para realimentação, conectados a um microcontrolador Arduino Uno. A planta térmica foi identificada por meio de um ensaio de resposta ao degrau, e os parâmetros de um modelo de primeira ordem com tempo morto ganho estático K, tempo morto θ e constante de tempo τ foram estimados pelo método da Curva de Reação de Ziegler-Nichols. Com base no modelo identificado, seis métodos de sintonia PI foram avaliados: CHR (com e sem sobressinal), Cohen-Coon, 3C, IAE e ITAE. Foram realizadas simulações comparativas, seguidas de validação experimental com os três métodos de melhor desempenho. Nas simulações, os métodos Cohen-Coon e 3C apresentaram respostas mais agressivas, com maior sobressinal e saturação do sinal de controle, enquanto as sintonias CHR e IAE exibiram comportamento mais estável. Na validação experimental, o método IAE apresentou o melhor compromisso entre rapidez, suavidade do sinal PWM e estabilidade, mostrando-se o mais adequado para operação contínua da planta didática.

Palavras-chave: Controle de temperatura; Controlador PI; Arduino; Identificação de sistemas; Modelo FOPDT

ABSTRACT

This work presents the design and implementation of a temperature control system using an 12V incandescent lamp used as a thermal actuator as a heating actuator and an LM35 sensor for feedback, connected to an Arduino Uno microcontroller. The thermal plant was identified through a step response test, and the parameters of a first-order model with dead time (FOPDT) — static gain K, dead time θ ,



and time constant τ — were estimated using the Ziegler–Nichols reaction curve method. Based on the identified model, six PI tuning methods were evaluated: CHR (with and without overshoot), Cohen–Coon, 3C, IAE, and ITAE. Comparative simulations were performed, followed by experimental validation of the three best-performing methods. In the simulations, the Cohen–Coon and 3C methods presented more detrimental responses, with greater overshoot and saturation of the control signal, while the CHR and IAE tunings exhibited more stable behavior. In the experimental validation, the IAE method presented the best compromise between speed, PWM signal advances, and stability, proving to be the most suitable for continuous operation of the didactic plant.

Keywords: Temperature control; PI controller; Arduino; System identification; FOPDT model

1 INTRODUÇÃO

O controle de processos térmicos é uma área fundamental da engenharia de controle, com ampla aplicação industrial e educacional. Sistemas de controle de temperatura são particularmente adequados para o estudo de controladores PI, pois apresentam dinâmica lenta, resposta monotônica e características que os aproximam de modelos de primeira ordem com atraso (FOPDT) ^{[1][2]}. Essas propriedades permitem a aplicação direta de métodos clássicos de sintonia, como o de Ziegler–Nichols e suas variações ^{[3][4]}.

Neste contexto, este trabalho apresenta o desenvolvimento de uma planta de temperatura de bancada utilizando um Arduino Uno, uma lâmpada incandescente de 12 V como atuador, o sensor LM35 para medição da temperatura e um transistor TIP122 como driver de potência ^[5]. A planta foi identificada experimentalmente por meio de um ensaio de resposta ao degrau, e os parâmetros do modelo FOPDT foram estimados automaticamente por um script Python desenvolvido para esse fim.

Diferentemente de abordagens meramente demonstrativas, este trabalho propõe uma comparação estruturada entre diferentes métodos clássicos de sintonia PI aplicados a uma planta térmica de bancada identificada experimentalmente. A principal contribuição do estudo está em integrar identificação automática em Python, simulação comparativa e validação em hardware real, produzindo uma



plataforma de baixo custo com potencial didático para o ensino de modelagem e controle de processos térmicos.

2 MODELO DA PLANTA

A planta utilizada foi disponibilizada pelo professor da disciplina. Nela foi utilizado um Arduino Uno responsável por gerar o sinal PWM (Modulação por Largura de Pulso), que atua como variável manipulada do sistema. O PWM controla a potência fornecida à lâmpada por meio do chaveamento rápido de um transistor TIP122. O TIP122 funciona como driver, permitindo que o Arduino controle a lâmpada de 12 V sem sobrecarregar suas saídas^[5]. Para garantir a comutação adequada do transistor em frequências inferiores a 10 kHz, foi empregado um resistor de 1 k Ω na base do TIP122. A lâmpada incandescente de 12 V atua como fonte de aquecimento e converte a potência elétrica modulada em energia térmica.

