

Controladores em Tempo Real com o PYDAQ: A Ferramenta e Novos Resultados

Cayo R. Castoril * Vinícius S. Fernandes * Samir A. Martins *

* GCoM - Grupo de Controle e Modelagem, Departamento de Engenharia Elétrica, Universidade Federal de São João del-Rei, Praça Frei Orlando 170 - Centro, 36307-352. São João del-Rei, Minas Gerais, Brasil.

Abstract:

This work presents the implementation of PID controllers in PYDAQ. PYDAQ (Martins, 2023) is an open-source tool under continuous development, designed for the acquisition and processing of experimental signals using NIDAQ boards and Arduino. With an average of 556 monthly downloads, it is a Python-based open-source tool with various functionalities. Through this work, digital controllers can be studied, implemented, and developed using PYDAQ, combining theoretical investigations with the practical implementation of real-time controllers. The results demonstrate that the functionality implemented here is an important facilitator in engineering education and research practices involving active learning methodologies, enabling the virtualization of laboratories.

Resumo: Este trabalho apresenta a implementação de controladores PID no PYDAQ. O PYDAQ (Martins, 2023) é uma ferramenta de código aberto continuamente em desenvolvimento, projetada para aquisição e processamento de sinais experimentais utilizando placas NIDAQ e *Arduino*. Com uma média de 556 downloads mensais, é uma ferramenta *open-source* em Python com diversas funcionalidades. Com este trabalho, controladores digitais podem ser estudados, implementados e desenvolvidos por meio do PYDAQ, combinando investigações teóricas e implementação prática de controladores em tempo real. Os resultados provam que a funcionalidade aqui implementada é um importante facilitador em práticas de ensino e pesquisa em engenharia que envolvem metodologias ativas, possibilitando a virtualização de laboratórios.

Keywords: Digital Control; Discrete Control; Control Systems; PYDAQ; Python; Open Science; Open-Source; Virtual Environment

Palavras-chaves: Controle Digital; Controle Discreto; Sistemas de Controle; PYDAQ; Python; Open Science; Open-Source; Ambiente Virtual.

1. INTRODUÇÃO

O ensino de sistemas de controle é essencial na formação de engenheiros, permitindo que profissionais controlem processos físicos, conduzindo o seu comportamento para uma maneira desejada. Desde refrigeradores (Staff, 2009) até veículos autônomos e aeronaves (Sharkawy et al., 2015; Susanto et al., 2021), os sistemas de controle são amplamente utilizados. Dentre as opções de controle disponíveis, o controle proporcional, integral e derivativo (PID) tem sido uma alternativa de bastante adesão entre indústrias e pesquisadores nas últimas décadas (Atherton, 1996). Uma sintonia adequada viabiliza aplicações diversas, como o controle de motores brushless ou de temperatura (Romero et al., 2015; Wang et al., 2017). Para isso, métodos de sintonia foram desenvolvidos para obter uma estimativa inicial para as constantes PID, como o método de sintonia sugerido por Ziegler e Nichols no ano de 1942 (Dhanda and Niwas, 2015).

Embora mais de 90% dos controladores industriais sejam implementados baseados em algoritmos de controle PID (Ang et al., 2005), seu ensino e aplicação em cursos de

engenharia enfrentam barreiras, como a necessidade de softwares que requerem licenças de alto custo e de difícil acesso. Softwares como *LabVIEW*[®] e *MATLAB*[®] são comuns no ensino e pesquisa (Cao et al., 2023; Hammoodi et al., 2020), mas além do custo inicial, exigem licenças com atualizações pagas, tornando o acesso ainda mais restrito. Diante desse cenário, torna-se essencial buscar alternativas *open-source* para o ensino e a virtualização de laboratórios, democratizando o aprendizado e a experimentação na área.

