UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

GUILHERME PIRES SILVA
RAFAELA CRISTINA FORTUNA DA SILVA
VITOR ESTEVÃO PARAGUASSU
WILLIAN LUIZ GIACOMITTI

Sistema de Controle Digital Aplicado a um Balancing Bot

Proposta para trabalho final de Controle B do Curso de Engenharia Mecatrônica da Universidade Tecnológica Federal do Paraná.

Professor: Carlos Raimundo Erig Lima.

CURITIBA
2025

SUMÁRIO

1. INTRODUÇÃO

O controle de sistemas instáveis constitui um dos temas centrais da engenharia de controle, sendo o pêndulo invertido um exemplo clássico amplamente utilizado para a análise e validação de técnicas modernas de controle. Esse tipo de sistema apresenta desafios significativos devido à sua instabilidade inerente, exigindo estratégias de controle capazes de garantir estabilidade, desempenho dinâmico adequado e robustez frente a perturbações externas (OGATA, 2010; DORF; BISHOP, 2016).

Nesse contexto, robôs móveis de duas rodas com autoequilíbrio, conhecidos como *balancing bots*, destacam-se como plataformas didáticas e experimentais relevantes. Esses sistemas permitem a integração de conceitos fundamentais como modelagem matemática, representação em espaço de estados, síntese de controladores, sensoriamento inercial e implementação de controle digital em tempo real. Além disso, sua dinâmica é frequentemente modelada de forma análoga ao pêndulo invertido, possibilitando a aplicação direta de métodos clássicos e modernos de controle (GRASSI; TSINIAS, 2005).

Este trabalho apresenta o desenvolvimento, a simulação e a validação experimental de uma malha de controle digital aplicada a um robô de autoequilíbrio. O sistema foi modelado matematicamente, linearizado em torno do ponto de equilíbrio e representado em espaço de estados, permitindo a análise do comportamento dinâmico e o projeto de controladores por diferentes estratégias. A implementação do controlador em plataforma embarcada possibilitou a avaliação prática do desempenho do sistema, evidenciando as diferenças entre os resultados obtidos por meio de simulações computacionais e aqueles observados em ensaios experimentais (FRANKLIN; POWELL; WORKMAN, 1998).

2. OBJETIVOS

Os objetivos deste trabalho são:

- Desenvolver e implementar uma malha de controle digital para estabilização de um robô de duas rodas;

- Descrever detalhadamente o hardware, projeto mecânico e arquitetura de software utilizados;
- Modelar matematicamente o sistema com base na dinâmica de um pêndulo invertido;
- Analisar o comportamento do sistema em malha aberta e com controlador por realimentação de estados;
- Validar experimentalmente o controlador projetado e discutir os resultados obtidos.

3. MATERIAIS E MÉTODOS

Nesta seção são apresentados os materiais, componentes e métodos empregados no desenvolvimento do robô de auto equilíbrio, bem como os critérios adotados para o projeto eletrônico e mecânico. São descritos os dispositivos de hardware utilizados, a arquitetura do sistema embarcado, o projeto estrutural do robô e as escolhas técnicas que influenciam diretamente o comportamento dinâmico do sistema.

3.1. PROJETO ELETRÔNICO

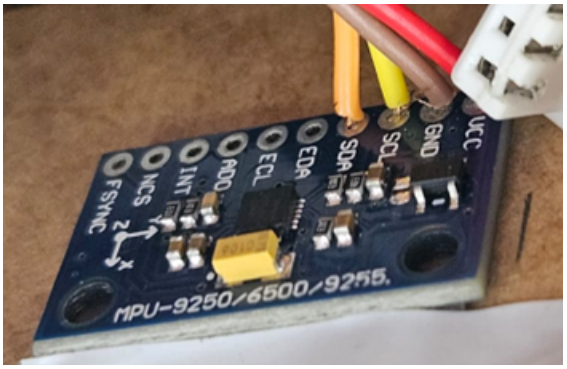
A eletrônica foi selecionada visando um equilíbrio entre alto poder de processamento para malhas de controle em tempo real e eficiência energética. Optou-se pelo microcontrolador ESP32 (Figura 1) como o cérebro do robô, pois diferente de microcontroladores de 8-bits (como o Arduino Uno), possui um processador dual-core de 32-bits de 240MHz, possibilitando a execução de várias tarefas ao mesmo tempo sem afetar o desempenho (ESPRESSIF SYSTEMS, 2023).

Figura 1: ESP32 devkit-c1

Fonte: Autores (2025).

Para a determinação da orientação do robô, foi utilizado o sensor MPU-9250 (Figura 2), pois este módulo comunica-se via protocolo I2C e possui 9 Graus de Liberdade (9-DOF), integrando acelerômetro, giroscópio e magnetômetro. A fusão dos dados do acelerômetro e giroscópio é fundamental para estimar o ângulo de inclinação do robô (pitch) com precisão e baixo ruído, permitindo a estabilização vertical. A utilização de sensores inerciais e técnicas de fusão sensorial é amplamente discutida na literatura (MADGWICK, 2010; INVENSENSE, 2016).

Figura 2: MPU-9250

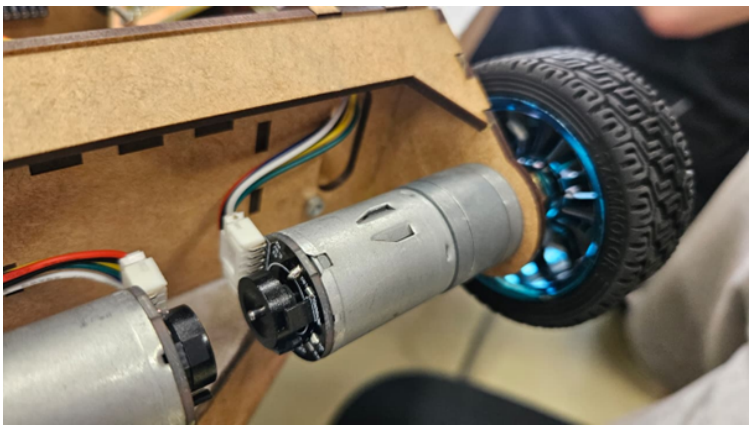


Fonte: Autores (2025).