Como sensor de temperatura foi utilizado o LM35, que possui saída linear de 10 mV por grau Celsius e precisão típica de $\pm 0,5^{\circ}\text{C}$. Ele fornece o sinal de realimentação necessário para medir a temperatura instantânea do processo e permitir que o Arduino ajuste a potência aplicada à lâmpada conforme o objetivo de controle. A leitura analógica do LM35 é processada pelo ADC (Conversor Analógico-Digital) de 10 bits do Arduino, permitindo a reconstrução da temperatura medida^[5].

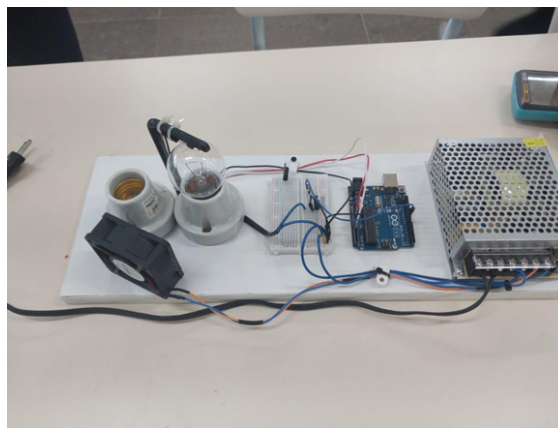
O comportamento térmico da planta foi aproximado por um modelo de primeira ordem com tempo morto (FOPDT), hipótese amplamente utilizada em processos térmicos com dinâmica dominante lenta^{[2][6]}. Essa escolha se justifica pelo perfil monotônico da resposta ao degrau e pela necessidade de um modelo simples, porém suficientemente representativo, para fins de sintonia e comparação entre controladores. Ainda assim, reconhece-se que a planta real pode apresentar não linearidades, saturação e variações paramétricas associadas às condições ambientais. de primeira ordem, pois a lâmpada aquece o ambiente gradualmente e a temperatura se eleva com uma constante de tempo dependente da massa térmica da



região aquecida e das perdas naturais para o meio externo^{[7][8]}. A taxa de amostragem escolhida foi de 10 Hz, equivalente a uma leitura a cada 100 milissegundos. O Arduino realiza a leitura do sensor e envia os dados via comunicação serial UART a 9600 baud, permitindo supervisão e registro em Python^[9]. A faixa de controle utilizada no PWM é de 0 a 255, correspondente à resolução de 8 bits.

Figura 1

Foto da planta de temperatura



Para introduzir perturbações externas controladas, foi empregado um ventilador axial de 12 V direcionado à região aquecida, promovendo aumento da convecção e redução da temperatura local^{[7][8]}.

3 OBTENÇÃO DOS MODELOS DE CONTROLE

A identificação do modelo da planta térmica foi realizada por meio de um ensaio experimental de resposta ao degrau. No Arduino, foi implementado um modo específico para identificação, acionado via comando serial "MODE:IDENT". Nesse modo, o microcontrolador aplica automaticamente o degrau completo na saída PWM e envia, a cada 100 milissegundos, a temperatura lida pelo LM35. Em paralelo, o script Python recebe esses dados pela porta serial, armazena as amostras e finaliza a coleta assim que o Arduino envia o sinal "IDENT_DONE". O ensaio de identificação foi realizado a partir da condição ambiente, com aplicação de um degrau máximo na variável manipulada