Com o objetivo de criar um ambiente facilitador para contribuir com pesquisas e ensinamentos de controladores PID, este trabalho apresenta uma adição ao PYDAQ (Martins, 2023), uma ferramenta *open-source* para a aquisição e análise de dados. A proposta permite que estudantes e pesquisadores utilizem placas de aquisição, como o *Arduino* e placas *USB* da *National Instruments* (NIDAQ), para o controle de sistemas. Por meio de um laboratório virtual, é permitido que acadêmicos e cientistas compreendam fundamentos do controlador PID de maneira iterativa e sem se preocupar com custos financeiros associados a softwares. A ferramenta também implementa sintonias baseadas no mé-

tudo de Ziegler-Nichols a partir de uma resposta ao degrau do sistema. Além de contribuir com o ensino do controle de maneira totalmente gratuita, o ambiente projetado contribui para uma visualização analítica e monitoramento de sistemas em tempo real. Dessa forma, é promovido um aprendizado prático e eficiente sobre controladores PID.

O artigo inicia com esta seção para uma introdução ao tema, seguida de conceitos preliminares essenciais para a compreensão do trabalho. A terceira seção detalha a implementação do controle PID no PYDAQ, seguida dos resultados obtidos. Por fim, a última seção lista as referências bibliográficas.

2. CONCEITOS PRELIMINARES

2.1 PYDAQ

O PYDAQ (Martins, 2023) é uma ferramenta *open-source* em *Python* que permanece em contínuo desenvolvimento. Sua documentação pode ser acessada em seu repositório de documentação¹. Atualmente, o PYDAQ conta com funcionalidades de aquisição e envio de dados, identificação de modelos em tempo real utilizando modelos NARX polinomiais, utilizando as placas como o *Arduino* e *NIDAQ*. Juntamente com a criação de um ambiente que permite uma virtualização de laboratório, o PYDAQ se torna um facilitador para o ensino prático de controle, tornando possível a realização de experimentos sem a necessidade de um laboratório físico. A interface e todas as ferramentas implementadas no PYDAQ podem ser acessadas com poucas linhas de código, permitindo o seu uso de maneira instrutiva e prática. De acordo com dados acessados no repositório em 31 de janeiro de 2025, possui cerca de 556 downloads por mês, tornando a ciência aberta a todos que queiram usá-la.

O PYDAQ reconhece automaticamente os dispositivos conectados no computador para utilizar as placas de aquisição. Utilizando o *Arduino*, é necessário carregar um código específico no controlador para habilitar sua funcionalidade. Esse código pode ser encontrado no GitHub² do PYDAQ, configurando as portas selecionadas para a aquisição e envio de dados. Ao utilizar o *NIDAQ*, as portas de entradas e saídas são selecionadas na interface. Para definir a configuração do terminal, pode-se escolher entre diferencial (Diff), referência ao terra do sistema (RSE) ou diferença ao terra não compartilhada (NRSE).

2.2 O Ensino De Controle Na Engenharia

Sistemas de controles são amplamente estudados na engenharia, pois permitem regular processos físicos de forma automatizada. Para isso, sistemas de controle com malha fechada são essenciais para manter a variável de saída do sistema próxima ao valor desejado. Um sistema de controle com essa realimentação pode ser visualizado como um diagrama de blocos, ilustrado pela Figura 1.

O *setpoint* (R) representa o valor de referência do sistema. Esse valor é comparado com a saída medida pelo sensor (B), que monitora a variável do processo. O controlador

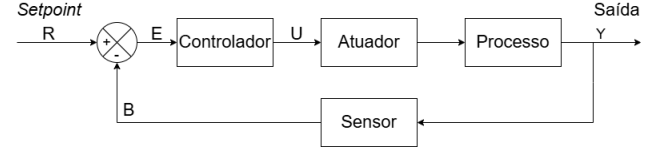


Figura 1. Diagrama de blocos de um sistema de controle em malha fechada.

calcula o erro $E = B - R$ e, com base nesse valor, gera um sinal de controle (U) para o atuador, que ajusta a operação do sistema. O sensor converte a grandeza física do sistema (Y), para uma unidade passível de processamento (B). A calibração é feita por funções polinomiais, conforme apresentada na equação (1). A função inversa da calibração segue a mesma estrutura polinomial e é utilizada para interpretar os sinais processados pelo usuário.

$$b(y) = a_n y^n + a_{n-1} y^{n-1} + \dots + a_1 y + a_0 \quad (1)$$

Nesta equação, $a, a_{n-1} \dots a_0$ são constantes da curva de calibração estática. y é a variável medida, $b(y)$ é a grandeza na unidade processável e n corresponde à ordem do polinômio.

