UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira

UNESP - Universidade Estadual Paulista "Júlio de Mesquita Filho"

FEIS - Faculdade de Engenharia de Ilha Solteira

DEE - Departamento de Engenharia Elétrica

KRYSTEL BEATRIZ GOLFETO DA SILVA

CONTROLE PROPORCIONAL-INTEGRAL VS CONTROLE FUZZY DA VELOCIDADE ANGULAR DE UM MOTOR DE CORRENTE-CONTÍNUA

Ilha Solteira

2026

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira

KRYSTEL BEATRIZ GOLFETO DA SILVA

CONTROLE PROPORCIONAL-INTEGRAL VS CONTROLE FUZZY DA VELOCIDADE ANGULAR DE UM MOTOR DE CORRENTE-CONTÍNUA

Trabalho de graduação apresentado à Faculdade de Engenharia de Ilha Solteira – UNESP como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Nome do orientador
Prof. Dr. Jean Marcos de Souza Ribeiro

Ilha Solteira
2026

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

S586c

Silva, Krystel Beatriz Golfeto da.
Controle proporcional-integral vs controle fuzzy da velocidade angular de um motor de corrente-contínua / Krystel Beatriz Golfeto da Silva. -- Ilha Solteira: [s.n.], 2026
101 f. : il.

Trabalho de conclusão de curso (Graduação em Engenharia Elétrica)- Universidade Estadual Paulista (UNESP), Faculdade de Engenharia, Ilha Solteira, 2026

Orientador: Jean Marcos de Souza Ribeiro

Inclui bibliografia

1. Motor de corrente contínua. 2. Arduino. 3. Matlab. 4. Controle PI. 6. Lógica fuzzy.

Sandra Montibeller - CRB-8/060/2025

unesp

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira

ATA DE DEFESA DE TRABALHO DE GRADUAÇÃO

Aos dezessete dias do mês de junho do ano de dois mil e vinte e seis, a discente *Krystel Beatriz Golfeto da Silva*, matriculada sob o nº 211054593, tendo como banca examinadora o seu orientador, *o Prof. Dr. Jean Marcos de Souza Ribeiro, a Profª. Drª. Suely Cunha Amaro Mantovani e o Dr. Lucas do Carmo Yamaguti*, apresentou o Trabalho de Graduação intitulado "Controle Proporcional-Integral vs Controle Fuzzy da velocidade angular de um motor de corrente contínua", obtendo a nota 10 (DEZ) e conceito aprovado.

Prof. Dr. Jean Marcos de Souza Ribeiro
- orientador-

Krystel Beatriz Golfeto da Silva
- discente-

Profª. Drª. Suely Cunha Amaro Mantovani
- Membro da Banca -

Dr. Lucas do Carmo Yamaguti
- Membro da Banca -

Faculdade de Engenharia – Departamento de Engenharia Elétrica
Avenida Professor José Carlos Rossi, 1370 - CEP 15.385-000, Ilha Solteira - SP - Brasil
Tel (18) 37431150 - Fax (18) 3742-2735 - www.feis.unesp.br

AGRADECIMENTOS

Agradeço a Deus pela minha vida, por estar sempre ao meu lado e por me dar força e determinação em todos os momentos da minha jornada.

Agradeço à minha mãe, Patrícia Salete Golfeto, por todo amor, dedicação e apoio incondicional, por sempre acreditar em mim e por ser minha base em todos os momentos.

Agradeço ao meu pai, Gilmar Expedito da Silva, por todo incentivo, por nunca duvidar da minha capacidade, por sempre me motivar a buscar meus sonhos e por me ensinar a encontrar um motivo para sorrir, independentemente da situação.

Agradeço ao meu namorado, Matheus Miranda Azadinho dos Santos, por todo carinho, paciência e compreensão, por estar ao meu lado nos momentos mais difíceis, por acreditar em mim quando eu mesma duvidei e por me mostrar que, independentemente do medo, vale muito a pena enfrentar tudo de frente.

Agradeço aos meus amigos, por todo apoio, incentivo e pelos momentos compartilhados ao longo dessa trajetória.

Agradeço a todos os professores e técnicos desta instituição, que fizeram parte da minha formação acadêmica e me mostraram o quanto a engenharia é incrível. Em especial, agradeço ao professor Jean Marcos de Souza Ribeiro, pela orientação e auxílio no desenvolvimento deste trabalho de conclusão de curso.

Agradeço também à professora Suely Cunha de Amaro Mantovani, por todo suporte, ensinamentos e conselhos, tanto acadêmicos quanto pessoais, e por ter sido essencial na minha formação.

Agradeço a FEIS-UNESP e seus funcionários, que contribuíram direta e indiretamente em minha formação profissional e pessoal, em especial ao técnico de laboratório Valdemir Chaves pela contribuição no protótipo desenvolvido.

DEDICATÓRIA

Dedico este trabalho aos meus pais, que sempre estiveram ao meu lado, se dedicaram e batalharam incansavelmente para me proporcionar o melhor, sendo a base de tudo o que conquistei até aqui.

LISTA DE SÍMBOLOS E ABREVIATURAS

a	Número de Caminhos Paralelos da Armadura
b	Coeficiente de Atrito Viscoso
CA	Corrente Alternada
CC	Corrente Contínua
e(s)	Erro no Domínio de Laplace
EA	Força Contraeletromotriz da Armadura
f	Frequência
FEM	Força Eletromotriz
IA	Corrente de Armadura
IF	Corrente de Campo
IHM	Interface Homem-Máquina
J	Momento de Inércia do Sistema
K	Constante do Motor
Kd	Ganho Derivativo
Ke	Constante da Força Contraeletromotriz
Ki	Ganho Integral
Kp	Ganho Proporcional
Kt	Constante de Torque
Kw	Constante de Velocidade
LA	Indutância da Armadura
LF	Indutância do Campo
Ms	Máximo Sobressinal
n	Velocidade de Rotação
NA	Velocidade Atual
Nreferência	Velocidade de Referência
P	Controle Proporcional
PI	Controle Proporcional-Integral
PPR	Pulsos por Revolução
PWM	Pulse Width Modulation
RA	Resistência de Armadura
RF	Resistência das Bobinas de Campo
rpm	Rotações por Minuto
Tb	Torque Viscoso
Td	Tempo Derivativo
Te	Torque Eletromagnético
Te(s)	Torque Elétrico no Domínio de Laplace
Ti	Tempo Integral
Tm	Torque Mecânico
Tm(s)	Torque Mecânico no Domínio de Laplace
Tp	Tempo de Pico
Ts	Tempo de Subida
u(s)	Sinal de Entrada no Domínio de Laplace
V	Tensão Terminal
VA	Tensão da Armadura
Vescova	Tensão na Escova

VF	Tensão de Campo
W(s)	Velocidade Angular no Domínio de Laplace
W0	Velocidade Nominal
Y(s)	Sinal de Saída no Domínio de Laplace
z	Quantidade de Condutores
ΔPWM	Incremento do Sinal PWM
ζ	Coeficiente de Amortecimento
φ	Fluxo Magnético de Campo
ω	Velocidade Angular
ωn	Frequência Natural
a	Número de Caminhos Paralelos da Armadura

RESUMO

Neste trabalho é proposto o projeto, a simulação e a implementação de controladores PI e *Fuzzy* para a regulação da velocidade angular de um motor de corrente contínua (CC), por meio do controle da tensão de armadura, mantendo o campo magnético constante. A motivação para o estudo está na ampla aplicação dos motores CC em diversos setores industriais, nos quais o controle preciso da velocidade é essencial para garantir a eficiência e a qualidade dos processos. Embora os controladores Proporcional-Integral (PI) apresentem bom desempenho em sistemas lineares, podem enfrentar limitações diante de não linearidades e variações de carga. Em contrapartida, controladores baseados em Lógica *Fuzzy* destacam-se pela maior robustez e adaptabilidade em sistemas complexos.

O desenvolvimento do trabalho compreende a modelagem matemática do sistema e a realização de simulações no ambiente MATLAB/Simulink, seguidas da construção de um protótipo experimental utilizando Arduino©, ponte H e *encoder* para a realimentação da velocidade. Como resultado, são apresentados e comparados os desempenhos dos controladores por meio de métricas como tempo de resposta, sobresinal (*overshoot*) e erro em regime permanente. As análises realizadas contribuem para a seleção de estratégias de controle mais adequadas a aplicações de automação e robótica.

Palavras-Chave: Motor de corrente contínua, *encoder*, Arduino, Matlab, controle PI, Lógica *Fuzzy*.

ABSTRACT

This work proposes the design, simulation, and implementation of PI and Fuzzy controllers for the regulation of the angular speed of a direct current (DC) motor through armature voltage control, while maintaining a constant magnetic field. The motivation for this study lies in the widespread use of DC motors in various industrial sectors, where precise speed control is essential to ensure process efficiency and quality. Although Proportional-Integral (PI) controllers exhibit satisfactory performance in linear systems, they may face limitations when dealing with nonlinearities and load variations. In contrast, controllers based on Fuzzy Logic stand out for their greater robustness and adaptability in complex systems.

The development of this work includes the mathematical modeling of the system and simulations performed in the MATLAB/Simulink environment, followed by the construction of an experimental prototype using an Arduino©, an H-bridge, and an encoder for speed feedback. As a result, the performance of the controllers is presented and compared through metrics such as response time, overshoot, and steady-state error. The analyses carried out contribute to the selection of more appropriate control strategies for automation and robotics applications.

Keywords: Direct current motor, *encoder*, Arduino, MATLAB, PI control, *Fuzzy* Logic.

LISTA DE FIGURAS

Figura 1: Representações esquemáticas de uma máquina CC 22
Figura 2: Princípio de funcionamento de um motor cc 24
Figura 3: Ligação do circuito de campo de motores CC 26
Figura 4: Comparação das características Torque–carga de motores CC 27
Figura 5: Circuito equivalente de um motor CC 29
Figura 6: Circuito equivalente dinâmico de um motor CC 30
Figura 7: Diagrama de blocos do motor CC com excitação independente 32
Figura 8: Diagrama de blocos do motor CC com excitação independente com $\boldsymbol{Tm(s)}$ 32
Figura 9: Sistema de malha aberta 34
Figura 10: Sistema de malha fechada 35
Figura 11: Curva típica de resposta impulsiva de um sistema de segunda ordem 38
Figura 12: Diagrama de blocos do Controle proporcional 39
Figura 13: Diagrama de blocos Controle Integral 40
Figura 14: Esquema de implementação de um controlador tipo PI 41
Figura 15: Esquema de implementação de um controlador tipo PID 43
Figura 16: Representação de uma variável linguística 45
Figura 17: Estrutura de um sistema de inferência Fuzzy 46
Figura 18: Tela inicial da IDE arduino 49
Figura 19: Motor CC usado para o controle 53
Figura 20: Motor CC utilizado para simular a carga 53
Figura 21: Placa do Encoder 54
Figura 22: Ponte H L298N 55
Figura 23: Circuito interno da ponte H 56
Figura 24: Estrutura do Arduino Uno 57
Figura 25: Conexão esquemática para o controle do motor cc 59
Figura 26:Gráfico comparativo entre as equações do motor CC 62
Figura 27: Diagrama de bloco do motor CC com o controlador PI 63
Figura 28: Gráfico de velocidade x tempo 63
Figura 29: Esforço do controlador PI 63
Figura 30: Gráfico de velocidade do motor CC, com o ganho Ki alto 64
Figura 31: Gráfico do esforço do controlador PI quando Ki é alto 65
Figura 32: Velocidade do motor CC com Kp alto 65
Figura 33: Gráfico do esforço do controlador PI com Kp alto 66

Figura 34: Funções de pertinência da variável de entrada erro do controlador Fuzzy 67
Figura 35: Funções de pertinência da variável de entrada derro do controlador Fuzzy 68
Figura 36:Funções de pertinência da variável de saída PWM do controlador Fuzzy.....69
Figura 37: Superfície de ingerência do controlador Fuzzy.....71
Figura 38: Esquamático do controlador Fuzzy 72
Figura 39: Diagrama de blocos do controlador Fuzzy.....72
Figura 40: Gráfico da velocidade do motor com o controlador Fuzzy 73
Figura 41: Gráfico do esforço do controlador Fuzzy.....73
Figura 42: Primeira camada da interface 74
Figura 43: Segunda camada da interface 75
Figura 44: Terceira camada da interface.....76
Figura 45: Quarta camada da interface=.....76
Figura 46: Motor CC com controlador PI sem carga adicional.....77
Figura 47: Motor CC com controle PI tendo apenas Kp 78
Figura 48: Motor somente com ganho integrador.....79
Figura 49:Motor CC com Ki = Kp.....79
Figura 50: Motor CC com controlador PI e carga baixa.....80
Figura 51:Motor CC com controlador PI e carga média 81
Figura 52: Motor CC com controlador PI e carga alta.....81
Figura 53: Motor CC com controlador Fuzzy.....82
Figura 54: Motor CC com controlador Fuzzy com a velocidade estabilizada.....82
Figura 55:Motor CC com controlador Fuzzy e carga baixa 83
Figura 56: Motor CC com controlador Fuzzy e carga média 84
Figura 57: Motor CC com controlador Fuzzy e carga alta 84

LISTA DE TABELAS

Tabela 1: Parâmetros das funções de pertinência da variável de entrada erro 68
Tabela 2: Parâmetros das funções de pertinência da variável de entrada derro 69
Tabela 3: Parâmetros das funções de pertinência da variável de saída PWM 70
Tabela 4: Base de regras do controlador *Fuzzy* 70

SUMÁRIO

1. INTRODUÇÃO 15

1.1 Objetivos 16

1.2 Organização Do Texto 16

2. REVISÃO DE LITERATURA 18

3. FUNDAMENTAÇÃO TEÓRICA 21

3.1 Máquinas CC 21

3.1.1 Aspectos construtivos das máquinas CC 21

3.1.2 Principio de funcionamento da máquina cc 23

3.2 Motores CC 25

3.3 Características Torque-Carga 27

3.4 Equação Da Velocidade E Modelo Dinâmicos Do Motor CC 28

3.5 Características Dinâmicas Do Motor Cc 30

3.6 Sistema De Controle 32

3.6.1 Controle de malha aberta 33

3.6.2 Controle de malha fechada 34

3.6.3 Controle de malha fechada vs controle de malha aberta 35

3.6.4 Análise da resposta transitória 35

3.7 Tipos De Controladores 38

3.7.1 Controle proporcional (P) 38

3.7.2 Controle integral 39

3.7.3. Controle proporcional-integral 40

3.7.4 Controle proporcional-derivativo 41

3.7.5 Controle proporcional integral derivativo 42

4. LOGICA FUZZY 43

4.1 Conjuntos Fuzzy 44

4.2 Variáveis Linguísticas e Funções De Pertinência 45

4.3 Sistema De Inferência Fuzzy 46

5. *SOFTWARE*S UTILIZADOS 47

5.1 Matlab 47

5.2 Elipse E3 47

5.3 IDE Arduino Uno 48

6. DESENVOLVIMENTO DO PRÓTOTIPO DE CONTROLE DE VELOCIDADE DE UM MOTOR CC 51

6.1 Componentes de Hardware utilizados 51
6.2 Motor de e *Encoder* 51
6.3 Ponte H L298n E Fonte Dc 54
6.4 Arduino Uno 57
6.5 Montagem Do Protótipo Na Bancada 58
7. RESULTADOS E DISCUSSÕES 60

7.1 Considerações Iniciais 60
7.2 Simulação Do Modelo Encontrado 62
7.2.1 Simulação do controlador pi7 62
7.2.2 Simulação do controlador fuzzy 66
5.3 Programas Do Arduino E Supervisório 74
5.4 Aplicando Cargas E Ligando O Controlador Pi 76
5.4.1 Controlador PI 77
5.4.2 Controlador pi com carga 80
5.5 Controlador *Fuzzy* 82
5.5.1 Controlador *fuzzy* com acréscimo de carga 83
8. CONCLUSÃO 86

9. REFERÊNCIAS BIBLIOGRÁFICAS 87

APÊNDICE A 89

APÊNDICE B 93

1. INTRODUÇÃO

As máquinas de corrente contínua (CC) são amplamente empregadas em sistemas de acionamento devido à sua versatilidade, facilidade de controle e boa resposta dinâmica. Apesar de apresentarem maior robustez e, em alguns casos, manutenção mais complexa quando comparadas às máquinas de corrente alternada (CA), as máquinas CC continuam sendo utilizadas em diversas aplicações industriais e acadêmicas, especialmente naquelas que exigem controle preciso de velocidade e Torque (CHAPMAN, 2013).