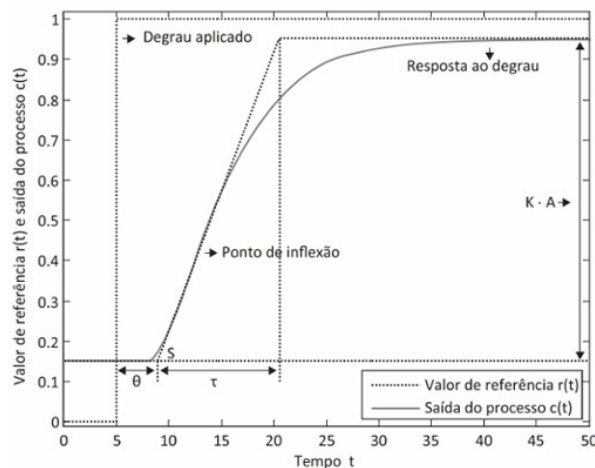


(PWM = 255). A aquisição ocorreu a 10 Hz durante aproximadamente 350 s, intervalo no qual a resposta térmica atingiu variação suficientemente pequena entre amostras consecutivas, caracterizando regime quase permanente para fins de identificação. Para aumentar a confiabilidade do procedimento, recomenda-se explicitar no texto a temperatura inicial média, as condições ambientais do ensaio e, quando possível, a repetição do experimento em múltiplas tomadas.

O Método de Ziegler–Nichols baseado na curva de reação é uma técnica fundamental para sintonia de controladores PI em malha aberta^[3]. Consiste em aplicar um degrau unitário na entrada do sistema e observar sua resposta sigmoidal. A partir dessa curva, extraem-se três parâmetros essenciais: θ (tempo morto), que representa o atraso inicial da resposta; τ (constante de tempo), que caracteriza a velocidade de alcance do equilíbrio; e K (ganho estático), que corresponde à razão $\Delta Y/\Delta U$ entre saída e entrada em regime permanente, conforme ilustrado na Figura 2.

Figura 2

Exemplo esquemático de degrau



Fonte: Ogata^[3]

Durante a coleta, as medições do LM35 apresentam ruído inerente ao ADC do Arduino. Para contornar esse problema sem distorcer a dinâmica original, foi aplicado um filtro EMA (Média Móvel Exponencial) com coeficiente $\alpha = 0,05$, resultando em uma



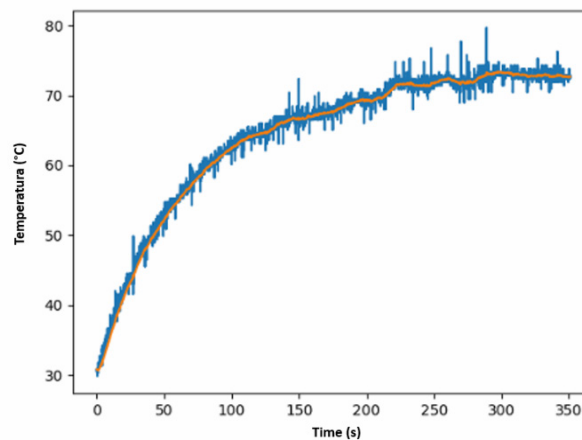
curva suavizada que preserva a tendência real do aquecimento. A Figura 3 apresenta simultaneamente o sinal bruto e o sinal filtrado obtidos durante o ensaio. A equação geral utilizada para o filtro EMA é apresentada na Equação 1:

$$yf[i] = \alpha y[i] + (1 - \alpha)yf[i - 1] \quad \text{Eq. (1)}$$

Onde yf é o valor filtrado no instante atual, y é a amostra bruta lida no mesmo instante e $yf[i - 1]$ é o valor filtrado no instante anterior. O parâmetro $\alpha = 0,05$ controla o peso dado à amostra atual em relação ao histórico.