Controlador PID O Controle PID gera um sinal de saída com base em constantes proporcional, derivativa e integral ao erro do valor do sistema em relação ao esperado. No domínio do tempo, a equação de um controlador clássico assume a seguinte forma:

$$u(t) = K_P e(t) + K_I \int_0^t e(t) dt + K_D \frac{d}{dt} e(t) \quad (2)$$

onde K_P , K_I e K_D são os parâmetros de controle proporcional, integral e derivativo. O $u(t)$ representa o sinal de controle emitido pelo controlador e o erro:

$$e(t) = r(t) - b(t)$$

pode ser obtido através do *setpoint* $r(t)$ selecionado com a saída do sistema $b(t)$. Para o processamento de dados digitais, é necessário trabalharmos com sinais discretizados. Dessa forma, conforme apresentado por Ogata (1995), o controlador clássico alterado para o domínio Z pode ser obtido pela equação a seguir:

$$u(z) = e(z) \left(K_P + \frac{K_I}{1 - Z^{-1}} + K_D (1 - Z^{-1}) \right) \quad (3)$$

A partir da equação no domínio Z , aplicando o método de diferença finita para conversão ao tempo discreto, conforme apresentado por Kim and Choi (2024), obtemos a equação computacional do controlador:

$$u(k) = K_P e(k) + K_I \sum_{i=0}^k e(i) T + K_D \frac{e(k) - e(k-1)}{T} \quad (4)$$

Onde a integral e a derivada podem ser representadas por uma soma acumulada dos erros e uma diferença dos valores consecutivos para cada instante de tempo k . O parâmetro T representa o período de tempo de cada amostra, que será o valor selecionado pelo usuário.

¹ Disponível em: www.pydaq.org

² Disponível em: www.github.com/samirmartins/pydaq

Sintonia dos parâmetros A Tabela 1 apresenta a metodologia para a obtenção de sintonia dos parâmetros segundo o método Ziegler-Nichols para uma obtenção inicial dos parâmetros K_p , K_i e K_d . Tendo o método como ponto de partida, uma sintonia fina pode ser realizada posteriormente a fim de otimizar os parâmetros obtidos.

Tabela 1. Parâmetros do controlador segundo Ziegler-Nichols.

Controlador	K_p	K_i	K_d
P	$\frac{T}{L}$	—	—
PI	$\frac{0,9T}{L}$	$\frac{0,3K_P}{L}$	—
PID	$\frac{1,2T}{L}$	$\frac{K_P}{2L}$	$0,5K_P L$

Dessa forma, os ganhos K_p , K_i e K_d do sistema podem ser obtidos em função de T e L , que representam o tempo morto (ou tempo de atraso) do sistema e o tempo em que o sistema leva para chegar em regime permanente.

3. METODOLOGIA

O desenvolvimento deste trabalho se baseia principalmente no uso de Python, no uso de ferramentas de código aberto e na implementação de um ambiente interativo que permite aos estudantes explorarem, simularem e testarem diferentes estratégias de controle.

3.1 Implementação da Interface Gráfica

A interface foi desenvolvida com PySide6 e QtDesigner, biblioteca e ferramenta gratuita que permite a criação de interfaces visuais intuitivas, facilitando a organização dos elementos visuais. A interface gerada é integrada ao código Python por meio de arquivos de interface do usuário (.ui), garantindo uma separação entre a lógica da aplicação e a apresentação visual.