A locomoção é realizada por dois motores DC de 6V com redução, apresentado pela Figura 3, operando a uma rotação nominal de 210 RPM. Cada motor está acoplado a um encoder de efeito Hall com resolução de 341 pulsos por revolução. A baixa rotação com caixa de redução oferece o torque necessário para vencer a inércia inicial. Os encoders são indispensáveis para fornecer *feedback* de

velocidade e posição ao microcontrolador, permitindo um controle de malha fechada preciso sobre a tração.

Figura 3: Motores DC 6V.



Fonte: Autores (2025).

Para o controle dos motores, utilizou-se o driver identificado no diagrama como TB66. O TB6612FNG é um driver baseado em MOSFETs, sendo muito mais eficiente que os drivers antigos baseados em transistores bipolares (como o L298N). Ele minimiza a queda de tensão e o aquecimento, aproveitando melhor a energia da bateria para os motores, além de suportar a lógica de 3.3V do ESP32 diretamente. (TEXAS INSTRUMENTS, 2017).

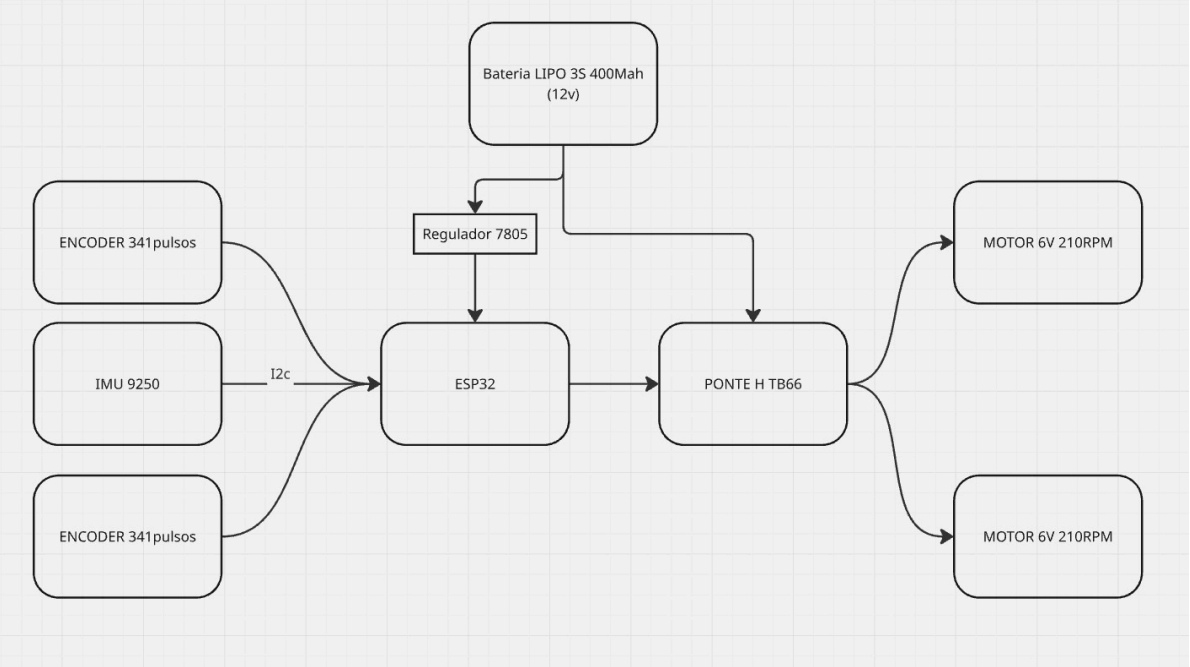
Utilizou-se uma bateria LiPo 3S 400mAh (12V nominais). Escolhida pela alta taxa de descarga e densidade energética leve. Em relação à regulação, um regulador de tensão linear 7805 converte os 12V da bateria para 5V estáveis, alimentando o ESP32 e a lógica dos sensores.

Figura 4: Bateria Lipo.

Fonte: Autores (2025).

A partir dessas definições, a Figura 5 demonstra o diagrama simplificado das conexões entre os componentes.

Figura 5: Diagrama simplificado de conexões do Robô.

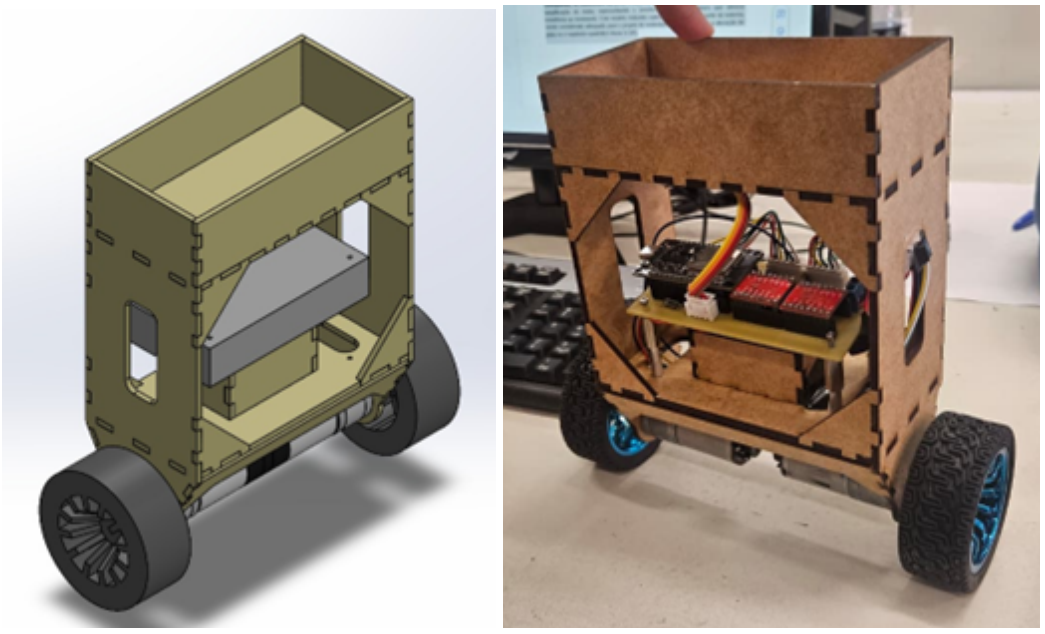


Fonte: Autores (2025).