Uma das principais vantagens das máquinas CC está relacionada às diferentes possibilidades de excitação (derivação, série ou composta), que resultam em variadas características de desempenho. Essa flexibilidade torna as máquinas CC altamente adaptáveis aos sistemas de controle, tanto manuais quanto automáticos, permitindo sua aplicação em processos que demandam controle refinado de variáveis como velocidade e posição (KOSOW, 2005).

Com o objetivo de melhorar o desempenho, a estabilidade e a eficiência dessas máquinas, diversos métodos de controle vêm sendo estudados desde o século XIX, consolidando a teoria de controle como uma das áreas mais tradicionais e relevantes da engenharia (CHAPMAN, 2013). No caso específico dos motores CC, o controle da velocidade angular pode ser realizado por meio de três abordagens principais: variação da tensão de armadura, variação da corrente de campo e inserção de resistência na armadura (FUENTES, 2005).

Dentre essas estratégias, o controle por variação da tensão de armadura destaca-se por sua simplicidade e eficiência, sendo utilizado em aplicações que exigem resposta rápida e precisão no controle da velocidade. Nesse método, o fluxo magnético do campo permanece constante, enquanto a velocidade do motor é ajustada por meio da modificação da tensão aplicada à armadura, característica que favorece sua implementação em sistemas de controle embarcados.

Entre as estratégias de controle aplicadas em sistemas industriais, destacam-se os controladores Proporcional-Integral (PI) e os controladores baseados em lógica Fuzzy. O controlador PI combina as ações proporcional e integral, oferecendo um compromisso entre estabilidade, precisão e rapidez de resposta. Esse tipo de controlador é empregado em sistemas lineares, sobretudo quando o modelo matemático da planta não é completamente conhecido, o que dificulta a aplicação de métodos analíticos mais complexos. A ação proporcional atua

diretamente sobre o erro instantâneo, contribuindo para a redução das oscilações do sistema, enquanto a ação integral atua na eliminação do erro em regime permanente, ajustando gradativamente a resposta até que a variável controlada alcance o valor de referência desejado. Dessa forma, o controlador PI apresenta simplicidade de implementação e boa eficiência prática em aplicações industriais (OGATA,2010).

Entretanto, em sistemas não lineares ou sujeitos a incertezas e variações de parâmetros, os controladores baseados em lógica *Fuzzy* podem apresentar desempenho superior (FREITAS, 2014). Fundamentados na teoria dos conjuntos nebulosos, proposta por Lotfi Zadeh em 1965, os controladores Fuzzy utilizam regras linguísticas do tipo "Se–Então", permitindo a representação de conhecimentos humanos e a tomada de decisão em situações imprecisas ou incertas (ZADEH *et al.,* 1996).

1.1 Objetivos

Este trabalho tem como objetivo realizar uma análise comparativa entre os controladores PI e *Fuzzy* aplicados ao controle de velocidade de um motor de corrente contínua. A implementação prática foi realizada utilizando um motor DC de 6 V e 210 rpm, um *encoder* para aquisição da velocidade, uma ponte H para acionamento dos motores (motor controlado e carga), o microcontrolador Arduino Uno e o *Software* supervisório Elipse E3. A comparação considera critérios como desempenho dinâmico, estabilidade, complexidade de implementação e precisão no controle da velocidade.

1.2 Organização Do Texto

Além desta introdução, no Capítulo 2 é apresentado a revisão de literatura, abordando os principais conceitos relacionados aos motores de corrente contínua, sistemas de controle e aplicações de controladores PI e Fuzzy.

No Capítulo 3 detalha-se a fundamentação teórica, descrevendo os princípios de funcionamento dos motores CC, os sistemas de controle e os controladores clássicos e Fuzzy.

No Capítulo 4 descreve-se os componentes utilizados e a montagem experimental do sistema.

No Capítulo 5 apresenta-se a metodologia adotada, a identificação do modelo do motor e as simulações realizadas.

No Capítulo 6 discute-se os resultados obtidos em simulação e em bancada experimental.

Por fim, no Capítulo 7 apresenta-se as conclusões e sugestões para trabalhos futuros.

2. REVISÃO DE LITERATURA

Neste capítulo, são apresentados alguns trabalhos que contribuíram com conhecimentos relevantes e serviram de fundamentação para o desenvolvimento desta pesquisa.

No trabalho desenvolvido por Gregório (2023), foi proposta a aplicação da lógica *Fuzzy* no controle de velocidade de um motor de corrente contínua utilizando um sistema embarcado baseado no Arduino Uno. O estudo foi realizado no contexto da engenharia mecatrônica e teve como objetivo avaliar o desempenho do controlador *Fuzzy* em comparação com o controlador PID tradicional. Para isso, Gregório (2023) desenvolveu um Sistema Baseado em Regras *Fuzzy*, operando em malha fechada, utilizando como variáveis de entrada o erro de velocidade e a variação do erro, obtidos por meio de leituras de um *encoder* acoplado ao eixo do motor.

O controle foi implementado utilizando um modelo *Fuzzy* do tipo Sugeno, escolhido devido à sua menor complexidade computacional e melhor adaptação a sistemas embarcados. As funções de pertinência utilizadas no controlador foram do tipo triangular, definidas empiricamente com base nos limites operacionais do motor e nos resultados observados durante os testes experimentais. A base de regras *Fuzzy* foi elaborada de forma a reproduzir o raciocínio de um especialista em controle, estabelecendo ações capazes de corrigir o erro de velocidade de maneira gradual e estável. Segundo Gregório (2023), o método de inferência adotado utilizou a T-norma de Zadeh, enquanto a saída do sistema *Fuzzy* correspondeu ao incremento do sinal de controle aplicado ao motor, convertido posteriormente em um sinal PWM.

Os resultados experimentais demonstraram que o controlador *Fuzzy* apresentou comportamento suave, com menor *overshoot* e melhor adaptação às variações de referência quando comparado ao controlador PID em determinadas condições de operação. O estudo evidencia que a definição criteriosa das funções de pertinência e da base de regras é fundamental para o sucesso da aplicação do controle *Fuzzy*. Além disso, o autor conclui que o controle *Fuzzy* se mostra uma alternativa eficiente e robusta para o controle de velocidade de motores CC, especialmente em sistemas sujeitos a não linearidades e incertezas.

No trabalho desenvolvido por Barbosa, Alves e Alexandre (2026), foi realizada uma análise comparativa entre os controladores PID clássico e PID-*Fuzzy* aplicados à regulação de velocidade de um motor de corrente contínua equipado com encoder. O estudo teve como objetivo avaliar experimentalmente o desempenho de ambas as estratégias de controle em uma bancada didática desenvolvida pelos autores, composta por um microcontrolador ESP32, um

driver ponte H e uma interface supervisória SCADA implementada no software Elipse E3. Inicialmente, a dinâmica do motor foi identificada em malha aberta por meio do método da curva de reação, sendo modelada como um sistema de primeira ordem com tempo morto. Para o controlador PID clássico, os parâmetros foram obtidos por técnicas de sintonia consagradas, como Ziegler-Nichols e Cohen-Coon, sendo posteriormente ajustados experimentalmente para melhorar a estabilidade e reduzir o erro em regime permanente. Já o controlador PID-*Fuzzy* foi desenvolvido com base em um sistema de inferência *Fuzzy* composto por duas variáveis de entrada, correspondentes ao erro e à derivada do erro, e três variáveis de saída, associadas aos ganhos proporcional, integral e derivativo do controlador.

Segundo Barbosa, Alves e Alexandre (2026), as funções de pertinência adotadas foram dos tipos trapezoidal e triangular, organizadas em conjuntos linguísticos classificados como pequeno, médio e grande. A lógica de inferência foi empregada para realizar o ajuste dinâmico dos ganhos do PID em tempo real, permitindo a adaptação às condições operacionais do sistema. Os testes experimentais foram conduzidos para um degrau de referência de 50 RPM, sendo avaliados parâmetros temporais e métricas de erro.

Os resultados mostraram que ambos os controladores foram capazes de estabilizar o motor e acompanhar a referência de velocidade; entretanto, o PID clássico apresentou desempenho superior, alcançando menor sobressinal (24,7%), menor tempo de acomodação (0,65 s) e menor erro em regime permanente (1,0 RPM). Em contrapartida, o PID-*Fuzzy* apresentou oscilações sustentadas próximas ao ponto de operação, resultando em maior erro acumulado e menor estabilidade. Os autores concluem que, para a planta analisada, o controlador PID tradicional mostrou-se mais eficiente, preciso e robusto, embora a estratégia PID-*Fuzzy* apresente potencial para aplicações em sistemas com maior grau de não linearidade e em pesquisas futuras envolvendo técnicas de otimização e adaptação automática de ganhos.

No trabalho desenvolvido por Boscolo (2019), foi proposta a implementação de um controlador Proporcional-Integral (PI) para o controle da velocidade angular de um motor de corrente contínua, com o objetivo de manter a velocidade constante mesmo diante de variações de carga aplicadas ao eixo do motor. O estudo foi desenvolvido utilizando os softwares Matlab/Simulink e Elipse E3, juntamente com a plataforma Arduino Uno, empregada para a aquisição de dados e atuação do sistema de controle. Inicialmente, foi realizada a modelagem matemática do motor CC a partir de seus parâmetros elétricos e mecânicos, os quais foram obtidos experimentalmente por meio de ensaios de identificação.

Com base nesse modelo, foram efetuadas simulações em ambiente Matlab/Simulink para analisar a resposta dinâmica do sistema sob diferentes condições de operação e realizar a sintonia dos ganhos do controlador PI por meio da ferramenta de ajuste gráfico disponível no software. Posteriormente, foi desenvolvida uma bancada experimental composta por um motor CC equipado com encoder para medição da velocidade, uma ponte H L298N para acionamento da armadura e um segundo motor utilizado para simular a aplicação de cargas mecânicas variáveis. Segundo Boscolo (2019), o sinal de realimentação foi obtido a partir das leituras do encoder, permitindo o cálculo do erro entre a velocidade de referência e a velocidade medida, enquanto a ação de controle foi aplicada por meio da modulação PWM gerada pelo Arduino.

A interface supervisória desenvolvida no Elipse E3 possibilitou o monitoramento em tempo real das variáveis do processo, a alteração dos ganhos de controle e a visualização do comportamento do sistema durante os experimentos. Os resultados das simulações e dos testes em bancada demonstraram que o controlador PI foi capaz de compensar as reduções de velocidade provocadas pelo aumento da carga, restabelecendo o valor de referência de forma satisfatória. Além disso, verificou-se que a escolha adequada dos ganhos proporcional e integral é fundamental para minimizar oscilações, reduzir o sobressinal e garantir tempos de acomodação adequados. O autor conclui que o controle PI apresentou desempenho eficiente e robusto para a aplicação proposta, constituindo uma solução de baixo custo e fácil implementação para o controle de velocidade de motores CC utilizados em sistemas de automação industrial

3. FUNDAMENTAÇÃO TEÓRICA

Nesta seção são apresentados os fundamentos teóricos que sustentam o desenvolvimento deste trabalho. Inicialmente, abordam-se os conceitos relacionados às máquinas de corrente contínua e aos motores de corrente contínua, incluindo seus aspectos construtivos, princípio de funcionamento, classificação e modelagem matemática. Em seguida, são discutidos os fundamentos dos sistemas de controle aplicados a motores elétricos, com ênfase nos controladores clássicos Proporcional (P), Proporcional-Integral (PI) e Proporcional-Integral-Derivativo (PID). Por fim, são apresentados os conceitos fundamentais da lógica *Fuzzy,* abrangendo conjuntos Fuzzy, variáveis linguísticas, funções de pertinência e sistemas de inferência Fuzzy, que fornecem a base teórica para a implementação do controlador *Fuzzy.*

3.1 Máquinas CC

As máquinas CC constituem um dos três tipos básicos de máquinas elétricas, juntamente com as máquinas síncronas e as máquinas de indução, nesse cenário destacam-se, pela facilidade de controle da velocidade e do Torque, o que as tornam particularmente adequadas para sistemas de acionamento que demandam elevada precisão e boa resposta dinâmica.

Deste modo, essas máquinas podem operar tanto como motores quanto como geradores, dependendo das condições de funcionamento, de modo geral, qualquer máquina elétrica é capaz de realizar a conversão de energia em ambos os sentidos, assim, um mesmo dispositivo pode, em princípio, operar como motor ou como gerador, na prática, contudo, a maioria das máquinas é projetada para desempenhar uma função específica, realizando a conversão de energia por meio da ação de campos magnéticos (CHAPMAN,2013).

3.1.1 Aspectos construtivos das máquinas CC

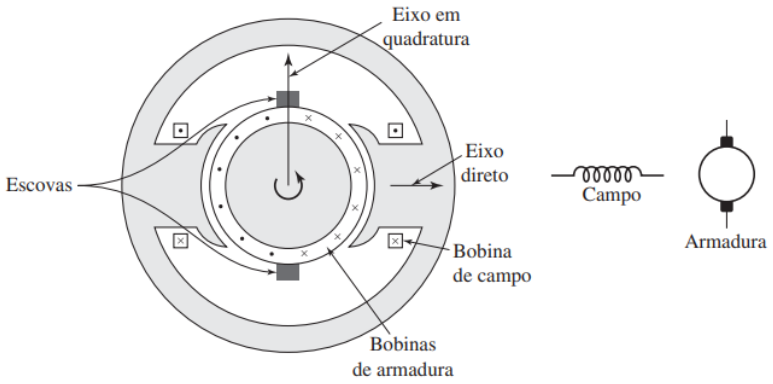
A estrutura física de uma máquina de cc é composta, essencialmente, por dois elementos principais: o estator, que constitui a parte estacionária da máquina, e o rotor, que corresponde à parte rotativa responsável pela conversão eletromecânica de energia.