O monitoramento em tempo real utiliza a ferramenta *animation* do *Matplotlib* para atualizar periodicamente os gráficos de *setpoint*, saída do sistema e erro. Durante a execução, os valores dessas variáveis, assim como a saída do controlador, são coletados em cada instante de tempo e podem ser salvos pelo usuário em arquivos (.dat) para posterior análise.

3.2 Implementação do Controlador PID Discreto

O controle PID discreto foi realizado empregando a equação (4), recebendo os parâmetros definidos na interface e retornando os valores de controle por meio de uma classe orientada a objetos.

Caso o usuário opte pela simulação, o controle é aplicado a uma função de transferência definida pelo usuário, e a resposta é obtida utilizando a função *signal.lsim* da biblioteca *SciPy*. Para a experimentação prática, o valor de controle calculado é enviado às portas de saída das placas de aquisição de dados, possibilitando o controle de processos reais. Esse recurso permite que estudantes desenvolvam a parte prática, empregando conhecimentos adquiridos em sala de aula em experimentos reais.

O *Arduino*, ferramenta de código aberto, permite aquisição de dados e controle via modulação por largura de pulso

(*PWM*), essencial para experimentos práticos. A relação entre a tensão de controle e a quantidade do ciclo do sinal em *PWM* é dada pela equação (5).

$$Ciclo_{PWM} = \frac{V_{controle}}{V_{máximo}} \times 255 \quad (5)$$

Dessa forma, o $Ciclo_{PWM}$ pode variar de 0 até 255, representando a razão cíclica do sinal modulado, que é a proporção entre o tempo em que o sinal está ativo (alto) e o período total do ciclo. $V_{controle}$ é a tensão de controle aplicada, e $V_{máximo}$ é a tensão máxima que o controlador pode fornecer (geralmente 5 V para placas *Arduino*, como o *Arduino Mega 2560*). A tensão eficaz enviada ao atuador é determinada pelo $Ciclo_{PWM}$.

As placas NIDAQ USB possibilitam maior precisão na aquisição de dados e no envio de sinais analógicos contínuos, permitindo a aplicação do controle PID sem a necessidade de conversão *PWM*. O PYDAQ conta com um reconhecimento automático dessas placas, facilitando a sua utilização em laboratórios de ensino.

3.3 Implementação da Sintonia Automática dos Parâmetros

A sintonia dos parâmetros do controlador pode ser obtida pelo método sugerido por Ziegler e Nichols, como uma ampliação da ferramenta Step Response³ do PYDAQ, a partir de uma resposta ao degrau do sistema. O ganho K é determinado com base na diferença entre os valores final e inicial normalizados. Para reduzir os efeitos de ruído na identificação dos parâmetros, os dados são suavizados por meio do filtro de Savitzky-Golay. Em seguida, traça-se uma reta tangente no ponto de inflexão da curva, permitindo a obtenção dos valores de tempo morto L e da constante de tempo T . Esses valores são utilizados para calcular os ganhos do controlador conforme a Tabela 1, de acordo com o método de sintonia escolhido pelo usuário.

3.4 Controle automático de diferentes processos

Para validar a metodologia proposta, foram implementados controles em diferentes processos. Neste trabalho, serão apresentados dois casos: um controle simulado de um sistema de primeira ordem e um controle aplicado a uma planta real de temperatura.

No ambiente simulado, o usuário pode definir a função de transferência de um sistema arbitrário, permitindo a experimentação com diferentes tipos de processos sem a necessidade de um hardware físico. Essa abordagem amplia as possibilidades de ensino, permitindo que estudantes compreendam a resposta de diferentes sistemas ao controle PID de maneira acessível e interativa.

Para o controle de uma planta real, utilizou-se um módulo composto por um elemento de aquecimento resistivo, um amplificador (*driver*) *PWM* tiristorizado para acionamento, um transdutor de temperatura a semicondutor e condicionadores de sinais (amplificadores e filtros) para o transdutor e driver. O driver transforma o sinal de controle em um sinal de potência para a lâmpada, permitindo

³ Documentação disponível em: www.pydaq.org/step_response_nidaq

a regulação da temperatura. A Figura 2 apresenta uma imagem do módulo utilizado no experimento.