3.2. PROJETO MECÂNICO

O design mecânico teve como prioridade a modularidade, facilidade de fabricação e a distribuição de massa adequada para a dinâmica do sistema, demonstrado pela Figura 6 abaixo.

Figura 6: Modelagem mecânica do robô.

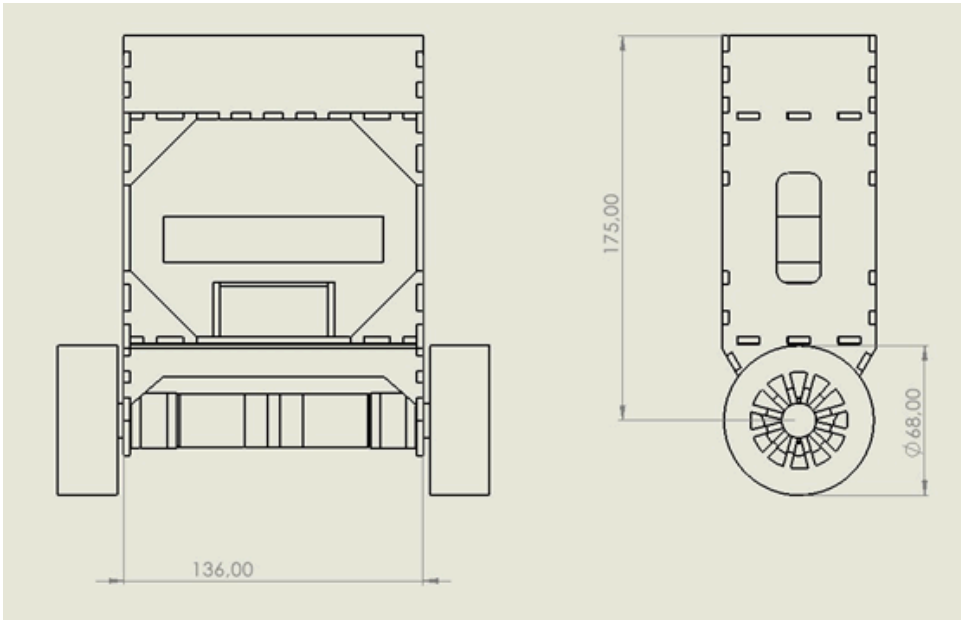


Fonte: Autores (2025).

A estrutura do chassi foi fabricada utilizando corte a laser. O chassi é composto por placas planas de MDF montadas através de encaixes do tipo "dedo" (finger joints) e parafusos passantes. O corte a laser permite prototipagem rápida com alta precisão dimensional (furos de 68mm para as rodas). O design "caixa aberta" facilita o acesso aos componentes eletrônicos para manutenção e passagem de cabos.

A partir desses parâmetros, tem-se o desenho técnico com as respectivas dimensões do robô, conforme a Figura 7.

Figura 7: Desenho técnico do robô.



Fonte: Autores (2025).

Em relação ao centro de massa (C.M.) a disposição vertical dos componentes foi feita para que certas configurações fossem obtidas. A bateria e os motores (componentes mais pesados) estão posicionados na parte inferior e média da estrutura, enquanto o microcontrolador e sensores ficam acima.

O design centraliza a massa no eixo vertical (Z), garantindo que o robô esteja equilibrado lateralmente, exigindo esforço dos motores apenas para correção frontal/traseira (pitch).

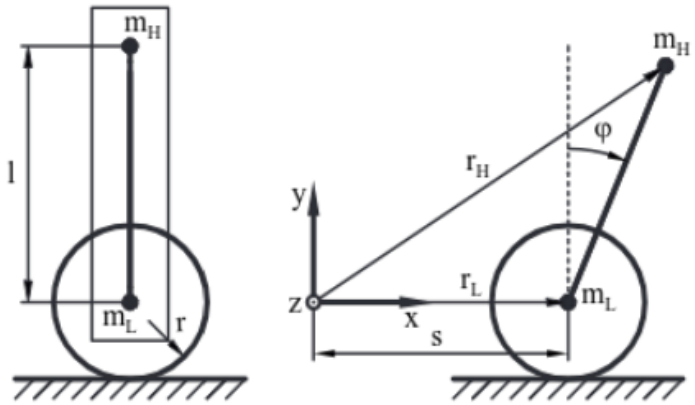
4. METODOLOGIA

O robô de autoequilíbrio é um sistema inerentemente instável, cujo objetivo de controle é manter o ângulo de inclinação próximo de zero (posição vertical), compensando perturbações externas e imperfeições mecânicas, característica típica de sistemas do tipo pêndulo invertido (OGATA, 2010; GRASSI; TSINIAS, 2005).

4.1. MODELAGEM MATEMÁTICA DO SISTEMA

O objetivo fundamental do controle é manter o corpo do robô na posição vertical (ângulo de inclinação φ=0°) enquanto gerencia a velocidade linear de deslocamento v. A estrutura física do sistema, bem como o diagrama de corpo livre considerando as forças atuantes, são apresentados na Figura 8.

Figura 8: Diagrama de corpo livre do sistema.



Fonte: MAHLER, B.; HAASE, J. (2013).

O modelo matemático utilizado baseia-se nas equações de movimento derivadas através da mecânica Lagrangiana, as quais consideram as energias cinética e potencial (translacional e rotacional) de todos os componentes do sistema. As equações encontram-se detalhadas no artigo de Mahler e Haase (2013). Para a simulação e o projeto do controlador, foram utilizados os parâmetros físicos mensurados do protótipo construído, conforme detalhado na Tabela 1 a seguir.

Tabela 1: Parâmetros reais do robô.

Parâmetro	**Símbolo**	**Valor**	**Unidade**
Dist. do C.M.	L	0,0438	m
Raio da roda	r	0,0335	m
Massa do corpo	mh	0,460	kg
Massa do par de rodas	ml	0,096	kg
Momento Iner. da roda	Jr	3,5344e-5	kg*m^2
Momento Iner. do corpo	Jp	70,08e-5	kg*m^2

Coeficiente de torque do motor	km	0,06	Nm/A
Resistência da armadura	Rm	2	ohm
Aceleração da gravidade	g	9,81	m/s^2

Fonte: Autores (2025).