Figura 3

Resposta ao degrau com sinal bruto (azul) e filtrado (laranja)



A partir do sinal filtrado, procedeu-se à identificação dos parâmetros da planta térmica utilizando o método da curva de reação de Ziegler–Nichols em malha aberta^[3]. O ganho estático foi calculado conforme a Equação 2:

$$K = \frac{\Delta T}{\Delta u} = \frac{(T_{\infty} - T_0)}{\Delta u} \quad \text{Eq. (2)}$$

Onde T_{∞} é a temperatura em regime permanente, T_0 é a temperatura inicial do sistema e para evitar ambiguidade, neste trabalho a variável manipulada foi



normalizada em relação ao intervalo do PWM, de modo que Δu foi considerado na escala digital aplicada ao Arduino. Consequentemente, o ganho estático K é interpretado em termos de variação de temperatura por variação da entrada normalizada. Alternativamente, caso se deseje interpretação física em volts, deve-se explicitar a conversão entre duty cycle, tensão média aplicada e potência térmica dissipada.. O tempo morto θ foi obtido pela extrapolação da tangente no ponto de maior inclinação da curva filtrada, conforme a Equação 3:

$$\theta = \frac{t_{inf} - (T_{inf} - T_0)}{\left(\frac{dT}{dt}\right)_{máx}} \quad \text{Eq. (3)}$$

A constante de tempo τ foi determinada pelo método dos 63,2%, conforme a Equação 4:

$$T(\theta + \tau) = T_0 + 0,632 \cdot (T_{\infty} - T_0) \quad \text{Eq. (4)}$$

Com esses parâmetros, o sistema foi representado por um modelo FOPDT (Primeira Ordem com Tempo Morto), descrito pela Equação 5:

$$G(s) = \frac{K}{(\tau s + 1)} \cdot e^{-\theta s} \quad \text{Eq. (5)}$$

Os valores obtidos experimentalmente foram $K = 0,183257$, $\theta = 2,012241$ s e $\tau = 64,17306$ s. Esse modelo apresentou boa correspondência com o comportamento observado fisicamente, sendo utilizado como base para todos os métodos de sintonia aplicados na Seção 4.

4 MÉTODOS DE SINTONIA

Com o modelo de primeira ordem com atraso obtido experimentalmente, foram aplicados seis métodos clássicos de sintonia de controladores PI, todos implementados



por funções no script Python desenvolvido. Cada método gera valores específicos de (ganho proporcional) e (tempo integral) a partir dos parâmetros identificados.

4.1 Método CHR (Chien, Hrones e Reswick)

O método CHR ^{[1][4]} define duas sintonias distintas: uma voltada para eliminar sobressinal e outra permitindo cerca de 20% de sobressinal. A sintonia CHR sem sobressinal utiliza as relações da Equação 6:

$$K_c = \frac{0,35 \cdot \tau}{(K \cdot \theta)} ; T_i = 1,16 \cdot \tau \quad \text{Eq. (6)}$$

Na sintonia CHR com 20% de sobressinal, os parâmetros são calculados conforme a Equação 7:

$$K_c = \frac{0,6 \cdot \tau}{(K \cdot \theta)} ; T_i = \tau \quad \text{Eq. (7)}$$

Essa variação admite um sobressinal moderado para acelerar a resposta do sistema. O ganho proporcional é aumentado em relação ao caso sem sobressinal, reduzindo o tempo de acomodação, enquanto o tempo integral é mantido igual à constante de tempo da planta.

4.2 Método Cohen-Coon

O método de Cohen-Coon [2] é particularmente adequado para processos com tempo morto significativo. Os parâmetros do controlador PI são obtidos conforme a Equação 8:

$$K_c = \left(\frac{1}{K}\right) \cdot \left[0,9 \cdot \left(\frac{\tau}{\theta}\right) + \frac{1}{12}\right] ; T_i = 3 \cdot \theta \cdot \left[\frac{(10 + \frac{\theta}{\tau})}{(9 + 20 \cdot \frac{\theta}{\tau})}\right] \quad \text{Eq. (8)}$$



O método Cohen-Coon ajusta τ e θ de forma agressiva quando a razão é significativa, gerando resposta mais rápida ao custo de maior oscilação.