Figura 2. Visão vertical do módulo de temperatura.

Antes da aplicação do controle, foi realizada uma calibração estática para garantir maior exatidão. O procedimento envolveu a aplicação de diferentes tensões ao módulo, com medição da tensão de saída e da temperatura resultante, obtida por um termopar ligado a um multímetro digital. Caso a curva de calibração já seja conhecida, o usuário pode inseri-la diretamente na interface gráfica ou coletar os dados com a ferramenta *Get Data*⁴, desenvolvida para aquisição no PYDAQ.

4. RESULTADOS

4.1 Interface Geral do Controle PID

A interface permite ajuste dos ganhos PID, setpoint e calibração dos sensores. A Figura 3 apresenta a interface desenvolvida contendo todas as opções de ajuste necessárias, ao selecionar a opção de controle por *Arduino* e *NIDAQ*.

A utilização dessas placas permite que alunos trabalhem com sistemas laboratoriais reais, desenvolvendo habilidades práticas que contribuirão para o entendimento de controladores. O suporte às diferentes placas reforça a versatilidade do PYDAQ como ferramenta educacional, possibilitando que estudantes tenham contato com opções de abordagens populares e de qualidade.

4.2 Simulação

i) ou controlar uma planta real; ii) ou simular uma planta controlada. A depender de sua escolha, a interface adapta ao seu caminho. Ao selecionar a opção de simulação, o usuário poderá informar os valores polinomiais que

⁴ Documentação disponível em: www.pydaq.org/get_data_nidaq/

Figura 3. Interface geral para o controle PID no PYDAQ utilizando *Arduino* e *NIDAQ*.

descrevem a dinâmica do sistema, como visto na interface apresentada na Figura 4.

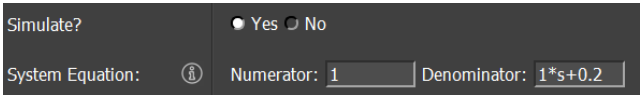


Figura 4. Dados para a simulação de controle de sistemas no PYDAQ.

A dinâmica de um controlador digital para o sistema selecionado pode ser monitorada pelo ambiente virtual apresentado na Figura 5.

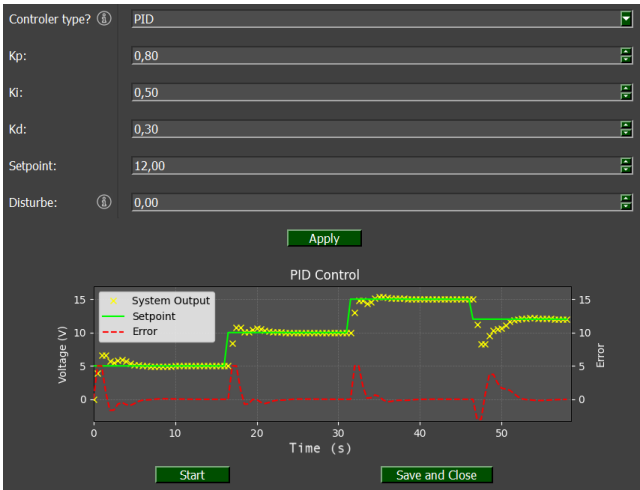


Figura 5. Ambiente virtual de um sistema simulado.

Dessa forma, comportamentos dinâmicos de diferentes controladores podem ser explorados e analisados. O uso de simulações e ajuste de controladores em tempo real proporciona uma experiência de ensino baseada na experimentação, alinhada com métodos ativos de ensino e aprendizagem em engenharia. Estudantes podem visualizar a resposta de sistemas a diferentes configurações de controle, facilitando a compreensão de conceitos como estabilidade, erro em regime permanente e tempo de resposta. Além disso, o erro entre o valor desejado e a saída real do sistema

é registrado continuamente e pode ser analisado posteriormente, permitindo uma avaliação mais aprofundada do desempenho dos controladores.