O estator é formado por uma carcaça, cuja função principal é fornecer suporte mecânico e rigidez estrutural ao conjunto, além de proteger os componentes internos. Acopladas à carcaça encontram-se as peças polares, que se projetam em direção ao interior da máquina e fornecem um caminho de baixa relutância para o fluxo magnético principal. As extremidades

dessas peças, localizadas próximas à superfície do rotor, apresentam um alargamento denominado sapata polar, cuja finalidade é distribuir o fluxo magnético de maneira mais uniforme ao longo da periferia do rotor. A superfície exposta da sapata polar é denominada face polar, e o espaço existente entre esta face e a superfície do rotor recebe o nome de entreferro de ar, ou simplesmente entreferro (CHAPMAN, 2013).

Na Figura 1 apresenta-se uma representação esquemática de uma máquina de corrente contínua, destacando seus principais componentes.

Figura 1: Representações esquemáticas de uma máquina CC



Fonte: (Fitzgerald, 2014).

Em uma máquina de corrente contínua existem dois conjuntos fundamentais de enrolamentos: os enrolamentos de campo e os enrolamentos de armadura. Os enrolamentos de campo são responsáveis pela produção do fluxo magnético principal da máquina, enquanto os enrolamentos de armadura são aqueles nos quais a tensão é induzida e por onde circula a corrente que resulta na produção do Torque eletromagnético. Nas máquinas CC convencionais, os enrolamentos de campo estão localizados no estator, ao passo que os enrolamentos de armadura se encontram no rotor. Por esse motivo, o rotor de uma máquina de corrente contínua é frequentemente denominado armadura (CHAPMAN, 2013).

A armadura é constituída por um eixo, geralmente fabricado em aço, sobre o qual é montado um núcleo ferromagnético composto por diversas lâminas de aço silicioso empilhadas. A utilização de lâminas tem como principal objetivo a redução das perdas por correntes parasitas, melhorando a eficiência da máquina. O núcleo laminado apresenta ranhuras em sua superfície externa, destinadas ao alojamento das bobinas do enrolamento de armadura, que são distribuídas de forma a garantir adequada produção de Torque e equilíbrio eletromagnético.

Em uma das extremidades do eixo encontra-se o comutador, ao qual são conectadas as extremidades das bobinas da armadura. O comutador é formado por segmentos individuais de cobre, eletricamente isolados entre si, mantidos na forma cilíndrica por meio de um aro de aperto. A alimentação elétrica da armadura é realizada por uma fonte de tensão contínua, sendo a corrente conduzida até os segmentos do comutador por meio de escovas, que permanecem estacionárias durante a operação.

A principal função do conjunto comutador–escovas é promover a inversão adequada do sentido da corrente nos enrolamentos da armadura ao longo da rotação do rotor. Essa inversão garante que o Torque eletromagnético desenvolvido mantenha sempre o sentido, possibilitando a rotação contínua do eixo do motor e assegurando o correto funcionamento da máquina sob diferentes condições de carga.

3.1.2 Princípio de funcionamento da máquina cc

A ação motora em máquinas elétricas ocorre quando uma corrente elétrica circula por condutores imersos em um campo magnético. Nessas condições, os condutores ficam sujeitos à ação de uma força eletromagnética resultante da interação entre o campo magnético e a corrente elétrica (CHAPMAN, 2013). A intensidade dessa força pode ser determinada a partir da lei da força de Lorentz, sendo expressa matematicamente pela Equação (1),

$$F_c = B\, i\, l\, \sin\theta \quad . \tag{1}$$

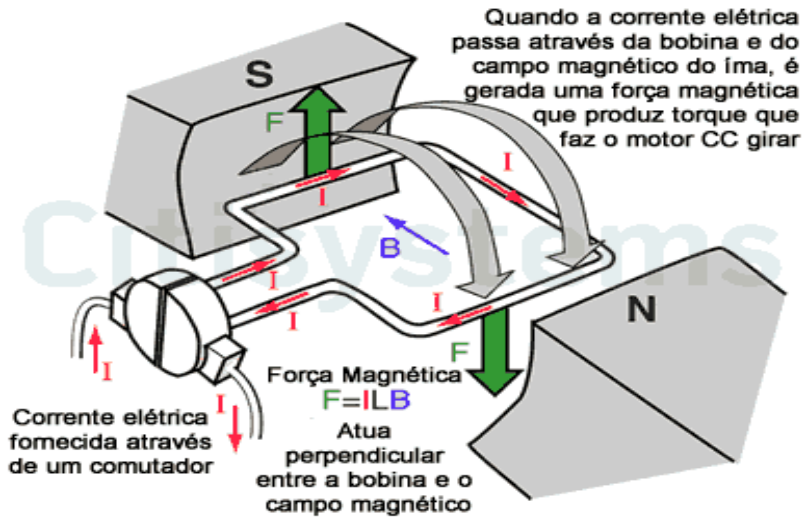
Quando esses condutores são dispostos em uma estrutura mecanicamente livre para girar, como ocorre na armadura de um motor de corrente contínua, as forças eletromagnéticas atuantes resultam na produção de um conjugado eletromagnético (T_e). Esse conjugado é o responsável pela rotação do eixo da máquina e pela velocidade angular (ω).

O princípio de funcionamento do motor de corrente contínua fundamenta-se, portanto, na interação entre campos magnéticos e correntes elétricas. Contudo, para uma análise mais completa do comportamento dessas máquinas, torna-se indispensável considerar também a lei da indução eletromagnética de Faraday, segundo a qual uma tensão é induzida em um condutor sempre que houver variação do fluxo magnético ao qual ele está submetido. O módulo da tensão induzida pode ser expresso pela Equação (2),

$$e_{ind} = B\, l\, v\, \sin\theta \quad . \tag{2}$$

Na Figura 2 apresenta-se uma representação esquemática do princípio de funcionamento de um motor cc.

Figura 2: Princípio de funcionamento de um motor cc



Fonte: (Bertulucci, 2018)

No motor de cc, essa tensão induzida manifesta-se sob a forma da força contraeletromotriz (FCEM), cuja polaridade se opõe à tensão aplicada aos terminais da armadura. A força contraeletromotriz desempenha papel fundamental no equilíbrio elétrico da máquina, atuando como um fator limitador da corrente de armadura e exercendo influência direta sobre a regulação e o controle da velocidade do motor.

Em uma armadura real, composta por muitos condutores distribuídos uniformemente ao longo do rotor, as forças eletromagnéticas atuam de maneira simultânea e combinada. Como resultado, o Torque eletromagnético total desenvolvido pela máquina corresponde à soma dos Torques individuais produzidos por cada condutor ativo da armadura.

De forma simplificada, o conjugado eletromagnético total pode ser expresso pela Equação (3),

$$T_e = F_c \, Z_a \, r \quad . \tag{3}$$

A equação (3), evidencia que o Torque desenvolvido pelo motor de cc é diretamente proporcional tanto à força eletromagnética atuante nos condutores quanto à quantidade e à disposição geométrica desses condutores no rotor, sendo um resultado fundamental para a análise do desempenho eletromecânico da máquina.

Conforme definido por Kosow (2005), o Torque representa a tendência de uma força, aplicada a uma determinada distância do eixo de rotação, produzir movimento rotacional, sendo expresso no Sistema Internacional pela unidade newton-metro (N·m). Diferentemente do trabalho mecânico, o Torque pode existir mesmo na ausência de movimento, desde que haja uma força atuando a uma distância radial do eixo de rotação.

Em motores de corrente contínua comerciais, os parâmetros construtivos da máquina, tais como o número de polos, a quantidade de condutores, o número de caminhos da armadura e a geometria do rotor, permanecem constantes durante a operação. Dessa forma, o Torque eletromagnético pode ser expresso de maneira mais conveniente apenas em função das variáveis elétricas e magnéticas do sistema.

Nessas condições, o conjugado eletromagnético desenvolvido pela armadura é diretamente proporcional ao fluxo magnético estabelecido no entreferro e à corrente que circula pela armadura, conforme expresso pela Equação (4),

$$T_e = K_t \, \phi \, i_a \ . \tag{4}$$

A equação (4), evidencia que, para condições em que o fluxo magnético permanece aproximadamente constante o Torque pode ser controlado diretamente por meio da corrente de armadura. Tal característica confere aos motores de corrente contínua elevada flexibilidade e precisão no controle de Torque e velocidade, tornando-os particularmente adequados para aplicações que exigem desempenho dinâmico controlado e respostas rápidas a variações de carga (KOSOW, 2005).

3.2 Motores CC

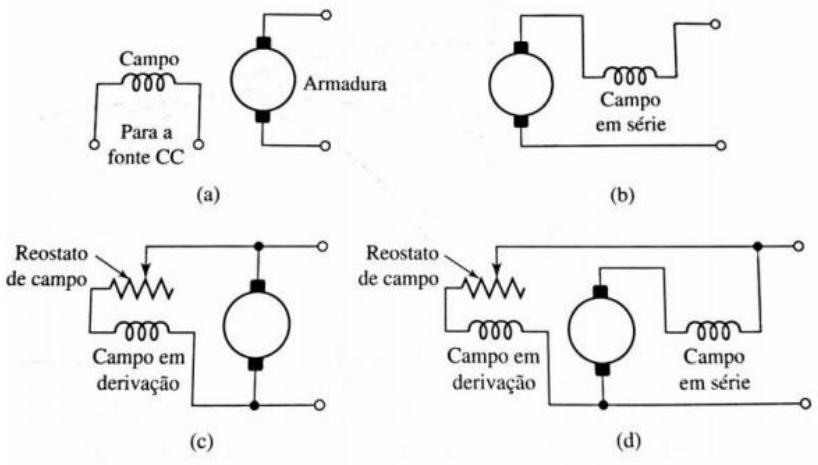
Segundo Patané (2008), os motores de cc apresentam uma ampla variedade de características de funcionamento, envolvendo relações entre tensão e corrente, bem como entre velocidade e conjugado eletromagnético. Essas características estão diretamente associadas à forma de excitação dos enrolamentos de campo, a qual exerce influência significativa sobre o comportamento elétrico e mecânico da máquina, conforme ilustrado na Figura 3.

Os motores de cc podem ser classificados de acordo com o método de excitação do campo magnético em excitação independente ou autoexcitação. Na configuração de excitação independente, os enrolamentos de campo são alimentados por uma fonte externa de corrente contínua, permitindo o controle independente do fluxo magnético e da corrente de armadura. Já na autoexcitação, o campo magnético é alimentado a partir da própria tensão da armadura,

podendo o enrolamento de campo ser conectado em série, em paralelo (*shunt*) ou de forma composta, combinação que resulta em diferentes características de Torque e velocidade para a máquina (FITZGERALD, 2014).

Figura 3: Ligação do circuito de campo de motores CC

(a) Excitação independente; (b) Série; (c) Shunt; (d) Composto.



Fonte: (Fitzgerald, 2014)

O motor de cc de excitação independente é indicado para aplicações que exigem controle preciso e uma ampla faixa de variação da velocidade angular. Nessa configuração, o enrolamento de campo é alimentado por uma fonte externa de tensão contínua, independente do circuito da armadura, o que possibilita o controle separado do fluxo magnético e da corrente de armadura. Essa característica confere elevada flexibilidade ao motor, tornando-o especialmente adequado para sistemas de acionamento de alta performance e para aplicações industriais que demandam controle rigoroso de velocidade e Torque (FITZGERALD, 2014).

O motor de cc série é empregado em aplicações que requerem elevado Torque de partida, como veículos elétricos, guindastes e sistemas de tração metroviária. Nesse tipo de motor, os enrolamentos de campo são conectados em série com o circuito da armadura, de modo que a corrente de campo é a mesma corrente da armadura. Como consequência, o fluxo magnético torna-se proporcional à corrente da armadura, resultando em elevado conjugado eletromagnético em baixas velocidades.

O motor *shunt*, também denominado motor de derivação ou paralelo, possui o enrolamento de campo conectado em paralelo com a armadura. Essa disposição faz com que o fluxo magnético permaneça aproximadamente constante, mesmo diante de variações da carga mecânica. Como resultado, o motor apresenta uma característica de velocidade relativamente estável, sendo indicado para aplicações em que a manutenção da rotação é mais importante do

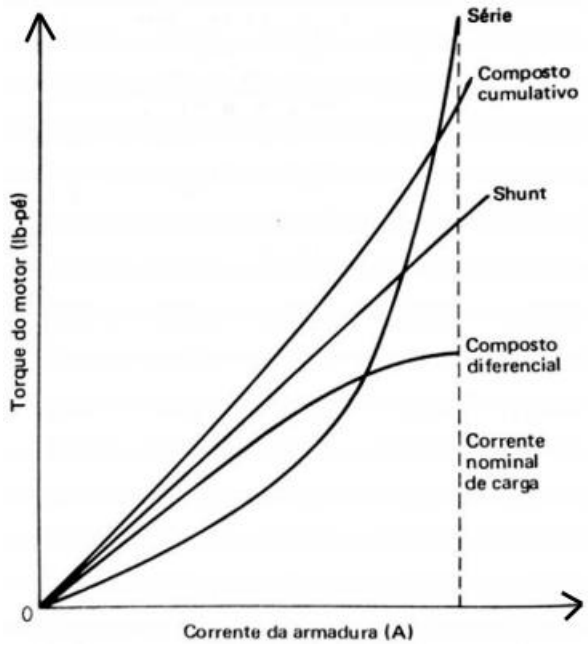
que o Torque de partida, como em ventiladores, bombas e máquinas-ferramenta (FITZGERALD, 2014).

Os motores de cc compostos combinam as características dos motores série e shunt, utilizando simultaneamente um enrolamento de campo em série e outro em paralelo. Esses motores podem ser classificados como compostos cumulativos ou compostos diferenciais, conforme o sentido do fluxo produzido pelo campo série em relação ao campo shunt. No motor composto cumulativo, os fluxos se somam, resultando em maior Torque sob carga, enquanto no motor composto diferencial os fluxos se opõem, produzindo uma característica de Torque menos acentuada.

3.3 Características Torque-Carga

Na Figura 4 apresenta-se uma comparação das características Torque–carga dos diferentes tipos de motores CC operando em regime permanente.

Figura 4: Comparação das características Torque–carga de motores CC.



Fonte: (Kosow, 2005)

Para o motor shunt, considerando-se que o fluxo magnético permanece aproximadamente constante, o Torque eletromagnético varia de forma praticamente linear com a corrente da armadura, conforme a relação apresentada anteriormente na Equação (4). Esse comportamento proporciona boa estabilidade de velocidade frente a variações moderadas da carga mecânica, característica desejável em aplicações que exigem manutenção da rotação sob diferentes condições de operação.

No motor série, a corrente que percorre o enrolamento de campo é a mesma corrente da armadura, fazendo com que o fluxo magnético seja diretamente proporcional à corrente. Nessas condições, o conjugado eletromagnético passa a depender do quadrado da corrente de armadura, e a equação do Torque pode ser expressa pela equação (5)

$$T_e = K_t i_a^2 \quad , \tag{5}$$

resultando em uma característica fortemente não linear, com elevado Torque em baixas velocidades (KOSOW, 2005).

Essa relação evidencia que o motor série desenvolve elevado Torque em baixas velocidades, o que justifica sua aplicação em sistemas que demandam alto Torque de partida. Entretanto, essa característica também torna o motor série inadequado para operação em vazio, uma vez que a velocidade pode aumentar excessivamente na ausência de carga.