4.3 Método 3C

O método 3C [1, 10] utiliza ajustes empíricos para sistemas de primeira ordem com atraso, buscando equilíbrio entre velocidade e estabilidade. A relação utilizada é apresentada na Equação 9:

$$K_c = \left(\frac{0,928}{K} \right) \cdot \left(\frac{\tau}{\theta} \right)^{0,946} ; T_i = (0,928) \cdot \tau \cdot \left(\frac{\tau}{\theta} \right)^{0,583} \quad \text{Eq. (9)}$$

4.4 Métodos Baseados em Critérios de Erro Integrado (IAE e ITAE)

Os métodos IAE (Integrated Absolute Error) e ITAE (Integral of Time multiplied Absolute Error) visam minimizar métricas de erro acumulado da resposta do sistema [4][10]. As equações utilizadas são apresentadas nas Equações 10 e 11, respectivamente:

$$K_c = \left(\frac{0,758}{K} \right) \cdot \left(\frac{\theta}{\tau} \right)^{-0,861} ; T_i = \frac{\tau}{[1,02 - 0,323 \cdot \left(\frac{\theta}{\tau} \right)]} \quad \text{Eq. (10)}$$

$$K_c = \left(\frac{0,586}{K} \right) \cdot \left(\frac{\theta}{\tau} \right)^{-0,916} ; T_i = \frac{\tau}{[1,03 - 0,165 \cdot \left(\frac{\theta}{\tau} \right)]} \quad \text{Eq. (11)}$$

No caso do critério IAE, a sintonia privilegia respostas suaves com boa rejeição a perturbações. Já o critério ITAE penaliza erros persistentes, favorecendo transições mais limpas e redução de oscilações tardias.

4.5 Ganhos Obtidos pelos Métodos de Sintonia

Após a aplicação das equações apresentadas para cada método, os ganhos proporcional e integral foram calculados automaticamente pelo script Python,



utilizando os parâmetros da planta identificados na Seção 3. A Tabela 1 apresenta os valores obtidos para cada uma das sintonias consideradas.

As expressões clássicas de sintonia forneceram inicialmente os parâmetros K_c e T_i . Para implementação computacional do controlador PI na forma paralela, foi adotada a relação $K_i = K_c/T_i$, mantendo-se $K_d = 0$ por se tratar de um controlador sem ação derivativa. Assim, a Tabela 1 apresenta os ganhos na forma efetivamente utilizada no algoritmo embarcado.

Tabela 1

Ganhos obtidos pelos métodos de sintonia

Método	K_c	K_i	K_d
CHR (sem sobressinal)	60,9072	0,8182	0
CHR (20% sobressinal)	104,4124	1,6271	0
Cohen-Coon	157,0733	24,9712	0
3C	133,9525	16,9316	0
IAE	81,5194	1,2829	0
ITAE	76,2414	1,2176	0

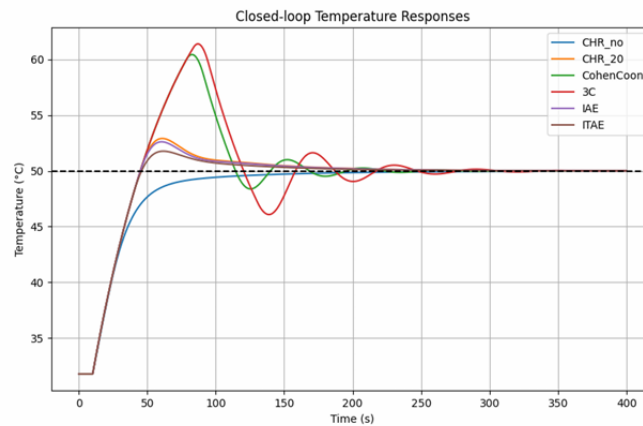
5 SIMULAÇÕES COMPARATIVAS DOS MÉTODOS DE SINTONIA

Com os parâmetros identificados e os ganhos calculados para cada método, foi realizada uma simulação completa das respostas em malha fechada utilizando o modelo FOPDT obtido experimentalmente. As simulações em malha fechada foram realizadas em ambiente Python considerando controlador PI discreto com período de amostragem de 0,1 s, saturação do sinal de controle no intervalo PWM $\square$ [0,255] e referência em degrau para 50 °C. Recomenda-se explicitar no manuscrito o método de discretização empregado, a forma de representação do tempo morto e a eventual adoção de mecanismo de anti-windup, pois tais escolhas afetam diretamente o comportamento transitório e o esforço de controle.