4.3 Sintonia de Parâmetros

Com a ampliação da ferramenta *Step Response* do PYDAQ, é permitida a sintonia de parâmetros por meio da interface apresentada na Figura 6.

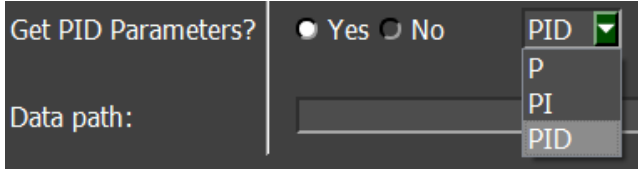


Figura 6. Opções para a sintonia de um controlador P, Pi ou PID.

No contexto educacional, o PYDAQ se apresenta como um facilitador desse processo, permitindo a análise da resposta ao degrau do sistema e a obtenção dos parâmetros iniciais do controlador, para sistemas com características exponenciais.

4.4 Resultados de um Controlador em um Módulo de Temperatura

Os dados estáticos adquiridos do módulo de temperatura descrito na seção 3.4 são apresentados nas Figuras 7 e 8.

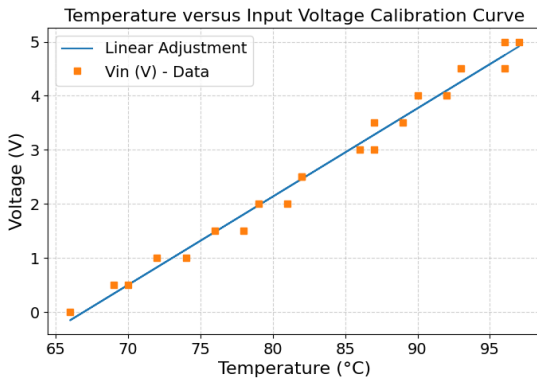


Figura 7. Dados e equação linear que relaciona tensão de entrada (V_{in}) à temperatura ($^{\circ}\text{C}$).

Os dados coletados permitiram estabelecer duas equações: uma relacionando a tensão de entrada emitida ao atuador à temperatura do sistema, e outra relacionando a temperatura do sistema à tensão de saída do sensor do módulo, cujas variáveis podem ser visualizadas no diagrama de blocos da Figura 1. As equações 6 e 7 apresentam as curvas de calibrações estáticas do sistema. Os parâmetros das retas foram estimados pelo método dos mínimos quadrados.

$$V_{in}(C) = 0,1506C - 9,7043 \quad (6)$$

$$C(V_{out}) = 23,5167V_{out} + 4,2400 \quad (7)$$

A taxa de correlação (R^2) indica o quão bem o modelo matemático se ajusta aos dados coletados. No caso das equações de calibração, os valores $R = 0,9526$ para $V_{in}(C)$ e

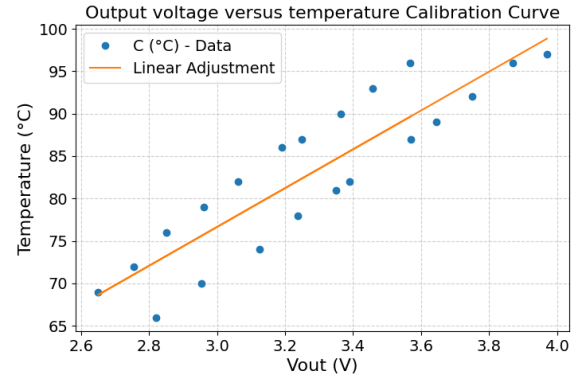


Figura 8. Dados e equação linear que relaciona a temperatura à tensão de saída.

$R = 0,7970$ para $C(V_{out})$ indicam que 95,2% e 79,7% da variação dos dados coletados podem ser explicados pela equação obtida. O modelo indica que a exatidão do controle pode garantir valores de até 79% dos valores de referência. Esses fatores também indicam que fatores como ruídos e não linearidade do módulo de temperatura podem impactar a exatidão do controle.