Para o motor composto cumulativo, o Torque eletromagnético pode ser expresso por:

$$T_e = K_t(\phi_f + \phi_s) i_a \quad . \tag{6}$$

Nessa configuração, os fluxos do campo *shunt* e do campo série atuam no mesmo sentido, resultando em um fluxo total superior ao do motor *shunt* isoladamente. Como consequência, o motor composto cumulativo apresenta uma curva de Torque sempre mais elevada do que a do motor shunt para uma mesma corrente de armadura, combinando boa capacidade de Torque de partida com relativa estabilidade de velocidade.

No caso do motor composto diferencial, o fluxo produzido pelo enrolamento de campo série se opõe ao fluxo do campo shunt, de modo que o Torque eletromagnético pode ser expresso por:

$$T_e = K_t(\phi_f - \phi_s) i_a \quad . \tag{7}$$

Nessa condição, o fluxo total no entreferro é reduzido à medida que a corrente de armadura aumenta, o que resulta em uma característica de Torque inferior à do motor shunt. Esse comportamento pode comprometer a estabilidade do motor, sobretudo sob variações de carga, razão pela qual os motores compostos diferenciais possuem aplicação prática limitada (KOSOW, 2005).

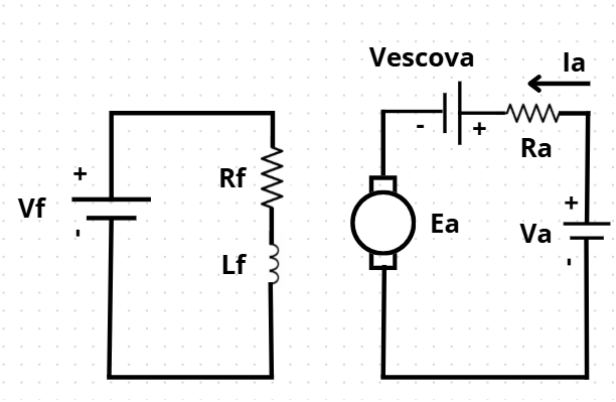
3.4 Equação Da Velocidade E Modelo Dinâmicos Do Motor CC

Segundo Chapman (2013), o circuito equivalente de uma máquina cc pode ser representado conforme ilustrado na Figura 5.

Nesse modelo, o circuito da armadura é caracterizado pela tensão aplicada à armadura V_A, pela resistência da armadura R_A e pela força contraeletromotriz E_A, a qual é gerada pelo movimento dos condutores da armadura no campo magnético. A queda de tensão nas escovas é, em geral, relativamente pequena e, em muitas análises, pode ser desprezada ou incorporada ao valor da resistência da armadura, sem prejuízo significativo da precisão do modelo.

O circuito de campo, por sua vez, é representado pela resistência do enrolamento de campo R_F e pela indutância do enrolamento de campo L_F, enquanto V_F corresponde à tensão aplicada ao circuito de excitação. Esses elementos são responsáveis pela produção do fluxo magnético principal da máquina, que exerce influência direta sobre o Torque eletromagnético e sobre a força contraeletromotriz desenvolvida no circuito da armadura.

Figura 5: Circuito equivalente de um motor CC



Fonte: Próprio autor.

Aplicando-se a Lei de Kirchhoff das Tensões ao circuito da armadura e desprezando-se a queda de tensão nas escovas, obtém-se a equação (8),

$$V_A - E_A = I_A R_A \quad . \tag{8}$$

Sabendo-se que a força contraeletromotriz é proporcional ao fluxo magnético no entreferro e à velocidade angular do rotor, essa grandeza pode ser expressa por:

$$E_A = K\phi \, , \tag{9}$$

em que K é uma constante dependente das características construtivas da máquina, ϕ é o fluxo magnético e ω_n representa a velocidade angular do rotor.

$$\omega_n = \frac{V_A - I_A R_A}{K\phi} \quad . \tag{10}$$

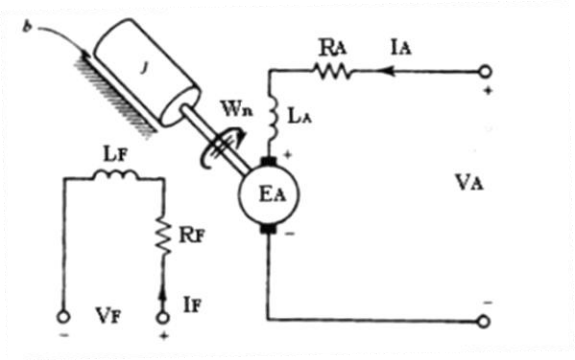
A equação (10), evidencia a relação direta entre a velocidade do motor e a tensão aplicada à armadura, sob a hipótese de fluxo magnético constante. Tal condição é típica de motores de corrente contínua com excitação independente ou de ímã permanente, nos quais a regulação da velocidade pode ser realizada de forma eficiente por meio do ajuste da tensão da armadura.

3.5 Características Dinâmicas Do Motor Cc

Em aplicações de controle de velocidade, é fundamental considerar a forma como o motor responde a variações na tensão de alimentação, na carga mecânica e nas condições iniciais de operação, aspectos diretamente relacionados à dinâmica do sistema eletromecânico (PATANÉ, 2008).

Na Figura 6 apresenta-se o circuito equivalente dinâmico do motor cc, no qual é incluída a indutância da armadura L_A. A consideração desse elemento permite uma modelagem mais realista do comportamento transitório da corrente de armadura, sendo essencial para a análise da resposta do motor a variações rápidas na tensão aplicada e para o projeto de controladores em malha fechada.

Figura 6: Circuito equivalente dinâmico de um motor CC



Fonte: Adaptado de Patané (2008).

A equação dinâmica do circuito da armadura pode ser descrita pela equação (11),

$$V_A = E_A + I_A R_A + L_A \frac{dI_A}{dt} \ . \tag{11}$$

Considerando-se a corrente de campo constante, a força contraeletromotriz assume a forma:

$$E_A = K_w \omega_n \ . \tag{12}$$

No circuito de campo, a equação dinâmica é dada por:

$$V_F = I_F R_F + L_F \frac{dI_F}{dt} \ . \tag{13}$$

Do ponto de vista mecânico, o Torque eletromagnético desenvolvido deve equilibrar o Torque resistente, o atrito viscoso e a variação de energia cinética associada à inércia do sistema:

$$T_e = J\frac{d\omega_n}{dt} + b\omega_n + T_m \ , \tag{14}$$

reorganizando os termos, tem se:

$$T_e - T_m = J\frac{d\omega_n}{dt} + b\omega_n \, . \tag{15}$$

Aplicando a Transformada de Laplace às equações mecânicas, assumindo condições iniciais nulas, obtém-se:

$$T_e(s) - T_m(s) = (Js + b)\,\omega_n(s) \, . \tag{16}$$

Para condições de fluxo constante, o Torque eletromagnético é proporcional à corrente da armadura:

$$T_e = K_t I_A \ . \tag{17}$$

Substituindo a Equação (17) na Equação (16):

$$\frac{\omega_n(s)}{K_t I_A(s) - T_m(s)} = \frac{1}{Js+b} \ . \tag{18}$$

A relação entre corrente de armadura, tensão aplicada e velocidade angular resulta de:

$$V_A - E_A = I_A R_A + L_A\frac{dI_A}{dt} \ . \tag{19}$$

Aplicando a Transformada de Laplace e isolando a corrente de armadura:

$$I_A(s) = \frac{V_A(s) - K_w\omega_n(s)}{R_A + sL_A} \ . \tag{20}$$

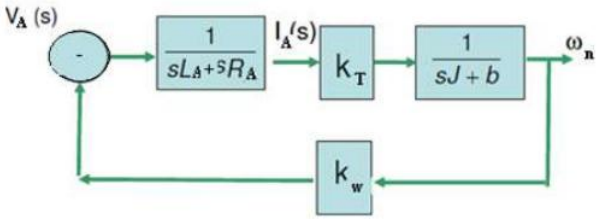
Sabendo que a função de transferência mostra como o sistema responde a uma determinada entrada, permitindo prever o comportamento dinâmico do sistema sem precisar resolver diretamente as equações diferenciais que o descrevem, ao qual é expressa por: G(s)=U(s)Y(s), e desprezando o Torque perturbador $T_m(s)$, obtém-se a função de transferência que relaciona a velocidade angular do rotor à tensão aplicada na armadura:

$$\frac{\omega_n(s)}{V_A(s)} = \frac{K_t}{(R_A + sL_A)(Js+b) + K_t K_w} \ . \tag{21}$$

As equações desenvolvidas neste capítulo permitem descrever matematicamente a dinâmica do motor CC e servem como fundamento para a elaboração de seu diagrama de blocos. Dessa forma, as Figuras 7 e 8 apresentam a representação do modelo dinâmico do motor

com excitação independente, destacando as relações entre as variáveis elétricas e mecânicas do sistema, bem como a influência da tensão de armadura sobre a corrente, o torque desenvolvido e a velocidade angular do rotor.

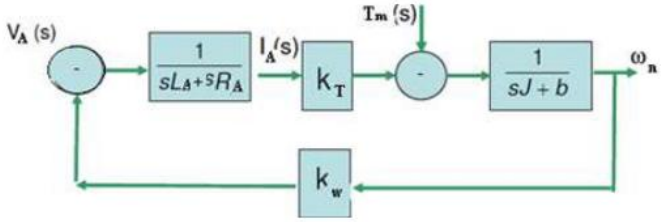
Figura 7: Diagrama de blocos do motor CC com excitação independente



Fonte: Adaptado de Patané (2008).

Ao considerar-se a atuação do Torque de carga $T_m(s)$, observa-se que o seu aumento tende a provocar a redução da velocidade angular do eixo do motor. Para compensar esse efeito e manter a velocidade no valor desejado, torna-se necessário o incremento da tensão aplicada à armadura, de modo a elevar o Torque eletromagnético desenvolvido. Essa compensação é tipicamente realizada por meio de sistemas de controle em malha fechada, os quais ajustam automaticamente a entrada do sistema em função do erro entre a velocidade de referência e a velocidade medida. A análise e o desenvolvimento dessas estratégias de controle constituem o foco dos capítulos seguintes deste trabalho.

Figura 8: Diagrama de blocos do motor CC com excitação independente com $\boldsymbol{T_m(s)}$



Fonte: Adaptado de Patané (2008).

3.6 Sistema De Controle

Com o avanço tecnológico, os sistemas passaram a apresentar níveis cada vez maiores de complexidade, frequentemente envolvendo múltiplas variáveis de entrada e saída, o que tornou a modelagem e o projeto de sistemas de controle matematicamente mais elaborados. Nesse contexto, a teoria clássica de controle, voltada principalmente para sistemas de entrada única e saída única, mostrou-se limitada para atender às novas exigências das aplicações modernas. A partir da década de 1960, impulsionada pela evolução dos computadores digitais e pela viabilidade da análise no domínio do tempo, surgiu a teoria de controle moderno, fundamentada na representação em espaço de estados e no uso de variáveis de estado, com o

objetivo de tratar sistemas mais complexos e atender a requisitos mais rigorosos de precisão, desempenho e custo, especialmente nos setores industrial, militar e espacial (OGATA, 2010).

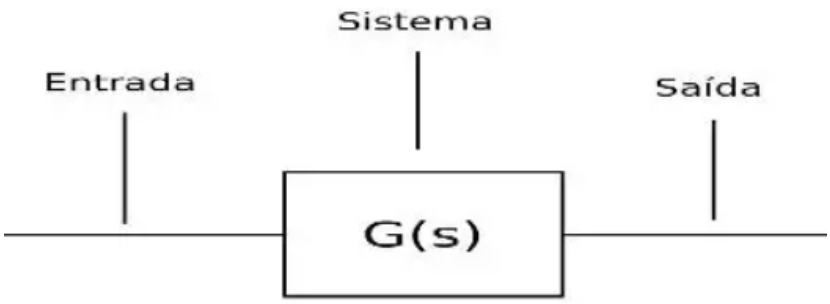
A teoria de controle moderno baseia-se na análise temporal de modelos matemáticos descritos por sistemas de equações diferenciais, que representam de forma mais fiel o comportamento dinâmico dos sistemas físicos reais. Essa abordagem contribuiu para tornar o projeto de controladores mais sistemático e alinhado à realidade do processo controlado. Entretanto, a presença de incertezas e diferenças entre o modelo matemático e o sistema real pode comprometer a estabilidade e o desempenho do controle. Para mitigar esses efeitos, desenvolveu-se a teoria do controle robusto, cujo foco está no projeto de controladores capazes de assegurar estabilidade e desempenho aceitáveis mesmo diante de variações paramétricas e erros de modelagem previamente considerados (OGATA, 2010).

Nesse contexto, um sistema de controle pode ser compreendido como um conjunto de elementos interconectados que operam de forma coordenada para comandar, regular ou supervisionar o comportamento de um sistema dinâmico, de modo que sua saída acompanhe um valor de referência estabelecido. Esse processo normalmente envolve a comparação entre a saída do sistema e o sinal de referência, podendo empregar mecanismos de realimentação para minimizar erros, rejeitar perturbações externas e garantir estabilidade e desempenho adequados (OGATA, 2010).

3.6.1 Controle de malha aberta

Os sistemas de controle de malha aberta são aqueles em que a ação de controle não depende da saída do sistema, ou seja, o sinal de saída não é medido nem comparado com um valor de referência. Nesse tipo de sistema, cada entrada corresponde a uma condição pré-definida de funcionamento, sendo necessária uma calibração adequada para que a saída atenda ao desempenho esperado. Como não há realimentação, esses sistemas não corrigem automaticamente erros causados por distúrbios externos ou variações internas dos parâmetros do processo (OGATA, 2010). Entre suas principais vantagens destacam-se a simplicidade de implementação, o baixo custo, a facilidade de manutenção e a ausência de problemas de estabilidade. Entretanto, suas desvantagens incluem maior sensibilidade a perturbações e a necessidade de ajustes periódicos para manter a qualidade da saída, o que limita sua aplicação a sistemas onde as condições de operação são bem conhecidas e previsíveis.

Figura 9: Sistema de malha aberta



Fonte: Autoria própria

3.6.2 Controle de malha fechada

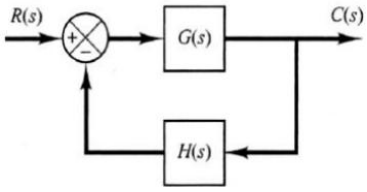
Os sistemas de controle de malha fechada, também denominados sistemas de controle com realimentação, utilizam a saída do sistema como parte do processo de controle. Nesse caso, a saída é medida e comparada com um sinal de referência, gerando um sinal de erro, que corresponde à diferença entre esses dois sinais. Esse erro é utilizado pelo controlador para ajustar a ação de controle, de forma a minimizar desvios e conduzir a saída ao valor desejado (OGATA, 2010).

Um exemplo clássico é o controle de temperatura de um ambiente, no qual um termostato compara a temperatura medida com a temperatura desejada e aciona sistemas de aquecimento ou resfriamento conforme necessário. Sistemas de malha fechada são amplamente utilizados não apenas na engenharia, mas também em sistemas biológicos, como o corpo humano, que mantém variáveis como temperatura corporal e pressão sanguínea por meio de mecanismos de realimentação fisiológica, tornando o sistema menos sensível a perturbações externas. Em geral, a análise desses sistemas é realizada no domínio da frequência por meio de funções de transferência, sendo a relação entre saída e entrada dada por:

$$\frac{\mathrm{C}(s)}{\mathrm{R}(s)} = \frac{G(s)}{1+\mathrm{G}(s)*\mathrm{H}(s)} \quad , \tag{22}$$

sendo $R(s)$ a entrada, $C(s)$ a saída, $G(s)$ a função de transferência do ramo direto e $H(s)$ a função de transferência da realimentação, conforme apresentado na Figura 10.