Figura 4

Respostas de cada método para temperatura em malha fechada

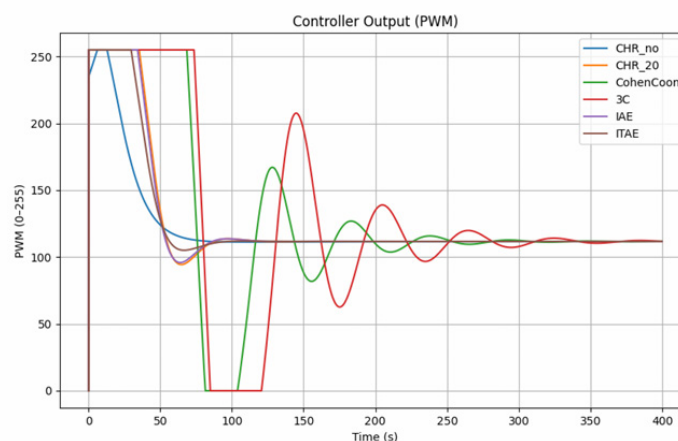


Os métodos Cohen-Coon e 3C produzem respostas mais agressivas, caracterizadas por sobressinais elevados e oscilações perceptíveis antes da estabilização. O método CHR com 20% de sobressinal apresenta comportamento intermediário. Os métodos IAE e ITAE apresentam respostas mais suaves e com oscilações reduzidas, refletindo sua orientação voltada para minimizar erros acumulados ao longo do tempo. O método CHR sem sobressinal mantém o comportamento mais conservador, oferecendo estabilidade e baixa ultrapassagem.

A análise do esforço de controle aplicado pelo controlador é apresentada na Figura 5.

Figura 5

Respostas de cada método para PWM em malha fechada





Os métodos Cohen-Coon e 3C geram sinais PWM com amplitudes elevadas e variações rápidas, incluindo períodos prolongados em saturação. Os métodos IAE e ITAE apresentam sinais significativamente mais estáveis, com transições suaves e esforço moderado. Os resultados das simulações permitiram concluir que métodos agressivos alcançam tempos de resposta menores, porém com maior risco de oscilação e saturação, enquanto métodos mais conservadores oferecem maior estabilidade. Essas observações foram essenciais para selecionar quais controladores seriam levados a teste na planta real.

6 IMPLEMENTAÇÃO DOS CONTROLADORES NA PLANTA REAL

Após a avaliação das respostas simuladas, foram selecionados três métodos de sintonia para implementação física: CHR sem sobressinal, CHR com 20% de sobressinal e IAE. A escolha considerou tanto o desempenho obtido nas simulações quanto a estabilidade exigida pela planta térmica real. Os métodos 3C e Cohen-Coon, apesar de apresentarem respostas rápidas, exibiram oscilações pronunciadas e saturações extensas no sinal de controle, comportamento indesejável para um atuador térmico baseado em lâmpada incandescente.

A implementação experimental foi realizada em duas etapas. A primeira consistiu no carregamento dos ganhos de cada controlador no Arduino, utilizando um firmware dedicado projetado para execução autônoma e operação contínua. A segunda etapa envolveu o monitoramento em tempo real pelo script Python, responsável por exibir a evolução da temperatura, o sinal de controle e o setpoint, além de permitir ajustes dinâmicos da referência e comandos de habilitação ou desabilitação do sinal de aquecimento.

O firmware do Arduino contém a implementação direta de um controlador PI discreto em sua forma paralela, com taxa de amostragem de 10 Hz. O monitoramento em Python opera com uma interface gráfica construída com a biblioteca Tkinter, com atualização dos gráficos em tempo real via Matplotlib.