Para o projeto do sistema de controle, optou-se pela representação em espaço de estados. Originalmente, o modelo completo seria de quarta ordem, incluindo a dinâmica da corrente elétrica. No entanto, adota-se a simplificação sugerida por Mahler e Haase (2013), assumindo que a constante de tempo elétrica é desprezível em comparação à mecânica (indutância *Lm* ≈0). Desta forma, a equação diferencial da corrente é substituída por uma relação algébrica baseada na Lei de Ohm, permitindo incorporar as características eletromecânicas, como o torque resistivo e a força contra-eletromotriz, diretamente na matriz dinâmica como termos de amortecimento viscoso.

Após a linearização em torno do ponto de operação vertical (φ=0°), o sistema é descrito pelas equações de estado padrão:

$$x' = Ax + Bu$$

$$y = Cx$$

Substituindo os parâmetros da Tabela 1 nas equações linearizadas, obtém-se a matriz dinâmica *A* do sistema:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 24 & -346 & 10327 \\ 0 & 17 & -494 \end{bmatrix}$$

O vetor de estados *x* é composto pelas três variáveis fundamentais para o controle de equilíbrio e deslocamento: o ângulo de inclinação (φ), a velocidade angular de inclinação (φ') e a velocidade linear do robô (*v*):

$$\mathbf{x} = \begin{bmatrix} \varphi \\ \dot{\varphi} \\ v \end{bmatrix}$$

A variável de controle *u* corresponde à tensão elétrica aplicada aos motores. Com o modelo em espaço de estados definido e a matriz *A* determinada, aplicou-se a técnica de alocação de pólos para calcular os ganhos de controle necessários para estabilizar o sistema.

A verificação das propriedades estruturais do modelo linearizado foi realizada através da análise do posto das matrizes de controlabilidade e observabilidade, requisitos fundamentais para a implementação da estratégia de controle por alocação de pólos. O cálculo da matriz de controlabilidade resultou em um posto igual à ordem do sistema (n=3), o que indica que a variável de atuação, a tensão aplicada aos motores, possui autoridade suficiente para influenciar todos os estados dinâmicos, permitindo a transição do robô de um estado inicial arbitrário para a referência desejada em tempo finito.

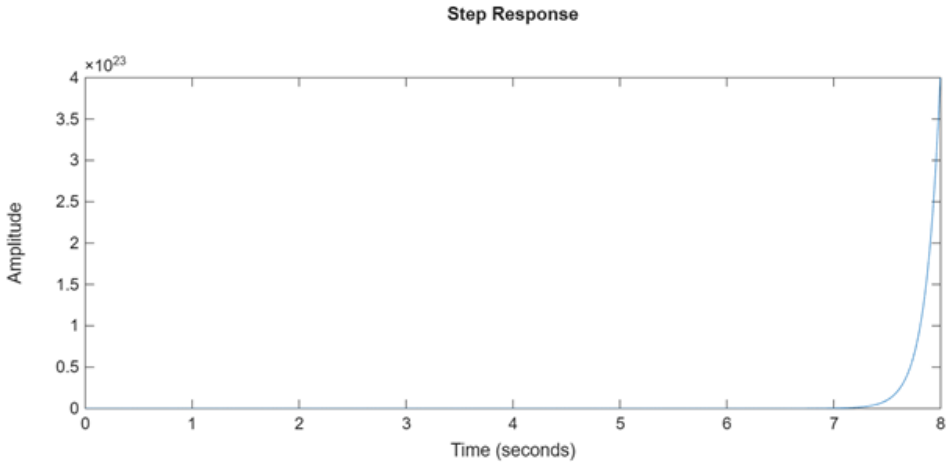
Simultaneamente, a análise da matriz de observabilidade também apresentou posto completo, confirmando que o sistema é observável. Esse resultado demonstra que é matematicamente viável estimar as variáveis de estado internas, como a velocidade angular e a velocidade linear, a partir exclusivamente do histórico da saída medida (o ângulo de inclinação) e do sinal de controle. A confirmação dessas duas propriedades valida o modelo para a síntese do controlador, garantindo que não existem modos dinâmicos ocultos ou incontroláveis que poderiam comprometer a estabilidade da malha fechada.

4.2. REALIMENTAÇÃO DE ESTADOS

A análise preliminar do modelo linearizado do robô de auto-balanceamento, representado pela matriz de estados A, revela a natureza inerentemente instável do sistema. Isso é característico de sistemas do tipo pêndulo invertido, onde ao menos um dos autovalores da matriz de dinâmica possui parte real positiva. A Figura 9 ilustra o comportamento do sistema em malha aberta (sem controle) para uma

condição inicial de desequilíbrio; nota-se que, sem intervenção, o ângulo de inclinação (φ) diverge exponencialmente, levando à queda do robô.

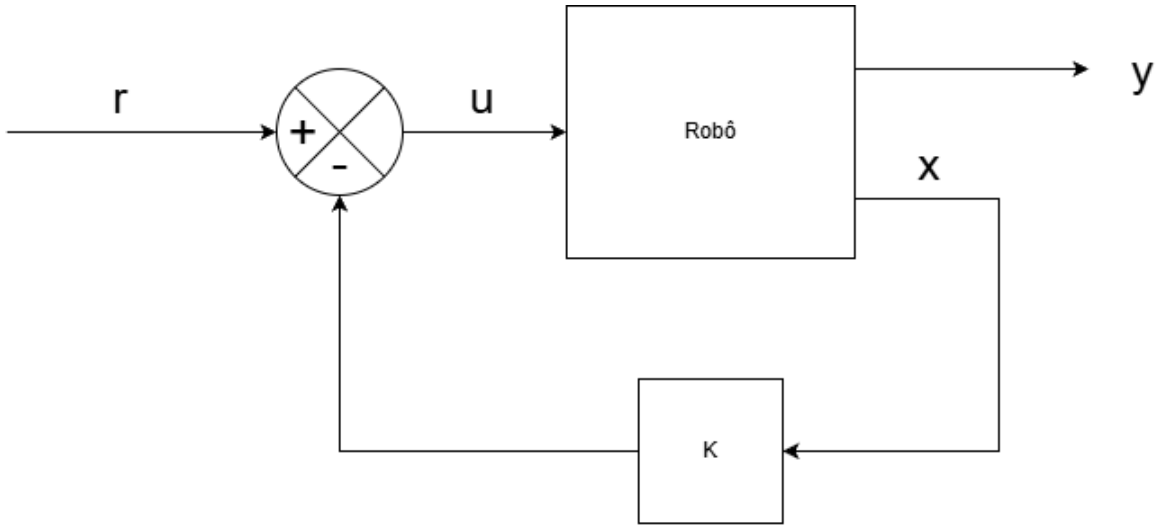
Figura 9: Resposta do sistema em malha aberta.