Figura 10: Sistema de malha fechada



Fonte: Autoria própria

3.6.3 Controle de malha fechada vs controle de malha aberta

A principal diferença entre os sistemas de malha aberta e malha fechada está na presença da realimentação. Enquanto o controle de malha aberta apresenta estrutura mais simples e não requer medição da saída, ele é altamente sensível a distúrbios e variações dos parâmetros do sistema. Por outro lado, o controle de malha fechada proporciona uma resposta mais robusta, sendo capaz de reduzir o efeito de perturbações externas e variações internas, mantendo a saída próxima do valor desejado. No entanto, sistemas de malha fechada são mais complexos, possuem maior custo e exigem cuidados adicionais com relação à estabilidade. Dessa forma, o controle de malha aberta é indicado para sistemas nos quais a entrada é conhecida antecipadamente e as perturbações são desprezíveis, enquanto o controle de malha fechada é mais adequado para sistemas sujeitos a distúrbios e incertezas, nos quais se deseja maior precisão e confiabilidade (OGATA, 2010).

3.6.4 Análise da resposta transitória

A análise de um sistema de controle exige, como etapa fundamental, a obtenção de um modelo matemático capaz de representar adequadamente o comportamento dinâmico do sistema. A partir desse modelo, torna-se possível avaliar o desempenho do sistema utilizando diferentes métodos analíticos. Entretanto, em situações práticas, o sinal de entrada de um sistema de controle nem sempre é conhecido antecipadamente, o que torna necessária a definição de critérios padronizados que permitam a comparação entre diferentes sistemas. Essa referência é normalmente estabelecida por meio da aplicação de sinais de entrada de teste, cujas respostas possibilitam inferir o comportamento do sistema diante de condições reais de operação. A utilização desses sinais se justifica pela correlação existente entre a resposta do sistema a entradas de teste padronizadas e sua capacidade de responder a sinais reais. Entre os sinais mais empregados destacam-se o degrau, a rampa, a parábola de aceleração, o impulso e os sinais senoidais, uma vez que são funções simples do tempo, facilitando tanto a análise matemática quanto a avaliação experimental dos sistemas de controle.

De acordo com Teixeira e Assunção (2016), a função de transferência de um sistema linear e invariante no tempo é definida como a razão entre a transformada de Laplace da saída e a transformada de Laplace da entrada, considerando-se todas as condições iniciais nulas. Em termos práticos, essa função é obtida a partir da equação diferencial que descreve o sistema, formulada com base em princípios físicos, mecânicos, elétricos ou matemáticos. Aplicando-se a transformada de Laplace a essa equação e isolando-se a relação entre a saída e a entrada, obtém-se a função de transferência na forma $\frac{Y(s)}{U(s)}$, (OGATA,2010).

Em grande parte das aplicações, as especificações de desempenho de um sistema de controle são definidas no domínio do tempo, uma vez que sistemas físicos que armazenam energia, isto é, que possuem condições iniciais diferentes de zero, não respondem instantaneamente às variações de entrada ou a perturbações externas. Nessas situações, o sistema apresenta inicialmente uma resposta transitória, até atingir o regime permanente. Por esse motivo, a avaliação do desempenho é frequentemente realizada a partir da resposta a uma entrada em degrau unitário, por se tratar de um sinal de fácil implementação e suficientemente abrupto para evidenciar as principais características dinâmicas do sistema.

Os sistemas de segunda ordem são descritos por equações diferenciais que envolvem apenas a primeira e a segunda derivadas da variável de saída. O comportamento dinâmico desses sistemas é determinado por dois parâmetros principais: o coeficiente de amortecimento ζ e a frequência natural ω_n. Quando $0 < \zeta < 1$, os polos da malha fechada são complexos conjugados e situam-se no semiplano esquerdo do plano s, caracterizando um sistema subamortecido, cuja resposta transitória apresenta oscilações amortecidas. Para $\zeta = 0$, o sistema não possui amortecimento, resultando em oscilações contínuas. No caso em que $\zeta = 1$, o sistema é denominado criticamente amortecido, apresentando a resposta mais rápida possível sem oscilações. Já para valores de $\zeta > 1$, o sistema é classificado como superamortecido, com resposta lenta e não oscilatória.

A função de transferência típica de um sistema de segunda ordem é expressa pela Equação (23):

$$\frac{Y(s)}{X(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad . \tag{23}$$

Antes de atingir o regime permanente, a resposta transitória de um sistema de controle pode apresentar oscilações amortecidas, sendo caracterizada por parâmetros temporais específicos.

Entre esses parâmetros destacam-se o tempo de acomodação (t_e), definido como o tempo necessário para que a resposta permaneça dentro de uma faixa de erro pré-estabelecida em torno do valor final, geralmente de 2% ou 5%, e que pode ser estimado por

$$\text{Tempo de acomodação} = \frac{4{,}6}{\omega_n \zeta} \ . \tag{24}$$

Outro parâmetro importante é o tempo de subida (t_r), que corresponde ao intervalo necessário para que a resposta varie entre determinados percentuais do valor final, normalmente de 10% a 90% ou de 0% a 100%, dependendo do tipo de sistema analisado.

O tempo de pico (t_p) representa o instante em que ocorre o primeiro pico de sobressinal da resposta transitória e é dado pela expressão:

$$T_p = \frac{\pi}{\omega_n \sqrt{1-\zeta^2}} \ . \tag{25}$$

Observa-se que esse parâmetro é inversamente proporcional à frequência natural e depende diretamente do coeficiente de amortecimento, de modo que sistemas com maior ω_n apresentam respostas mais rápidas, enquanto valores menores de ζ aumentam a tendência a oscilações.

Outro indicador relevante é o máximo sobressinal (*Overshoot*), representado por M_p, que expressa o valor máximo da resposta acima do valor final e está diretamente relacionado à estabilidade relativa do sistema. Esse parâmetro pode ser calculado por meio da expressão:

$$\text{Porcentagem de } \textit{Overshoot}.(\%) = 100\, e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}} \ . \tag{26}$$

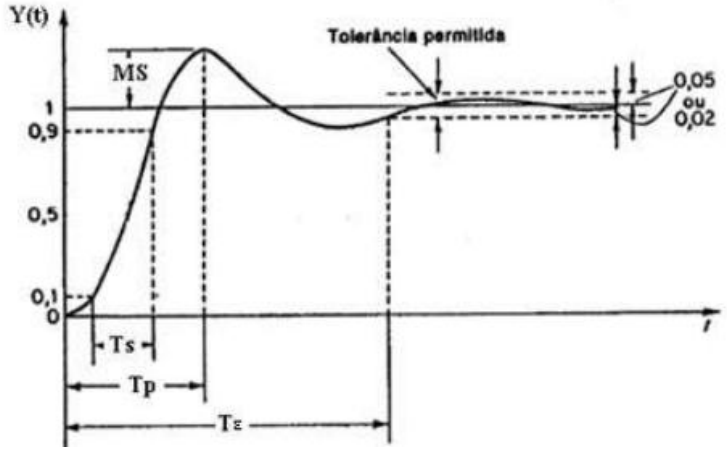
Além disso, o tempo de subida pelo critério simplificado pode ser aproximado por:

$$T_s = \frac{1{,}8}{\omega_n} \ , \tag{27}$$

indicando a relação inversa entre a rapidez da resposta e a frequência natural do sistema.

Essas especificações são fundamentais, uma vez que a maioria dos sistemas de controle opera no domínio do tempo e deve apresentar respostas transitórias adequadas aos requisitos do projeto. Dessa forma, o sistema de controle deve ser ajustado ou modificado até que seu comportamento dinâmico atenda aos critérios de desempenho estabelecidos. Os parâmetros discutidos são geralmente representados de forma gráfica, conforme ilustrado na Figura 11, que apresenta a curva típica de resposta a um degrau unitário.

Figura 11: Curva típica de resposta impulsiva de um sistema de segunda ordem



Fonte: Adaptado de Teixeira e Assunção (2016)

3.7 Tipos De Controladores

Segundo Ogata (2010), existem vários tipos de controle para malha fechada, como exemplo a ser abordados nesta seção, proporcional P, integral I, proporcional-integral PI, proporcional-derivativo PD e proporcional-integral-derivativo PID.

3.7.1 Controle proporcional (P)

O controlador proporcional (P) atua diretamente sobre o erro do sistema, aplicando um ganho que ajusta a intensidade da ação de controle de acordo com a magnitude desse erro. Ao amplificar o sinal de erro, esse tipo de controlador contribui para a melhoria do desempenho do sistema, reduzindo o erro em regime permanente e mantendo a estabilidade em uma ampla faixa de operação. Uma de suas principais vantagens é a simplicidade de implementação, além da capacidade de diminuir significativamente o erro estacionário. Em contrapartida, o aumento excessivo do ganho proporcional pode provocar respostas mais lentas, elevando o tempo de acomodação e, em determinadas condições, comprometendo a estabilidade do sistema.

Para um controlador com ação proporcional, a relação entre o sinal de saída do controlador $u(t)$e o sinal de erro $e(t)$ no domínio do tempo é dada por:

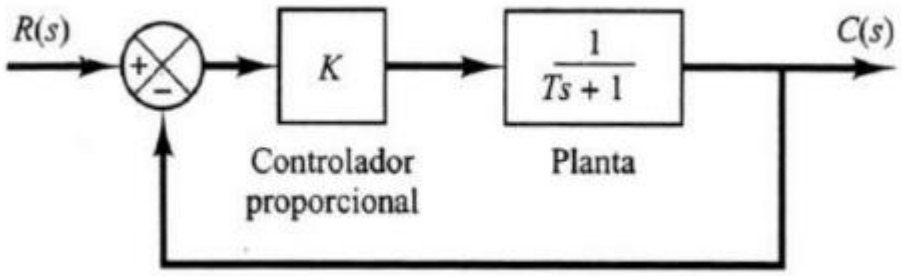
$$u(t) = K_p e(t) \,. \tag{28}$$

Aplicando-se a transformada de Laplace, obtém-se a relação no domínio da frequência:

$$\frac{U(s)}{E(s)} = K_p \,. \tag{29}$$

De forma simplificada, o ganho proporcional pode ser interpretado como um amplificador ajustável, cujo valor determina o nível de correção aplicado ao sistema. À medida que o ganho proporcional é aumentado, o erro em estado estacionário tende a diminuir; entretanto, valores elevados de K_p podem resultar em respostas excessivamente lentas ou até mesmo instáveis. Ressalta-se que o controle proporcional não altera a ordem do sistema, pois não introduz novos polos ou zeros, atuando apenas como um ajuste do ganho global do sistema original, conforme discutido na literatura especializada (OGATA,2010).

Figura 12: Diagrama de blocos do Controle proporcional



Fonte: Próprio autor.

3.7.2 Controle integral

O controlador integral (I) atua acumulando o erro do sistema ao longo do tempo, fazendo com que a ação de controle seja ajustada de forma contínua enquanto existir diferença entre o valor desejado e o valor real da saída. Essa característica permite que o erro em regime permanente seja eliminado, uma vez que qualquer erro persistente resulta no aumento progressivo da ação de controle. Em contrapartida, a utilização isolada da ação integral pode comprometer a estabilidade do sistema, pois a acumulação excessiva do erro tende a gerar respostas lentas ou instáveis, especialmente em sistemas com dinâmica mais sensível (OGATA,2010).

No controle integral, o sinal de saída do controlador $u(t)$ é obtido a partir da integração do erro ao longo do tempo, conforme expresso por:

$$u(t) = K_i \int_0^t e(t)\, dt \ , \tag{30}$$

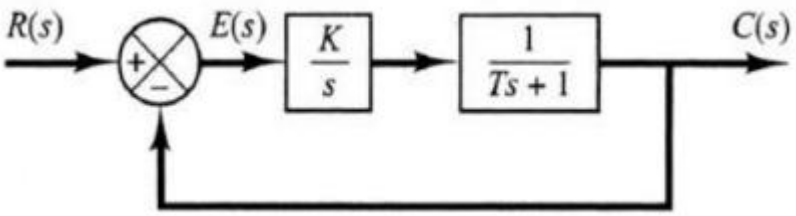
em que K_i representa o ganho integral, responsável por ajustar a intensidade da ação acumulativa do controlador.

Ao aplicar a transformada de Laplace à equação anterior, considerando condições iniciais nulas, obtém-se a função de transferência do controlador integral, dada por:

$$\frac{U(s)}{E(s)} = \frac{K_i}{s} \ . \tag{31}$$

Nessa expressão, $U(s)$ corresponde ao sinal de saída do controlador no domínio da frequência, enquanto $E(s)$ representa o sinal de erro. Observa-se que o controlador integral introduz um polo na origem do plano s, o que explica sua capacidade de eliminar o erro estacionário, mas também justifica a maior propensão à instabilidade quando utilizado sem a combinação com outras ações de controle (OGATA,2010).

Figura 13: Diagrama de blocos Controle Integral



Fonte: Próprio autor.

3.7.3. Controle proporcional-integral

O controlador (PI) combina as características do controle proporcional e do controle integral, reunindo a capacidade do proporcional de garantir estabilidade ao sistema com a eficiência do integrador na eliminação do erro em regime permanente. Essa atuação conjunta ocorre porque, mesmo quando o erro instantâneo se torna pequeno, a ação integral continua acumulando esse erro ao longo do tempo, aumentando gradualmente o sinal de controle até que o atuador seja efetivamente acionado. À medida que o sistema se aproxima do valor de referência, a contribuição do termo proporcional diminui, reduzindo sua influência sobre a resposta, enquanto a ação integral passa a dominar o comportamento do sistema, assegurando a correção final do erro.

Assim como no controle proporcional, o controlador PI possui um ganho associado à ação proporcional, e um parâmetro relacionado à ação integral, normalmente expresso por meio do tempo integrativo T_i. O ajuste adequado desses parâmetros é essencial para o bom desempenho do sistema, uma vez que valores elevados da ação integral podem tornar a resposta excessivamente lenta durante as transições, comprometendo o desempenho dinâmico do sistema.

A ação de controle do controlador PI no domínio do tempo é descrita pela Equação (32):

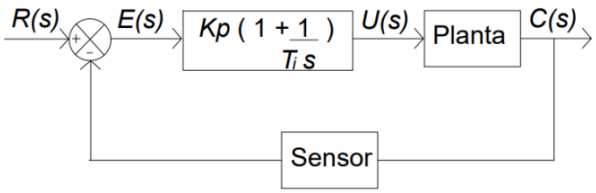
$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(t)\,dt \ . \tag{32}$$

No domínio da frequência, a função de transferência do controlador PI é dada por:

$$\frac{U(s)}{E(s)} = K_p\left(1+\frac{1}{T_i s}\right) . \tag{33}$$

Na Figura 14 apresenta-se o diagrama de blocos correspondente à aplicação de um controlador PI em um sistema de controle.

Figura 14: Esquema de implementação de um controlador tipo PI



Fonte: Adaptado de Ogata (2010).