Incluir, para cada um dos três controladores implementados, ao menos um gráfico experimental de temperatura versus tempo e um gráfico do sinal PWM correspondente. Recomenda-se também a inclusão de uma tabela comparativa contendo métricas como tempo de subida, sobressinal percentual, erro estacionário, tempo de acomodação e valor máximo do PWM. Um texto possível para introduzir essa análise seria: 'A Tabela X resume os principais índices de desempenho obtidos experimentalmente, permitindo comparar de forma quantitativa a aderência entre simulação e planta real. O controlador CHR sem sobressinal apresentou a resposta mais estável, com sinal de controle suave. O método CHR com 20% de sobressinal exibiu comportamento intermediário, com um pequeno pico inicial seguido de estabilização relativamente rápida. O método IAE apresentou dinâmica rápida e comportamento suave, equilibrando velocidade e estabilidade de forma muito satisfatória, com boa rejeição de perturbações e sem saturação prolongada no sinal PWM.

Para avaliar qualitativamente a rejeição a perturbações, foi empregado um ventilador axial de 12 V direcionado à região aquecida, aumentando a convecção local e reduzindo a temperatura medida ^{[7][8]}. Recomenda-se explicitar no manuscrito o instante de aplicação da perturbação, sua duração e o critério utilizado para comparar a recuperação do sistema, de modo a transformar essa etapa em um teste experimental mais sistemático.

7 CONCLUSÕES

Este trabalho apresentou o projeto, a identificação e o controle de uma planta de temperatura de bancada com Arduino Uno e lâmpada incandescente. O modelo FOPDT identificado experimentalmente mostrou boa correspondência com o comportamento físico da planta, validando a abordagem baseada na Curva de Reação de Ziegler–Nichols automatizada em Python.

A comparação com resultados de um grupo anterior, que utilizou MATLAB/Simulink e obteve parâmetros distintos ($\theta = 0,778$ s, $\tau = 86,21$ s e $K = 0,766$), evidenciou



a forte dependência dos parâmetros de plantas térmicas em relação às condições ambientais e ao método de identificação empregado. Ambos os modelos são válidos dentro das condições específicas em que foram identificados.

O controlador sintonizado pelo método IAE foi selecionado para operação contínua durante a apresentação do trabalho na Feira de Ciências do campus de Cachoeira do Sul, por apresentar a melhor combinação entre rapidez, estabilidade e suavidade do sinal de controle. Os resultados obtidos indicam que a abordagem proposta é adequada para fins didáticos e para estudos introdutórios de identificação e controle de processos térmicos. Entre os métodos avaliados, a sintonia IAE apresentou, nas condições estudadas, o comportamento mais equilibrado entre rapidez de resposta, estabilidade e suavidade do sinal de controle. Trabalhos futuros podem ampliar a análise experimental com métricas quantitativas, repetição de ensaios e investigação de estratégias adicionais de controle e compensação de perturbações.

REFERÊNCIAS

- [1] Garcia, C. *Controle de Processos Industriais: Estratégias Convencionais*. São Paulo: Edgard Blücher, 2017.
- [2] Seborg, D. E., Edgar, T. F., Mellichamp, D. A., & Doyle III, F. J. *Process Dynamics and Control*. 4. ed. Hoboken: John Wiley & Sons, 2016.
- [3] Ogata, K. *Modern Control Engineering*. 5. ed. Upper Saddle River: Prentice Hall, 2010.
- [4] Åström, K. J., & Hägglund, T. *PID Controllers: Theory, Design, and Tuning*. 2. ed. Research Triangle Park: Instrument Society of America, 1995.
- [5] Blum, J. *Exploring Arduino: Tools and Techniques for Engineering Wizardry*. 2. ed. Indianapolis: Wiley, 2020.
- [6] Camacho, O., & Smith, C. A. Sliding mode control: an approach to regulate nonlinear chemical processes. *ISA Transactions*, v. 39, n. 2, p. 205–218, 2000.
- [7] Incropera, F. P., Dewitt, D. P., Bergman, T. L., & Lavine, A. S. *Fundamentals of Heat and Mass Transfer*. 6. ed. Hoboken: John Wiley & Sons, 2007.
- [8] Cengel, Y. A., & Ghajar, A. J. *Heat and Mass Transfer: Fundamentals and Applications*. 5. ed. New York: McGraw-Hill, 2015.