Fonte: Autores (2025).

Para garantir a estabilização vertical e o controle de velocidade, optou-se pela técnica de controle por alocação de pólos (ou *pole placement*). Esta estratégia baseia-se na realimentação completa de estados, onde a lei de controle é definida por u(t)= −Kx(t). O objetivo fundamental é calcular um vetor de ganhos K tal que os autovalores da matriz de malha fechada (A - BK) sejam deslocados para posições pré-determinadas no plano complexo, garantindo que todas as partes reais sejam negativas e o sistema se torne assintoticamente estável, conforme demonstra o diagrama presente na Figura 10.

Figura 10: Diagrama de controle por realimentação por espaço de estados.



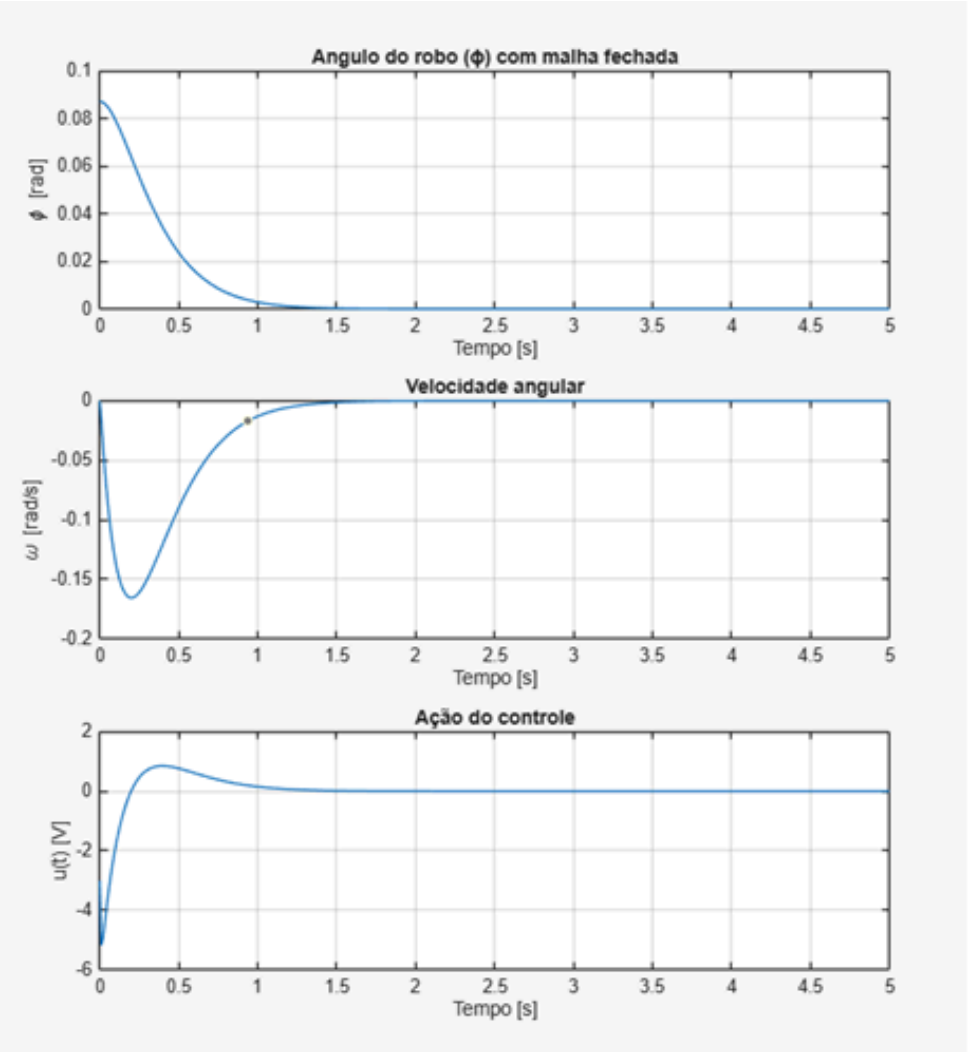
Fonte: Autores (2025).

A sintonia do controlador foi realizada através da escolha dos pólos desejados *p*. Para este projeto, o vetor de polos escolhido foi *p = [−5+i, −5−i, −200]*. Esta escolha reflete duas dinâmicas distintas:

a. Pólos Dominantes (*−5±i*): O par complexo conjugado foi posicionado para ditar a resposta transitória mecânica do robô (o equilíbrio em si). A parte real assegura um tempo de assentamento adequado, enquanto a parte imaginária permite um leve comportamento oscilatório amortecido, necessário para a suavidade do movimento de balanceamento.
b. Polo Rápido (*−200*): O terceiro pólo real foi alocado em uma frequência muito mais alta (mais à esquerda no plano complexo). Isso garante que a dinâmica associada a esta variável (geralmente ligada à resposta elétrica ou velocidade linear) decaia rapidamente e não interfira na estabilidade dominante do ângulo.

Com base nestes critérios, o algoritmo de Ackerman (executado no MATLAB) determinou o vetor de ganhos $K = [35, \ -630, 186.98]$ ideal para impor essa dinâmica. A validação do controlador projetado pode ser observada na Figura 11, que apresenta a resposta do sistema em malha fechada partindo de uma condição inicial de φ=10° (0,087 rad).

Figura 11: Resposta do sistema em malha fechada.



Fonte: Autores (2025).