3.7.4 Controle proporcional-derivativo

A associação da ação derivativa ao controle proporcional resulta no chamado controlador PD, cuja principal característica é o aumento da sensibilidade do sistema às variações do erro, proporcionando uma melhoria significativa no comportamento em regime transitório. A ação derivativa baseia-se na taxa de variação do erro, permitindo que o controlador antecipe a tendência do sistema e atue de forma preventiva, corrigindo a resposta antes que o erro atinja valores elevados. Essa característica contribui para a redução de oscilações e para uma resposta mais rápida durante as transições.

Entretanto, a ação derivativa apresenta como limitação sua elevada sensibilidade a ruídos, uma vez que variações rápidas e indesejadas no sinal de erro podem ser interpretadas como mudanças bruscas no sistema, levando a ações incorretas de controle. Além disso, quando utilizada de forma inadequada, a ação derivativa pode comprometer a estabilidade do sistema, razão pela qual, na prática, raramente é empregada de forma isolada.

A expressão matemática que descreve a ação de controle de um controlador PD no domínio do tempo é dada por:

$$u(t) = K_p e(t) + K_p T_d \frac{de(t)}{dt} \ . \tag{34}$$

No domínio da frequência, a função de transferência do controlador PD pode ser expressa por:

$$\frac{U(s)}{E(s)} = K_p(1 + T_d s) \ . \tag{35}$$

Observa-se que a ação derivativa introduz um zero no sistema, contribuindo para a melhoria da resposta transitória, especialmente no que se refere à redução do sobressinal e ao aumento da velocidade de resposta (OGATA,2010).

3.7.5 Controle proporcional integral derivativo

O controlador proporcional–integral–derivativo (PID) é amplamente reconhecido como uma solução versátil e eficiente em sistemas de controle, pois reúne em uma única estrutura as ações proporcional, integral e derivativa. Apesar de apresentar excelente desempenho em diversas aplicações, o controlador PID também é a alternativa mais complexa e onerosa, uma vez que exige o ajuste simultâneo de três parâmetros distintos: o ganho proporcional (K_p), o ganho integral (K_i) e o ganho derivativo (K_d). A combinação adequada desses parâmetros influencia diretamente o comportamento do sistema, determinando características como rapidez da resposta, presença ou ausência de oscilações e nível de estabilidade.

O processo de sintonia de um controlador PID não é trivial e, geralmente, é realizado por meio de modelagem matemática e simulações computacionais, nas quais são avaliados critérios de desempenho como máximo sobressinal, tempo de acomodação, erro em regime permanente e resposta transitória. A escolha apropriada dos ganhos permite obter um equilíbrio entre rapidez e estabilidade, conforme discutido na literatura especializada.

A grande vantagem dos controladores PID está em sua ampla aplicabilidade prática, sendo especialmente úteis em situações nas quais o modelo matemático da planta não é completamente conhecido, inviabilizando o uso de métodos analíticos clássicos de projeto.

A ação de controle de um controlador PID no domínio do tempo pode ser descrita pela Equação (36).

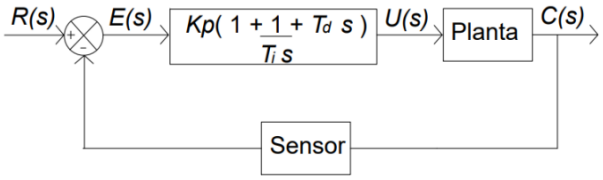
$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(t)\,dt + K_p T_d \frac{de(t)}{dt} \ . \tag{36}$$

No domínio da frequência, a função de transferência correspondente é dada pela Equação (37).

$$\frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s \ \ . \tag{37}$$

Uma característica importante do controlador PID (Figura 15) é sua flexibilidade, pois, ao anular-se um ou mais ganhos, é possível obter controladores mais simples a partir da mesma estrutura. Por exemplo, ao zerar o ganho derivativo, obtém-se um controlador PI; ao eliminar os ganhos integral e derivativo, resulta-se em um controlador puramente proporcional.

Figura 15: Esquema de implementação de um controlador tipo PID



Fonte: Adaptado de Ogata (2013).

4. LOGICA FUZZY

Neste tópico são fornecidos os principais conceitos necessários para o entendimento do algoritmo implementado para fazer o controle da velocidade do motor cc.

Os seres humanos possuem a capacidade de lidar com processos complexos mesmo quando as informações disponíveis são imprecisas, vagas ou incompletas. Em diversas situações, as decisões humanas não se baseiam em valores numéricos exatos, mas em avaliações qualitativas expressas por termos linguísticos, tais como baixo, adequado ou elevado. Dessa forma, as estratégias adotadas por operadores humanos tendem a ser subjetivas e imprecisas, porém eficazes na resolução de problemas do mundo real.

Nesse contexto, a Teoria dos Conjuntos *Fuzzy* e a Lógica *Fuzzy* possibilitam a tradução dessas informações linguísticas imprecisas em modelos matemáticos formais. Por meio da representação do conhecimento na forma de regras linguísticas do tipo “se...então”, torna-se possível construir algoritmos computacionais capazes de simular o raciocínio humano. Quando um especialista consegue expressar sua estratégia de decisão por meio desse tipo de regra, viabiliza-se a implementação de um sistema de inferência Fuzzy, no qual as regras são avaliadas e combinadas de forma sistemática.

Os fundamentos da Lógica *Fuzzy*, também conhecida como lógica nebulosa, foram propostos por Lotfi Asker Zadeh, na década de 1960. Diferentemente da lógica clássica, que opera com valores binários (verdadeiro ou falso), a lógica *Fuzzy* admite graus intermediários de verdade, representados por valores contínuos no intervalo [0 1]. Essa característica permite uma modelagem mais realista de fenômenos complexos e incertos, aproximando o processo

computacional da forma como os seres humanos interpretam e tomam decisões em ambientes reais.

Assim, a lógica *Fuzzy* fornece um arcabouço teórico e matemático adequado para o desenvolvimento de sistemas de apoio à decisão, classificação e controle, especialmente em aplicações nas quais a incerteza, a subjetividade e a não linearidade desempenham papel fundamental.

4.1 Conjuntos Fuzzy

A teoria clássica dos conjuntos, a pertinência de um elemento a um conjunto é definida de forma precisa. Dado um conjunto A pertencente a um universo X, um elemento pode apenas pertencer ou não a esse conjunto, relação representada pela função característica $\mu_A(x)$, que assume exclusivamente os valores 0 ou 1.

Em contraposição a essa abordagem, Zadeh propôs a generalização da função característica, permitindo que ela assumisse qualquer valor real no intervalo $[0, 1]$. Assim, um conjunto *Fuzzy* F, definido em um universo U, é caracterizado por uma função de pertinência μ_F, que associa a cada elemento $x \in U$ um grau de pertinência, dada pela expressão 38:

$$\mu_F: U \to [0,1] \quad . \tag{38}$$

Os valores $\mu_F(x) = 1$ e $\mu_F(x) = 0$ indicam, respectivamente, pertinência total e ausência de pertinência do elemento ao conjunto *Fuzzy.* Ressalta-se que um conjunto clássico pode ser interpretado como um caso particular de conjunto *Fuzzy,* no qual a função de pertinência assume apenas valores binários, dada pela expressão 39:

$$\mu_A: U \to \{0,1\}\,. \tag{39}$$

Um conjunto *Fuzzy* A pode ainda ser representado como um conjunto de pares ordenados:

$$A = \left\{ \frac{\mu_A(x)}{x} \;\middle|\; x \in X \right\}. \tag{40}$$

O suporte de um conjunto Fuzzy corresponde ao conjunto de elementos para os quais $\mu_A(x) > 0$. Quando o suporte contém apenas um elemento com grau de pertinência igual a 1, o conjunto é denominado *singleton*.

Para universos discretos, um conjunto Fuzzy pode ser representado pela expressão 41:

$$A = \frac{\sum_{i=1}^{n} \mu_A(x_i)}{x_i}\,. \tag{41}$$

Para universos contínuos, utiliza-se frequentemente a forma integral:

$$A = \frac{\int \mu_A(x)}{x}, \tag{42}$$

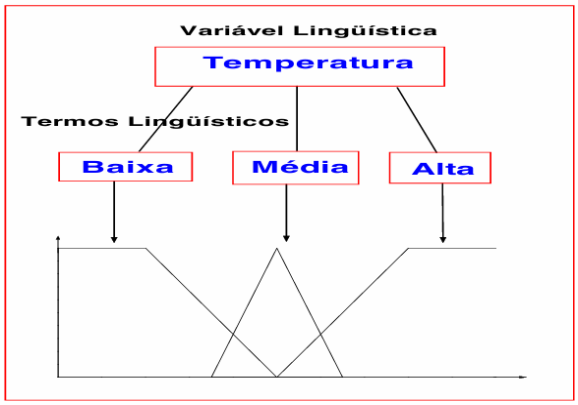
em que os símbolos de soma e integral não representam operações algébricas convencionais, mas sim notações simbólicas para descrever a associação entre elementos e seus respectivos graus de pertinência.

4.2 Variáveis Linguísticas e Funções De Pertinência

Uma variável linguística é definida como uma variável cujos valores não são números, mas termos linguísticos associados a conjuntos Fuzzy. Esses termos representam conceitos qualitativos do mundo real e são modelados por meio de funções de pertinência, que descrevem o grau de pertencimento de um valor numérico a cada termo linguístico.

Por exemplo, a variável temperatura pode assumir valores linguísticos como baixa, média e alta, sendo cada um desses termos representado por um conjunto Fuzzy distinto. Na Figura 16 ilustra-se a representação gráfica típica de uma variável linguística e de suas funções de pertinência.

Figura 16: Representação de uma variável linguística



Fonte: Adaptado de Jafelice(2012).

Formalmente, uma variável linguística pode ser definida pela quíntupla $(N, T(N), X, G, M)$, em que:

- N é o nome da variável linguística;
- $T(N)$ é o conjunto de termos linguísticos associados;
- X é o universo de discurso;
- G representa as regras sintáticas para a geração dos termos linguísticos;

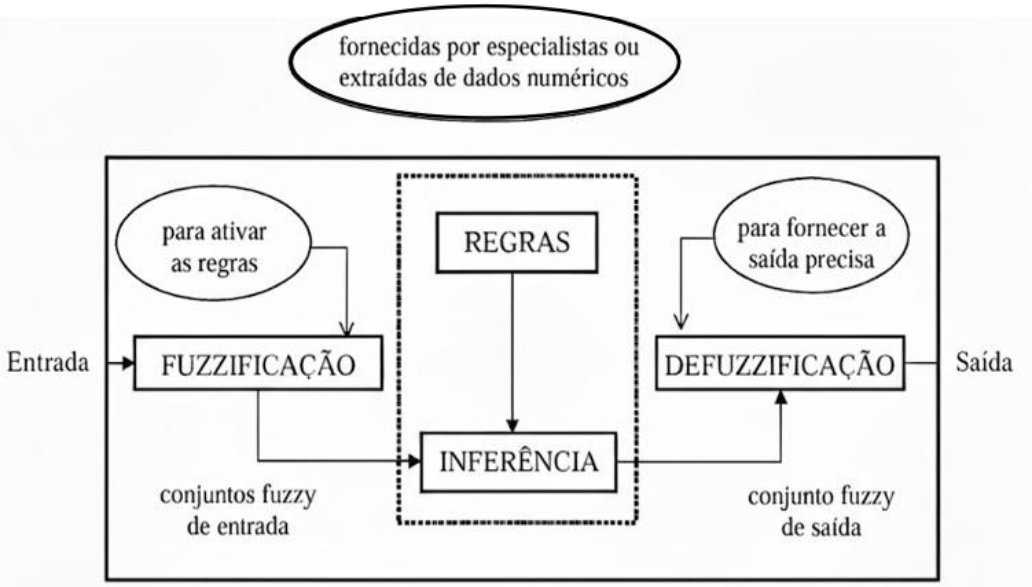
- M define as regras semânticas que associam cada termo linguístico a um conjunto Fuzzy em X (JAFELICE,2012).

As funções de pertinência podem assumir diferentes formatos, sendo os mais comuns os formatos triangular, trapezoidal e gaussiano, utilizados devido à sua simplicidade computacional e facilidade de interpretação. Em aplicações práticas, essas funções podem ser ajustadas iterativamente, com base no desempenho do sistema e na experiência do especialista.

4.3 Sistema De Inferência Fuzzy

Um Sistema de Inferência *Fuzzy (FIS)* é composto, de forma geral, pelas etapas de fuzzificação, inferência e defuzzificação, conforme ilustrado na Figura 17.

Figura 17: Estrutura de um sistema de inferência Fuzzy



Fonte: Elaborado pela autora.

Na etapa de fuzzificação, valores numéricos de entrada são convertidos em graus de pertinência associados a conjuntos Fuzzy. Em seguida, a máquina de inferência realiza o raciocínio aproximado com base na base de regras, geralmente expressas por proposições linguísticas do tipo “se…então”.

Entre os métodos de inferência mais utilizados destacam-se os métodos de Mamdani e Takagi–Sugeno. Neste trabalho, adota-se o método de Mamdani, devido à sua simplicidade conceitual, ampla aceitação e adequação a sistemas baseados em regras linguísticas.

Após o processo de inferência, obtém-se um conjunto Fuzzy de saída, o qual é convertido em um valor numérico por meio do processo de defuzzificação. Dentre os métodos

existentes para a defuzzificação, o método do centroide, utilizado neste trabalho, apresenta a vantagem de produzir respostas suaves e contínuas, sendo especialmente adequado para aplicações de controle.

5. *SOFTWARE*S UTILIZADOS

No desenvolvimento deste trabalho, foram empregados três *Software*s principais: o MATLAB/Simulink, utilizado para modelagem e simulação; o Elipse E3, destinado à supervisão; e a IDE da plataforma Arduino, empregada na programação do microcontrolador.

5.1 Matlab

O MATLAB consiste em um ambiente computacional empregado na área de engenharia, destinado ao desenvolvimento de modelos matemáticos, à simulação computacional e à análise de sistemas dinâmicos. Integrado a esse ambiente, o Simulink disponibiliza uma interface gráfica baseada em diagramas de blocos, a qual permite a construção e a simulação de sistemas de controle de forma visual e intuitiva.

Nesse ambiente, os sistemas são representados por blocos funcionais que, em geral, possuem sinais de entrada, sinais de saída e variáveis de estado associadas ao seu comportamento dinâmico. Essa estrutura possibilita a modelagem de sistemas contínuos, discretos ou híbridos, oferecendo elevada flexibilidade na implementação e na análise de estratégias de controle.

O Simulink também disponibiliza diferentes métodos numéricos para a integração das equações do sistema, permitindo ao usuário avaliar o desempenho do modelo sob distintos critérios de simulação. Além disso, o ambiente possibilita a interação direta com o MATLAB, viabilizando o uso de comandos, rotinas e ferramentas especializadas presentes nos *toolboxes,* ampliando significativamente as possibilidades de análise e desenvolvimento de sistemas de controle.

5.2 Elipse E3

O Elipse E3 é um software supervisório do tipo HMI/SCADA (Interface Homem-Máquina), utilizado para monitoramento e controle de processos industriais. A plataforma permite a aquisição, visualização e registro de dados em tempo real, além da interação com diversos dispositivos de automação, como controladores lógicos programáveis (CLPs), unidades remotas e sistemas de aquisição de dados.