- [9] Monk, S. *Programming Arduino: Getting Started with Sketches*. 2. ed. New York: McGraw-Hill, 2016.
- [10] Smith, C. A., & Corripio, A. B. *Principles and Practice of Automatic Process Control*. 3. ed. Hoboken: John Wiley & Sons, 2005.

Contribuição de autores

1 – João Vitor Hoffmann Spode

Discente de Engenharia Mecânica, Universidade Federal de Santa Maria, Campus Cachoeira do Sul, Rio Grande do Sul, Brasil.

Email: spode.joao@acad.ufsm.br - : <https://orcid.org/0009-0002-3103-0870>

Contribuição: Investigação, Escrita – primeira redação.

2 – Nicolas Araujo Paladino do Campo

Discente de Engenharia Mecânica, Universidade Federal de Santa Maria, Campus Cachoeira do Sul, Rio Grande do Sul, Brasil.

Email: nicolas.campo@acad.ufsm.br - <https://orcid.org/0009-0008-7142-5803>

Contribuição: Investigação, Escrita – primeira redação.

3 – André Francisco Caldeira

Doutor em Engenharia (UFSC, em cotutela com a Université Grenoble Alpes), docente da Universidade Federal de Santa Maria, Campus Cachoeira do Sul, Rio Grande do Sul, Brasil.

Email: andre.caldeira@ufsm.br - <https://orcid.org/0000-0002-4939-2709>

Contribuição: Conceituação, Metodologia, Recursos, Supervisão.

4 – Mathias Verдум de Almeida

Discente do Curso de Engenharia Mecânica, Universidade Federal de Santa Maria, Campus Cachoeira do Sul, Rio Grande do Sul, Brasil.

Email: mathias.verdum@acad.ufsm.br - <https://orcid.org/0009-0002-5371-9441>

Contribuição: Investigação, Validação.

5 – Arthur Sandri Lunkes

Discente do Curso de Engenharia Mecânica, Universidade Federal de Santa Maria, Campus Cachoeira do Sul, Rio Grande do Sul, Brasil.

Email: arthur.lunkes@acad.ufsm.br - <https://orcid.org/0009-0005-7750-8421>

Contribuição: Investigação, Validação.

5 – Charles Rech

Doutor em Fenômenos de Transporte (UFRGS), docente da Universidade Federal de Santa Maria, Campus Cachoeira do Sul, Rio Grande do Sul, Brasil.

Email: charles.rech@ufsm.br - <https://orcid.org/0000-0001-8523-6300>

Contribuição: Metodologia, Recursos, Escrita – revisão e edição.



6 – Simone Ferigolo Venturini

Doutoranda no Programa de Pós-Graduação em Engenharia de Produção da UFSM – PPGEPP, Rio Grande do Sul, Brasil.

Email: sfventurini@gmail.com - <https://orcid.org/0000-0002-9439-0008>

Contribuição: Análise Formal, Escrita – revisão e edição.

7 – Cristiano Frandalozo Maidana

Doutor em Engenharia Mecânica (UFRGS), docente da Universidade Federal de Santa Maria, Campus Cachoeira do Sul, Rio Grande do Sul, Brasil.

Email: cristiano.maidana@ufsm.br - <https://orcid.org/0000-0003-3137-6177>

Contribuição: Supervisão.

Como citar este artigo

Spode, J. V. H., do Campo, N. A. P., Caldeira, A. F., de Almeida, M. V., Lunkes, A. S., Rech, C., Venturini, S. F., & Maidana, C. F. (2026). Sistema de temperatura com lâmpada incandescente e sensor: projeto de implementação em Arduino IDE e Python. *JESTA*. Cachoeira do Sul, (4) e95903, Disponível em: <https://periodicos.ufsm.br/JESTA/article/view/95903>. Acesso em: XX/XX/XXXX.