4.3. ESCOLHA DO PERÍODO DE AMOSTRAGEM E DISCRETIZAÇÃO

O controlador foi projetado no domínio contínuo, mas implementado na plataforma digital. Para garantir que a discretização não comprometesse a estabilidade, adotou-se um período de amostragem de Ts = 10ms (100Hz). Este valor foi escolhido com base na dinâmica dominante do sistema (polos em -5±i), , garantindo que a frequência de amostragem fosse pelo menos 10 vezes a frequência natural do sistema, conforme recomendação clássica para controle digital (FRANKLIN et al., 1998). A lei de controle contínua u(t)= −Kx(t) foi aplicada a cada amostra, utilizando estados estimados a partir dos sensores. Embora o projeto tenha sido realizado em tempo contínuo, a implementação em tempo real exigiu a

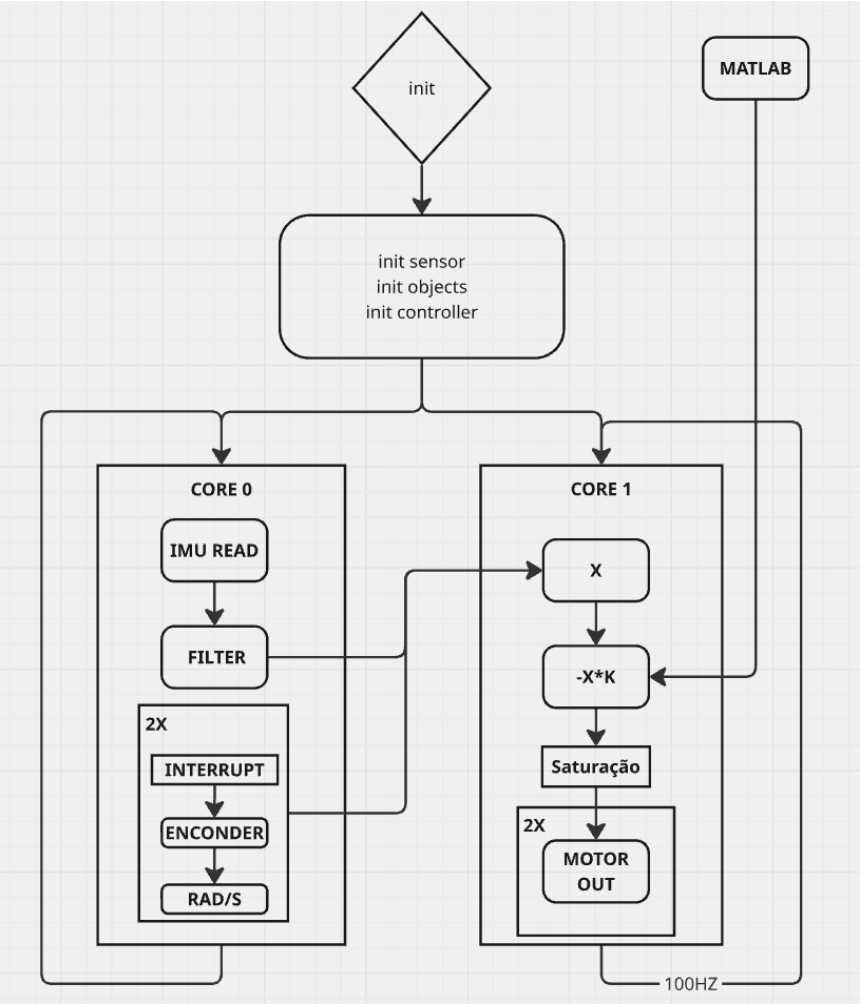
consideração de atrasos computacionais e ruídos de medição, fatores que foram mitigados com o uso de filtros digitais e a escolha conservadora do período de amostragem.

4.4. ARQUITETURA DE FIRMWARE

O desenvolvimento do software embarcado foi estruturado sob o paradigma de Orientação a Objetos (POO), visando a modularidade e a facilidade de manutenção. O código foi projetado para explorar a arquitetura *dual-core* do microcontrolador ESP32, segregando tarefas de aquisição de dados e processamento de controle para garantir a estabilidade temporal da malha. Para garantir a estabilidade do sistema de pêndulo invertido, utilizou-se uma estratégia de controle baseada no espaço de estados desenvolvido via Matlab como já descrito anteriormente.

Para assegurar uma frequência de amostragem e controle fixa de 100Hz sem *jitters* (instabilidades temporais), o firmware divide as responsabilidades entre os dois núcleos do ESP32, utilizando primitivas de sincronização do FreeRTOS, como demonstra a Figura 12.

Figura 12: Arquitetura do firmware.



Fonte: Autores (2025).

Núcleo 0 (Core 0): Aquisição e Pré-processamento

Este núcleo é dedicado às tarefas que exigem interrupções frequentes ou comunicação com periféricos, que poderiam bloquear o processador principal:

1. Leitura do IMU: Comunicação via barramento I2C com o sensor MPU-9250 para aquisição de dados brutos de aceleração e giroscópio.
2. Filtragem de Sinal: Aplicação de algoritmos de fusão sensorial (Filtro) para limpar o ruído do sensor e estimar o ângulo de inclinação real.
3. Odometria (Encoders): Tratamento das interrupções dos encoders (2x) para contagem de pulsos e cálculo da velocidade linear e angular das rodas (rad/s).

Núcleo 1 (Core 1): Malha de Controle Crítica

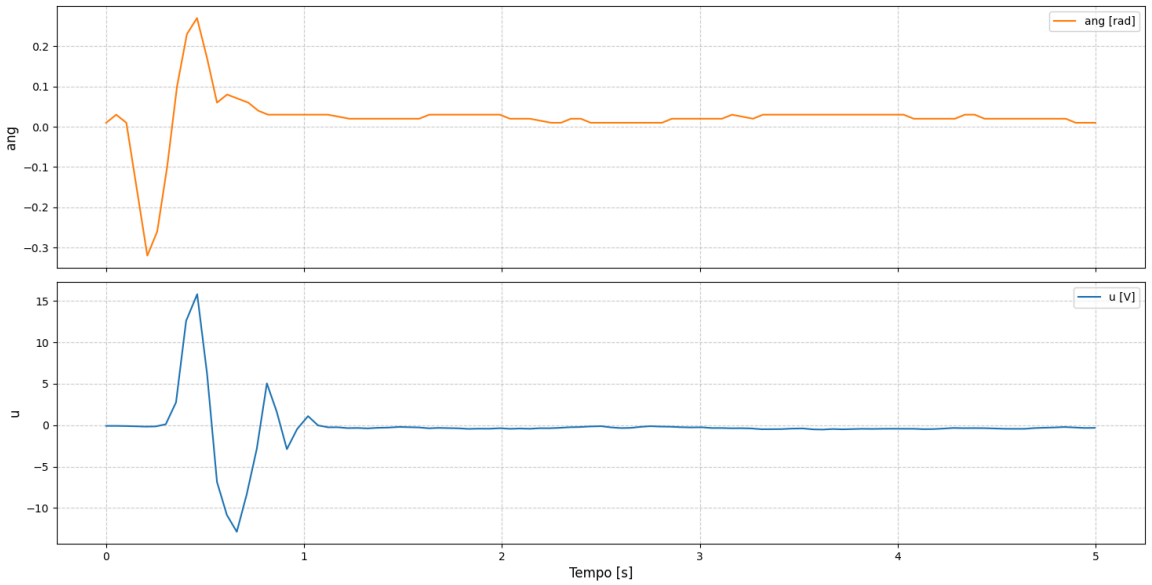
Este núcleo executa o algoritmo de controle propriamente dito, operando de forma síncrona com o tempo:

1. Montagem do Vetor de Estados (x): Consome as informações processadas pelo Core 0 (ângulo filtrado e velocidades).
2. Cálculo da Ação de Controle (-K*x): Realiza a multiplicação matricial do vetor de estados pelo ganho calculado no MATLAB.
3. Saturação: Aplica limites de segurança ao sinal de controle para não exceder a tensão máxima dos motores (Duty Cycle do PWM).
4. Atuação (Motor Out): Envia os comandos finais de PWM para a Ponte H, ajustando a velocidade e direção dos dois motores simultaneamente.

5. TESTES E RESULTADOS

A resposta dinâmica do robô real a uma entrada em degrau, ilustrada pela Figura 13, demonstra que o sistema segue a referência imposta, ainda que apresente divergências em relação ao modelo simulado. Diferentemente da simulação, que opera em condições ideais, os dados experimentais indicam a influência de não linearidades e incertezas paramétricas não contempladas na modelagem matemática.

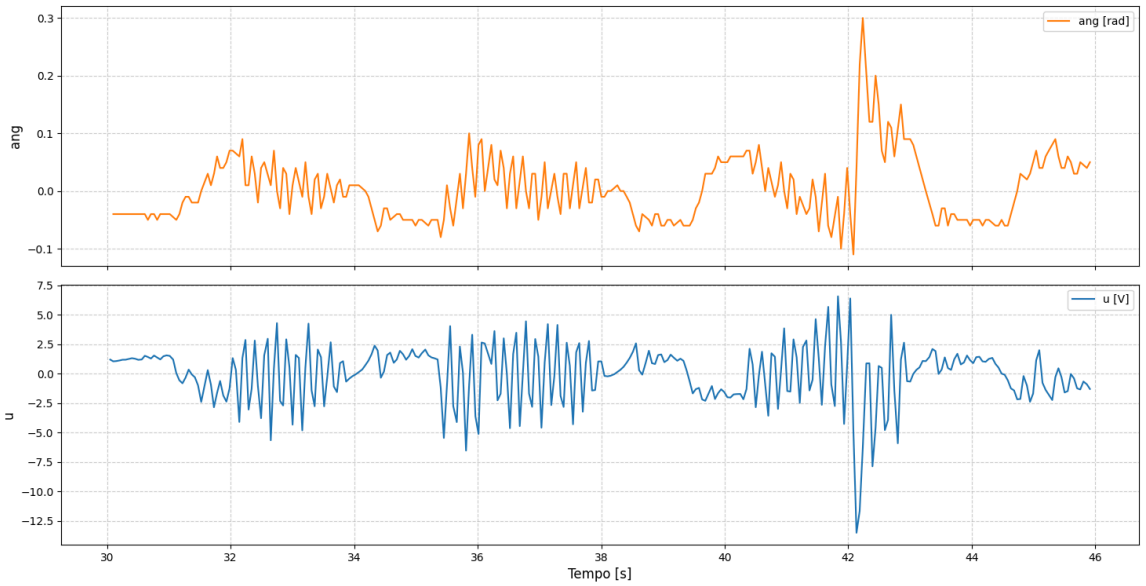
Figura 13: Entrada em degrau experimental.



Fonte: Autores (2025).

A análise do experimento em um intervalo de tempo estendido permite (Figura 14) verificar o comportamento do sistema em regime permanente, revelando oscilações que não seriam detectadas em observações de curta duração. Os dados indicam que o robô não alcança um equilíbrio estático absoluto, em vez disso, opera em um regime dinâmico caracterizado pela variação contínua da tensão de saída. Esse perfil de atuação sugere que o controlador atua de forma constante para corrigir desvios e manter o sistema próximo ao ponto de operação, resultando no ruído observado nos gráficos de sinal de controle.

Figura 14: Resultados experimentais ao longo do tempo.



Fonte: Autores (2025).

A oscilação no sinal de atuação é atribuída à sensibilidade do algoritmo aos ruídos de medição e às aproximações na estimativa dos estados, notadamente a velocidade linear, obtida via derivação numérica dos dados dos encoders. Adicionalmente, o comportamento dinâmico é agravado pelas assimetrias construtivas dos atuadores, como os motores não possuem características eletromecânicas idênticas, observa-se uma resposta desigual em regimes de baixa tensão, onde um dos motores pode iniciar o movimento enquanto o outro permanece estático devido a diferenças na zona morta e atrito interno. Essa não linearidade física, combinada à amplificação de ruídos na malha fechada, força o controlador a gerar correções contínuas para equilibrar o robô, impedindo a estabilização plena da tensão.

6. CONSIDERAÇÕES FINAIS

O desenvolvimento do robô de autoequilíbrio possibilitou a aplicação prática dos conceitos fundamentais abordados na disciplina de Sistemas de Controle B, com destaque para o modelo matemático em espaço de estados, a análise de estabilidade e a síntese de controladores aplicados a sistemas intrinsecamente instáveis. A integração entre teoria, simulação computacional e implementação prática permitiu uma compreensão dos desafios e das limitações associadas ao controle de sistemas reais.