A comunicação com esses equipamentos é realizada por meio de drivers específicos, que implementam diferentes protocolos industriais, possibilitando a troca de informações entre o processo e o sistema supervisório. A partir dos dados coletados, o software permite a criação de interfaces gráficas, alarmes, históricos e relatórios, facilitando a supervisão e a análise do desempenho do sistema.

A arquitetura do Elipse E3 é composta por três módulos principais: o E3 Studio, utilizado para o desenvolvimento das aplicações; o E3 Server, responsável pelo processamento e gerenciamento das informações; e o E3 Viewer, destinado à operação e visualização das telas supervisórias. Essa estrutura proporciona flexibilidade, confiabilidade e facilidade de integração com sistemas industriais e de automação.

5.3 IDE Arduino Uno

Para iniciar o desenvolvimento de aplicações utilizando a plataforma Arduino©, é necessário acessar o ambiente de programação específico da ferramenta, conhecido como Arduino IDE (*Integrated Development Environment*).

Os programas desenvolvidos para o Arduino© são denominados *sketches* e são escritos no editor de código disponibilizado na janela principal da IDE. Esses arquivos podem ser armazenados pelo usuário com uma extensão própria da plataforma, permitindo sua posterior edição, reutilização e organização em projetos distintos. O editor de texto da IDE oferece recursos básicos de edição, como copiar, colar, localizar e substituir trechos de código, além de fornecer mensagens de status relacionadas ao salvamento, à compilação e à exportação dos arquivos. Adicionalmente, a área de mensagens exibe informações detalhadas sobre eventuais erros ou avisos gerados durante o processo de compilação do programa, auxiliando na depuração do código (MULTILÓGICA SHOP, 2018).

Na região inferior da janela da IDE (Figura 18) são mostradas informações referentes à placa selecionada e à porta serial utilizada para comunicação com o computador. Já a barra de ferramentas localizada na parte superior da interface contém botões que permitem, de forma rápida, verificar o código, carregar o programa na placa, criar *sketches*, abrir arquivos existentes, salvá-los e acessar o monitor serial.

Figura 18: Tela inicial da IDE Arduino

Fonte: Autoria própria

Além das funcionalidades básicas de edição e compilação, o ambiente Arduino© possibilita a utilização de bibliotecas, que consistem em conjuntos de funções previamente implementadas para facilitar o acesso a recursos de *Hardware* e a manipulação de dados. Algumas bibliotecas já vêm incorporadas à IDE, enquanto outras podem ser adicionadas pelo usuário por meio de download ou até mesmo desenvolvidas de forma personalizada, conforme o grau de complexidade do projeto.

A linguagem de programação utilizada no Arduino© é baseada em C/C++ e pode ser organizada em três componentes principais: estruturas, valores e funções. As estruturas fundamentais de um sketch são representadas pelas funções setup () e loop (). A função setup () é executada uma única vez no início do programa e é utilizada para inicialização de variáveis, configuração dos modos dos pinos de entrada e saída e declaração do uso de bibliotecas. A função loop (), por sua vez, é executada continuamente, repetindo indefinidamente o conjunto de instruções que define o comportamento principal do sistema durante a operação (ARDUINO, 2018).

As variáveis são utilizadas para armazenar informações temporárias, como leituras de sensores, estados lógicos e valores intermediários de cálculo. Dentre os tipos de variáveis mais comuns, destacam-se as variáveis booleanas, utilizadas para representar valores lógicos verdadeiro ou falso, as variáveis inteiras, empregadas no armazenamento de números inteiros,

e as variáveis do tipo caractere, destinadas à representação de símbolos individuais, como letras e sinais (ARDUINO, 2018).

O controle das entradas e saídas digitais da placa Arduino© é realizado por meio de funções específicas, como pinMode (), utilizada para definir o modo de operação dos pinos, digitalRead (), responsável pela leitura de sinais digitais, e digitalWrite (), empregada para a escrita de níveis lógicos nas saídas digitais. Além disso, a linguagem oferece diversas funções matemáticas e utilitárias, como abs (), constrain () e map (), bem como recursos para geração de números aleatórios, manipulação de entradas e saídas analógicas, operações com bits e bytes, tratamento de interrupções, entre outros (ARDUINO, 2018).

Em função de sua flexibilidade, baixo custo e facilidade de programação, a plataforma Arduino© permite o desenvolvimento de uma ampla variedade de projetos. No contexto deste trabalho, o Arduino© é utilizado em conjunto com o *Software* Elipse E3, através da biblioteca modbus, ao qual determina a função mestre-escravo, que desempenha a função de interface homem–máquina, possibilitando a supervisão, visualização e interação com o sistema de controle implementado.

6. DESENVOLVIMENTO DO PRÓTOTIPO DE CONTROLE DE VELOCIDADE DE UM MOTOR CC

São descritas neste capítulo os componentes de *hardware* e o protótipo implementado.

6.1 Componentes de Hardware utilizados

Considerando critérios como custo, disponibilidade, facilidade de manuseio e praticidade de operação, a escolha de um motor de corrente contínua com tensão nominal de 6 V e velocidade nominal de 210 rpm mostra-se adequada para a realização dos testes dos controladores propostos.

Além disso, esse tipo de motor apresenta características compatíveis com as capacidades de acionamento do Arduino Uno e dos circuitos auxiliares empregados, possibilitando a avaliação do desempenho dos algoritmos de controle em condições representativas, sem a necessidade de equipamentos de grande porte ou de alto custo. Essa abordagem favorece a reprodutibilidade do sistema experimental e reforça o caráter didático e acessível da solução proposta.

Para a realização do experimento, foram escolhidos os seguintes componentes e equipamentos:

- Motor DC 6V (210 RPM) com *encoder* acoplado;
- Arduino UNO;
- Módulo Ponte H;
- Fonte DC 12 V

6.2 Motor de e *Encoder*

Os motores de cc normalmente empregados em aplicações com o microcontrolador Arduino são, em geral, de pequeno porte, operando com baixa tensão e corrente, o que facilita sua integração a sistemas embarcados e ambientes didáticos. Apesar de suas dimensões reduzidas, esses motores apresentam comportamento eletromecânico análogo ao de motores de maior potência, permitindo a extrapolação conceitual dos resultados obtidos para aplicações mais robustas.

Para a utilização de estratégias de controle, torna-se essencial definir adequadamente a variável a ser controlada e as variáveis manipuladas do sistema. Em motores cc, as variáveis manipuladas mais comuns são de natureza elétrica, como a tensão e a corrente da armadura,

uma vez que sua modulação possibilita o controle das variáveis mecânicas associadas ao motor, tais como Torque, posição angular e velocidade angular.

Na maioria das aplicações envolvendo controladores clássicos, como PI, bem como controladores baseados em lógica *Fuzzy*, a variável de interesse é a velocidade angular do eixo. Esse controle é geralmente realizado por meio da variação da tensão aplicada à armadura do motor. No contexto de sistemas baseados em microcontroladores, essa variação é implementada de forma eficiente utilizando sinais de modulação por largura de pulso (PWM), que permitem ajustar o valor médio da tensão aplicada às bobinas do motor.

Sendo o PWM uma técnica utilizada para controlar a potência fornecida a dispositivos elétricos e eletrônicos por meio da variação da largura dos pulsos de um sinal digital. Nessa técnica, a frequência do sinal permanece constante, enquanto o tempo em que o sinal permanece em nível alto é ajustado, alterando-se o chamado ciclo de trabalho (*duty cycle*). Quanto maior for a largura do pulso, maior será o valor médio da tensão aplicada à carga e, consequentemente, maior será a potência fornecida ao sistema. Em motores de corrente contínua, o PWM é frequentemente empregado para controlar a velocidade de rotação, permitindo ajustar a tensão média aplicada à armadura sem a necessidade de dissipar potência em elementos resistivos. Dessa forma, obtém-se um método de controle eficiente, preciso e amplamente utilizado em sistemas de automação e acionamentos elétricos.

Como resultado da modulação da tensão por PWM, obtém-se no eixo do motor uma velocidade angular ajustável, possibilitando a implementação de estratégias de controle em malha aberta ou fechada. Essa abordagem apresenta simplicidade de implementação, baixo custo e elevada flexibilidade, sendo adotada em sistemas de controle de velocidade de motores de corrente contínua.

O motor de corrente contínua utilizado neste trabalho foi selecionado com base no projeto desenvolvido por Filipe Urban Boscolo, 2019, sendo apresentado na Figura 19. Trata-se de um motor empregado em aplicações didáticas e experimentais, cujas características elétricas e mecânicas são adequadas para estudos de controle de velocidade.

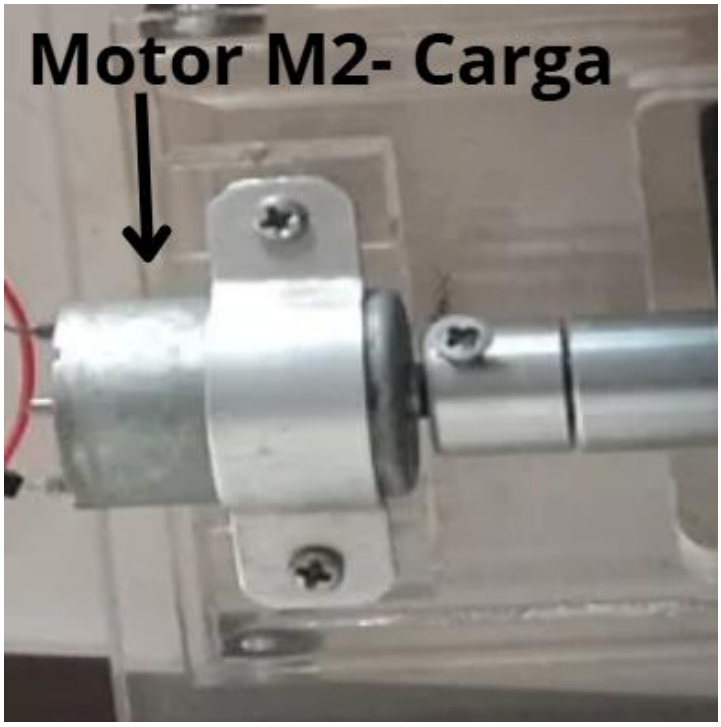
Figura 19: Motor CC usado para o controle

Fonte: (Filipeflop, 2018)

O eixo desse motor foi acoplado mecanicamente a um segundo motor, denominado M2, proveniente de uma impressora Epson, modelo TX-135, com tensão nominal de 12 V. Esse segundo motor é utilizado neste trabalho como carga mecânica. Por apresentar características típicas de motores de pequeno porte e permitir operação em uma ampla faixa de tensão, 1 a 12 V, o motor M2 possibilita a variação controlada do Torque resistente aplicado ao eixo principal, tornando-o adequado para a simulação de diferentes condições de carga durante os ensaios experimentais.

Na Figura 20 ilustra-se o motor M2 empregado para a simulação de carga mecânica.

Figura 20: Motor CC utilizado para simular a carga

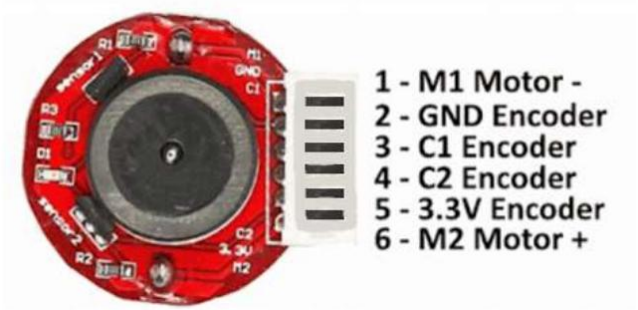


Fonte: Autoria própria.

Além do conjunto motor–carga, o sistema experimental conta com um sensor de velocidade do tipo *encoder*, responsável pela medição da velocidade angular do motor principal, variável que deve ser monitorada e controlada ao longo do funcionamento do sistema. Esse sensor encontra-se integrado ao motor M1 e é conectado por meio de uma placa eletrônica que disponibiliza um conector com seis pinos (M1 Motor; GND;C1;C2;3,3V;M2 Motor), responsáveis tanto pela alimentação do motor quanto pela alimentação do *encoder*.

Nesse conector, também estão disponíveis os sinais de saída do *encoder*, identificados como C1 e C2, os quais fornecem informações analógicas e digitais relativas à rotação do eixo. Na Figura 17 apresenta-se o detalhamento dos seis pinos do conector e suas respectivas funções, facilitando a compreensão da interface elétrica entre o sensor e o sistema de controle.

Figura 21: Placa do Encoder



Fonte: (Filipeflop, 2018)

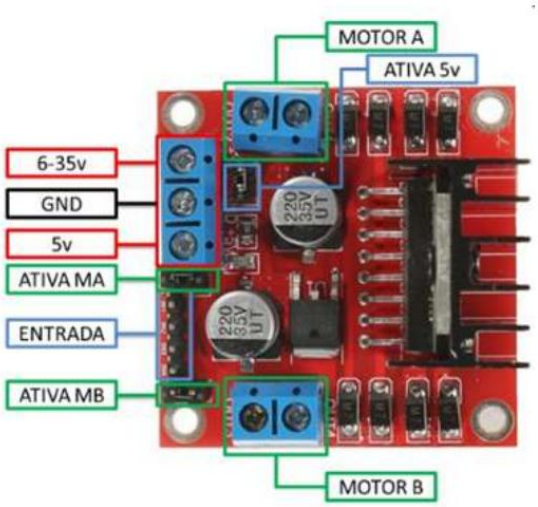
A principal vantagem da utilização desse dispositivo de medição está relacionada à simplicidade da interface com o microcontrolador, uma vez que é possível realizar a leitura das informações do *encoder* utilizando apenas um pino digital do Arduino Uno. No presente trabalho, foi utilizado o pino C1, configurado como saída digital do *encoder*.

6.3 Ponte H L298n e Fonte CC

Como a corrente exigida pelos motores e demais componentes do sistema ultrapassa a capacidade máxima de fornecimento de corrente das saídas do Arduino Uno, o acionamento direto poderia resultar em danos ao microcontrolador ou comprometer sua integridade. Para evitar esse problema e permitir o controle seguro do motor, foi empregado um módulo de ponte H L298N, responsável por realizar a interface de potência entre o Arduino e os motores.

O módulo L298N mostrado na Figura 22, consiste em um circuito integrado montado sobre uma placa de circuito impresso, projetado para o controle da velocidade e do sentido de rotação de motores cc.

Figura 22: Ponte H L298N



Fonte: (Filipeflop, 2018)

Esse módulo possibilita o acionamento simultâneo de até dois motores independentes, a partir de sinais de controle de baixa potência fornecidos por um dispositivo controlador, neste caso, o Arduino Uno.