Uma vez que determinadas as curvas de calibração estáticas, uma sintonia para um controlador PID foi obtida utilizando o método de Ziegler e Nichols. Os ganhos foram calculados conforme apresentado na Tabela 1. A Figura 9 apresenta a resposta do sistema ao degrau e a curva tangente ao ponto de maior inflexão, utilizada para calcular os valores dos ganhos.

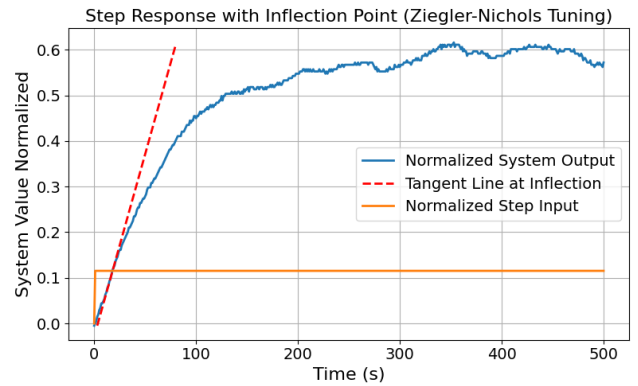


Figura 9. Resultados da resposta ao degrau para a sintonia de um controlador PID.

Os ganhos K_p , K_i e K_d obtidos para controlador PID foram de 3,01, 0,20 e 10,96. Utilizando esses valores, o sistema foi controlado via *Arduino* a um período de amostragem de 1 s, valor muito menor que a constante do sistema, que atinge o regime permanente em cerca de 500 segundos. A Figura 10 apresenta o *Arduino* em operação.

A resposta do sistema após alguns minutos de controle pode ser observada na Figura 11. Nota-se que o controle desenvolvido se mostrou eficiente para os setpoints iniciais e que, na última temperatura de referência, os parâmetros do controlador também foram variados, resultando em uma dinâmica mais oscilatória em comparação com o outro controlador. Em um ambiente acadêmico, a possibilidade de testar diferentes estratégias de controle em tempo real

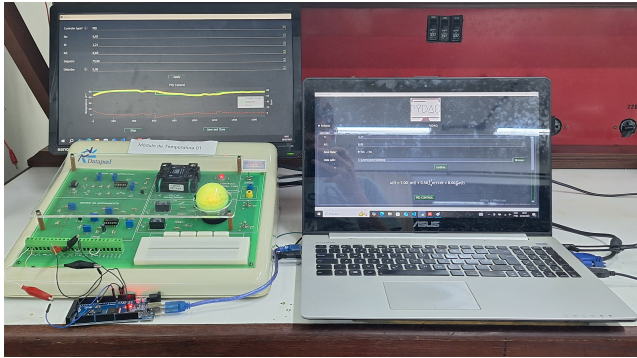


Figura 10. Monitoramento de um controlador PID aplicado ao módulo de temperatura.

e analisar suas dinâmicas é essencial para o aprendizado por meio de práticas laboratoriais.



Figura 11. Monitoramento de um controlador PID aplicado ao módulo de temperatura

O experimento demonstra como microcontroladores podem ser utilizados no ensino de controle em engenharia, permitindo a implementação de estratégias em processos físicos com ferramentas livres e hardware acessível. A possibilidade de testar diferentes ajustes de sintonias e observar a resposta do sistema em tempo real proporciona uma experiência interativa e exploratória, aproximando a teoria da prática.

5. CONCLUSÃO

Os resultados obtidos demonstraram a viabilidade do uso de microcontroladores e ferramentas de código aberto para o ensino de controle na engenharia. O sistema desenvolvido integra um controlador PID eficiente com uma interface gráfica intuitiva e funcional, permitindo ajustes de parâmetros em tempo real e a simulação de sistemas em um ambiente virtual. No entanto, algumas limitações ainda foram observadas, como dificuldades no processamento para o controle de sistemas com períodos de amostragem muito baixos, devido ao atraso na atualização nos gráficos de monitoramento.