Os objetivos estabelecidos no início do trabalho foram plenamente atendidos. Foi possível desenvolver, implementar e validar uma malha de controle digital capaz de estabilizar o robô de autoequilíbrio em torno da posição vertical. Além disso, o projeto possibilitou a comparação entre diferentes estratégias de controle, evidenciando a superioridade da realimentação de estados em relação ao controle proporcional para o problema proposto.

Durante o desenvolvimento do projeto, diversas dificuldades foram enfrentadas. Entre os principais desafios destacam-se o ruído presente nos sensores inerciais, a assimetria de resposta entre os motores e as limitações decorrentes da linearização do modelo matemático. Para mitigar esses efeitos, foram aplicadas técnicas de filtragem de sinais, ajustes finos nos ganhos do controlador e validações experimentais sucessivas, o que contribuiu para o aumento da robustez do sistema e para a obtenção de um desempenho satisfatório em condições reais de operação.

CONTRIBUIÇÃO DOS MEMBROS DA EQUIPE

O projeto foi desenvolvido de forma colaborativa, com a participação dos integrantes da equipe em diferentes etapas do trabalho, conforme descrito a seguir:

- Pesquisa do modelo matemático do sistema: Willian e Rafaela
- Simulações computacionais e projeto do controlador no MATLAB: Vitor e Willian

- Desenvolvimento do firmware e implementação da malha de controle no ESP32: Guilherme e Vitor
- Projeto mecânico, montagem do protótipo e integração do hardware: Todos
- Realização dos testes experimentais e análise dos resultados: Todos
- Documentação técnica e organização do trabalho: Todos

Essa divisão de tarefas contribuiu para o cumprimento dos prazos e para a qualidade técnica do projeto final.

UTILIZAÇÃO DA INTELIGÊNCIA ARTIFICIAL NO PROJETO

Ferramentas de inteligência artificial foram utilizadas como apoio ao desenvolvimento do projeto, principalmente na etapa de desenvolvimento de firmware e códigos do Matlab, além de auxiliar no desenvolvimento do relatório escrito.

REFERÊNCIAS BIBLIOGRÁFICAS

DORF, R. C.; BISHOP, R. H. **Sistemas de controle modernos**. 12. ed. Rio de Janeiro: LTC, 2016.

ESPRESSIF SYSTEMS. **ESP32 technical reference manual**. Shanghai: Espressif Systems, 2023.

FRANKLIN, G. F.; POWELL, J. D.; WORKMAN, M. **Digital control of dynamic systems**. 3. ed. Menlo Park: Addison-Wesley, 1998.

GRASSI, E.; TSINIAS, J. Robust control of a two-wheeled inverted pendulum. **IEEE Transactions on Control Systems Technology**, v. 13, n. 5, p. 847–854, 2005.

INVENSENSE. **MPU-9250 product specification**. San Jose: InvenSense Inc., 2016.

MAHLER, B.; HAASE, J. Mathematical model and control strategy of a two-wheeled self-balancing robot. In: **IECON 2013 – 39th Annual Conference of the IEEE Industrial Electronics Society**. Vienna, Austria, 2013. p. 4198–4203. DOI: 10.1109/IECON.2013.6699809. Disponível em: https://ieeexplore.ieee.org/document/6699809. Acesso em: out. 2025.

MADGWICK, S. O. H. **An efficient orientation filter for inertial and inertial/magnetic sensor arrays**. 2010. Dissertação (Mestrado) – University of Bristol, Bristol, 2010.

OGATA, K. **Engenharia de controle moderno**. 5. ed. São Paulo: Pearson Prentice Hall, 2010.

TEXAS INSTRUMENTS. **TB6612FNG dual H-bridge motor driver – datasheet**. Dallas: Texas Instruments, 2017.

ANEXO

Código para elaboração do sistema de controle em MATLAB

```
=========================================================

%% Modelo - Self-balancing robot

clc; clear; close all;

%% === Matrizes do modelo linearizado (Eq. 28 do artigo) ===

A = [     0    0.0001         0;

   0.0024   -0.0346    1.0327;

  -0.0000    0.0017   -0.0494]*10^4;

% Entrada = tensão no motor

B = [0; 1; 0];

% Medimos apenas o ângulo ϕ

C = [1 0 0];

D = 0;

%% === Análise de Controlabilidade e Observabilidade ===

% Matriz de Controlabilidade (Co)

Co = ctrb(A, B);

rank_Co = rank(Co);

% Matriz de Observabilidade (Ob)

Ob = obsv(A, C);

rank_Ob = rank(Ob);

% Número de estados do sistema

n_estados = length(A);

fprintf('-------------------------------------------------\n');

fprintf('Número de estados (n): %d\n', n_estados);

fprintf('Posto da Matriz de Controlabilidade: %d\n', rank_Co);

fprintf('Posto da Matriz de Observabilidade: %d\n', rank_Ob);

fprintf('-------------------------------------------------\n');

% resposta do sistema em malha aberta
```

```
step(A, B, C, D);
%% === Sistema contínuo ===
sys_c = ss(A,B,C,D)
%% === Pole Placement ===
p = [-5+i -5-i -200];
K = place(A, B, p)
%% === Sistema CL fechado contínuo ===
A_cl = A - B*K;
sys_cl = ss(A_cl, B, C, D)
% tempo de amostragem para a simulação
Ts = 0.01; % 100 Hz
%% === Simulação ===
t = 0:Ts:5;
x0 = [5*pi/180; 0; 0];   % 10° iniciais
u = zeros(size(t));          % sem referência externa
%% === Simulação Continua ===
[y, t, x] = lsim(sys_cl, u, t, x0);
%% === Plots ===
figure;
subplot(3,1,1)
plot(t, y); grid on;
xlabel('Tempo [s]'); ylabel('\phi [rad]');
title('Angulo do robo (ϕ) com malha fechada');
subplot(3,1,2)
plot(t, x(:,2)); grid on;
xlabel('Tempo [s]'); ylabel('\omega [rad/s]');
title('Velocidade angular');
subplot(3,1,3)
plot(t, -K*x'); grid on;
xlabel('Tempo [s]'); ylabel('u(t) [V]');
```

```
title('Ação do controle');
```

```
==========================================================
```

Código implementado no ESP32

https://github.com/DinossauroBebado/DUNK/blob/7349bee7aae0cb3256d6f3f45706e17015b031a9/lib/ControlTask/ControlTask.cpp#L13