O trabalho aqui desenvolvido e implementado no PYDAQ é de código aberto, pode ser encontrado temporariamente no GitHub do autor⁵ e está disponível a contribuições. Ele será incorporado ao repositório principal do PYDAQ⁶ na próxima data de lançamento de versão, conforme alinhado

⁵ Disponível em: <https://github.com/CayoRw/pydaq>

⁶ Disponível em: <https://github.com/samirmartins/pydaq>

com os desenvolvedores do projeto. Para trabalhos futuros, pretende-se otimizar o desempenho do sistema, expandir suas funcionalidades e possibilitar aplicações mais avançadas, como controles multivariáveis, para ampliar ainda mais o seu potencial como ferramenta de ensino e pesquisa.

AGRADECIMENTOS

Os autores agradecem o apoio da UFSJ e das agências CAPES, CNPq e FAPEMIG.

REFERÊNCIAS

- Ang, K.H., Chong, G., and Li, Y. (2005). PID control system analysis, design, and technology. *IEEE Trans. Control Syst. Technol.*, 13, 559–576. doi:10.1109/TCST.2005.847331.
- Atherton, D.P. (1996). Some advantages of the autotuning approach in PID control. In *IEE Colloquium on Getting the Best Out of PID in Machine Control*, volume 1996, 1–1. IEE.
- Cao, S., Zhao, W., and Zhu, A. (2023). Research on intervention PID control of VAV terminal based on LabVIEW. *Miscellaneous*, 45, 103002–103002. doi:10.1016/j.csite.2023.103002.
- Dhanda, A. and Niwas, D. (2015). Issue 01) publishing month. *Manag. Humanit. Soc. Sci. Paradigms (IJEMHS)*, 15, –. doi:10.21105/joss.05662.
- Hammoodi, S.J., Flayyih, K.S., and Hamad, A.R. (2020). Design and implementation speed control system of DC motor based on PID control and MATLAB SIMULINK. *Int. J. Power Electron. Drive Syst.*, 11, 127–134. doi:10.11591/ijpeds.v11.i1.pp127-134.
- Kim, E. and Choi, J.Y. (2024). Modified fast discrete-time PID formulas for obtaining double precision accuracy. *Electron. Lett.*, 60, –. doi:10.1049/ell2.70114.
- Martins, S.A.M. (2023). PYDAQ: Data acquisition and experimental analysis with Python. *J. Open Source Softw.*, 8, 5662. doi:10.21105/joss.05662.
- Ogata, K. (1995). *Discrete-time control systems*. Prentice-Hall, Inc.
- Romero, J.A., Sanchis, R., and Arrebola, E. (2015). Experimental study of event-based PID controllers with different sampling strategies. application to brushless DC motor networked control system. Technical report, Univ. Jaume I Campus del Riu Sec.
- Sharkawy, A.N., Ali, A.S., Ghazaly, N.M., Abdel-Jaber, G., El-Nasser, A.S., Ali, A.S., and el Jaber, G.T.A. (2015). PID controller of active suspension system for a quarter car model. *Int. J. Adv. Eng. Technol.*, 8, 899–909.
- Staff, I. (2009). *2009 5th International Colloquium on Signal Processing and Its Applications*. IEEE.
- Susanto, T., Setiawan, M.B., Jayadi, A., Rossi, F., Hamdhi, A., and Sembiring, J.P. (2021). Application of unmanned aircraft PID control system for roll, pitch and yaw stability on fixed wings. In *Proc. 2021 Int. Conf. Comput. Sci., Inf. Technol., Electr. Eng. (ICOMITEE)*, 186–190. IEEE. doi:10.1109/ICOMITEE53461.2021.9650314.
- Wang, Y., Zou, H., Tao, J., and Zhang, R. (2017). Predictive fuzzy PID control for temperature model of a heating furnace. In *2017 36th Chinese Control Conference (CCC)*, 4523–4527. IEEE.