Os principais terminais do módulo L298N podem ser descritos da seguinte forma:

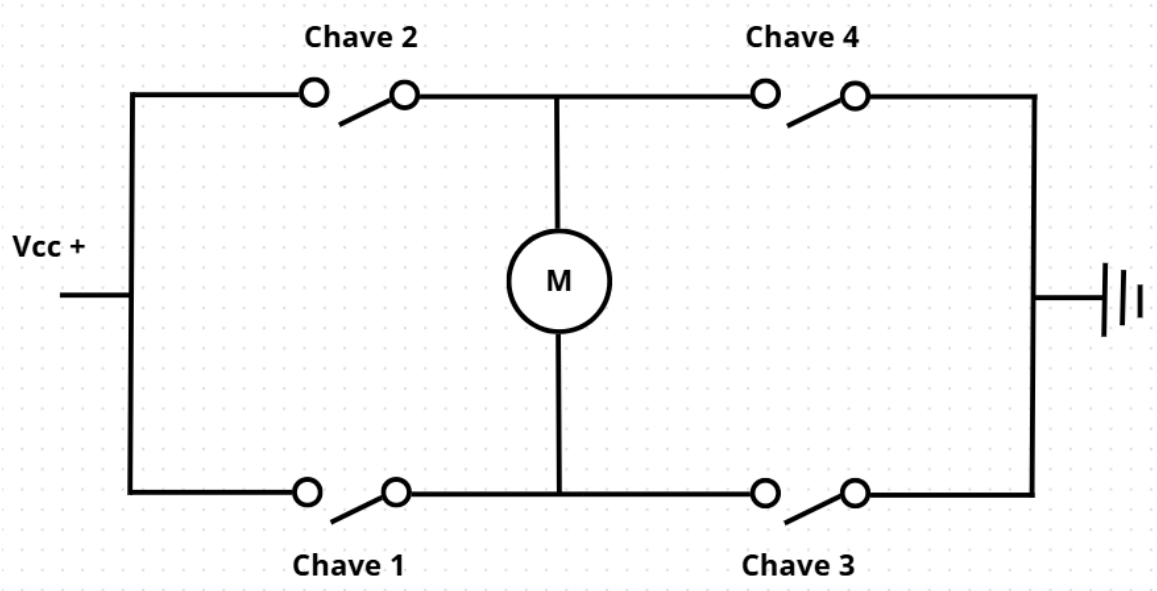
- Alimentação de potência (6–35 V): responsável por fornecer energia aos motores, devendo ser ajustada de acordo com a tensão nominal do motor utilizado, que neste trabalho é de 6 V;
- GND: terminal de referência (terra) do módulo, o qual deve ser comum à terra do Arduino para garantir o correto funcionamento do sistema;
- Alimentação lógica (+5 V): responsável pela alimentação dos circuitos internos de controle do módulo, sendo fornecida pelo Arduino ou por outra fonte regulada de 5 V quando a tensão de alimentação dos motores é inferior a 7 V;
- Jumper de habilitação do regulador: utilizado para ativar ou desativar o regulador interno de tensão do módulo, devendo permanecer desabilitado quando a alimentação dos motores for inferior a 7 V;
- Pinos de habilitação (ENA e ENB): responsáveis por ativar cada uma das pontes H, permitindo também o controle da velocidade dos motores por meio de sinais PWM;
- Entradas lógicas (IN1, IN2, IN3 e IN4): pinos digitais utilizados para definir o sentido de rotação dos motores.

Para a alimentação dos motores, é fundamental que a tensão aplicada não exceda a tensão nominal especificada pelo fabricante, de modo a evitar sobreaquecimento ou danos permanentes. Assim, a alimentação do módulo é ajustada para 6 V. A alimentação lógica do L298N foi realizada a partir da saída de 5 V do Arduino Uno, e o conector do regulador interno foi removido, uma vez que reguladores desse tipo não operam adequadamente com tensões de entrada próximas ao valor de saída.

O controle do sentido de rotação do motor é realizado por meio da configuração adequada das entradas lógicas da ponte H. Por exemplo, para o acionamento do motor A em um sentido, define-se IN1 = nível lógico alto (HIGH) e IN2 = nível lógico baixo (LOW); para a inversão do sentido de rotação, invertem-se esses níveis lógicos. O mesmo princípio se aplica ao motor B, utilizando os pinos IN3 e IN4.

O princípio de funcionamento de uma ponte H baseia-se no controle do sentido da corrente elétrica que atravessa o motor. Ao acionar pares específicos de chaves internas do circuito, a corrente pode fluir em sentidos opostos, resultando na rotação horária ou anti-horária do eixo do motor. Quando as chaves correspondentes aos ramos opostos da ponte são acionadas simultaneamente, a corrente percorre o motor em uma direção definida; ao acionar o par complementar, a direção da corrente é invertida, alterando o sentido de rotação do motor. Esse conceito é ilustrado na Figura 23.

Figura 23: Circuito interno da ponte H

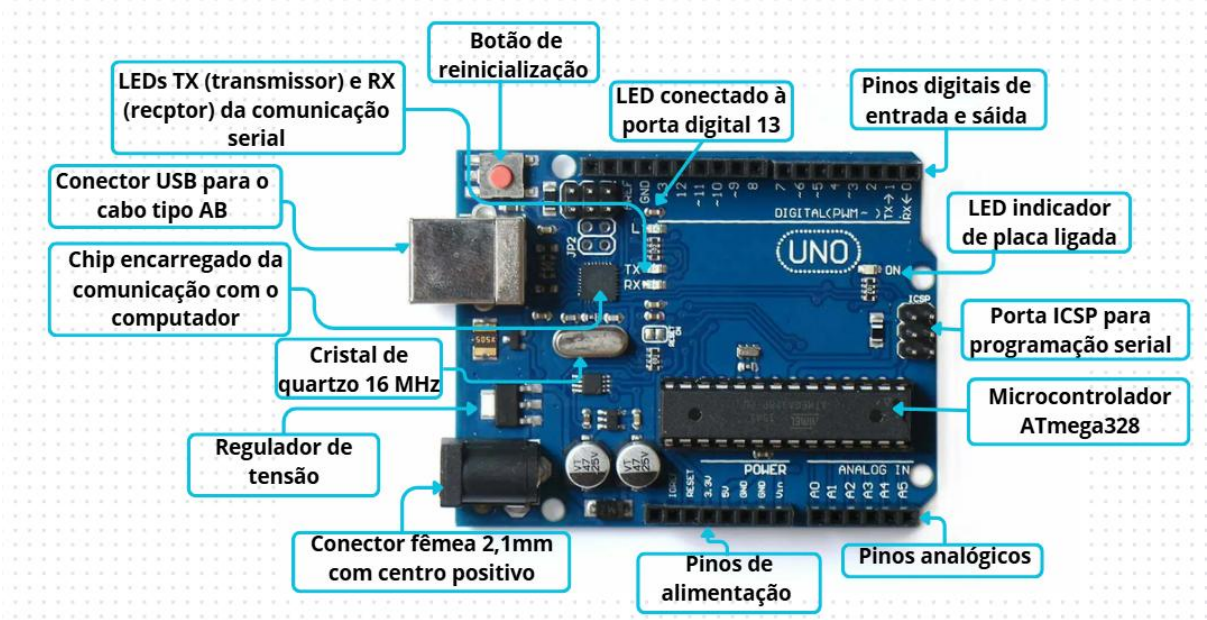


Fonte: Autoria própria.

6.4 Arduino Uno

O Arduino constitui uma plataforma de prototipagem eletrônica de código aberto, utilizada no desenvolvimento de sistemas embarcados, caracterizando-se pela integração de *Hardware* e *Software* de fácil utilização. O elemento central da placa Arduino Uno é um microcontrolador ATMega328P, fabricado pela Microchip (Figura 20), responsável pela execução das instruções do programa, pelo processamento de dados e pelo gerenciamento das entradas e saídas da placa.

Figura 24: Estrutura do Arduino Uno



Fonte: Autoria própria

Além do microcontrolador principal, a placa Arduino Uno incorpora um segundo microcontrolador, o ATMega16U2, cuja função é realizar a interface de comunicação USB entre a placa e o computador. Esse componente permite a transferência do código compilado no ambiente de desenvolvimento Arduino IDE para o microcontrolador principal, bem como a comunicação serial durante a operação do sistema.

A placa Arduino Uno dispõe ainda de um conjunto de pinos de entrada e saída, que possibilitam a conexão com diversos dispositivos externos, tais como sensores, atuadores, módulos eletrônicos e *shields* de expansão. No modelo Arduino Uno R3, estão disponíveis 20 pinos de entrada e saída, sendo 14 pinos digitais e 6 pinos analógicos. Os pinos analógicos, além de sua função principal de leitura de sinais analógicos, também podem ser configurados para operar como entradas ou saídas digitais.

Adicionalmente, seis pinos digitais, identificados pelo símbolo “~”, permitem a geração de sinais do tipo PWM. Esses sinais possibilitam a simulação de níveis analógicos de tensão por meio da modulação da largura de pulso, recurso empregado no controle de velocidade de motores de corrente contínua, no acionamento de cargas e na implementação de estratégias de controle em sistemas embarcados.

6.5 Montagem Do Protótipo Na Bancada

Para a montagem do sistema experimental, foi utilizada uma fonte de alimentação em corrente contínua de 12 V, alimentada pela rede elétrica de 127 V em corrente alternada, cuja função é realizar a conversão da tensão da rede para um nível adequado aos dispositivos do sistema. A partir dessa fonte, o terminal de referência (terra) foi conectado tanto ao Arduino Uno quanto ao módulo da ponte H, sendo imprescindível que ambos compartilhem o mesmo potencial de terra, a fim de garantir o correto funcionamento do circuito e evitar falhas de comunicação ou instabilidades no controle. A tensão de 12 V em corrente contínua foi destinada exclusivamente à alimentação da ponte H, responsável pelo acionamento dos motores.

No módulo da ponte H L298N, encontra-se um conector responsável pela habilitação da alimentação lógica de 5 V, o qual foi removido devido ao fato de a tensão de alimentação do módulo ser inferior a 7 V. Além disso, o jumper do pino ENA também foi retirado, uma vez que sua presença implica na ativação permanente do motor em regime de velocidade máxima, impossibilitando o controle da rotação por meio de sinais PWM.

A ponte H é responsável pelo acionamento dos dois motores utilizados no experimento, conforme esquemático mostrado na Figura 23. A interface entre o Arduino Uno e o módulo L298N foi realizada por meio das seguintes conexões: o pino digital 6 do Arduino foi conectado ao terminal ENA, permitindo o controle da velocidade do motor principal por modulação PWM; os pinos digitais 6 e 7 foram conectados aos terminais IN1 e IN2, responsáveis pelo controle do sentido de rotação do motor principal; os pinos digitais 8 e 9 foram associados aos terminais IN3 e IN4, utilizados para o controle do motor empregado como carga mecânica; adicionalmente, o terminal ENB foi conectado a um pino digital do Arduino para habilitação do segundo motor.

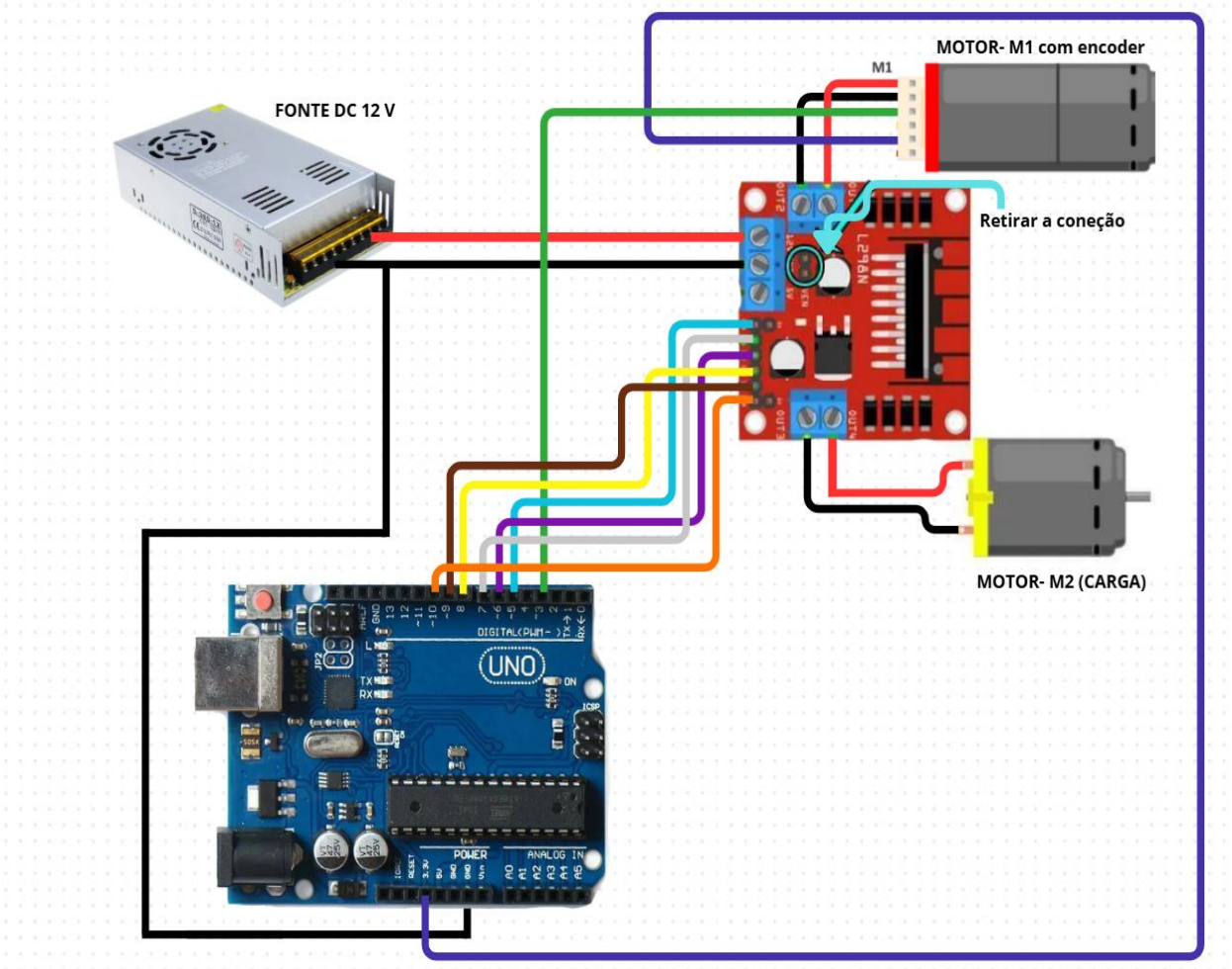
Nessa configuração, os terminais IN1 e IN2 controlam o sentido de rotação do motor principal, sendo associados aos polos positivo e negativo conforme a lógica definida no programa. De forma análoga, o motor utilizado para simular a carga mecânica foi conectado

aos terminais IN3 e IN4, com a polaridade ajustada de modo a permitir a variação controlada do Torque resistente aplicado ao eixo do motor principal.

No que se refere à medição da velocidade angular, o *encoder* foi conectado ao pino digital 3 do Arduino Uno, juntamente com seus respectivos sinais de alimentação e terra. A utilização desse pino permite a leitura adequada dos pulsos gerados pelo *encoder*, possibilitando a aquisição da velocidade do motor de forma confiável para a implementação do controle em malha fechada.

Cabe ressaltar que o motor de corrente contínua utilizado no experimento possui uma carga mecânica permanentemente acoplada ao seu eixo, representada pelo segundo motor empregado para simulação de Torque resistente. Em virtude dessa característica e da indisponibilidade de um motor de maior porte para realização de ensaios a vazio, não foi possível obter experimentalmente todos os parâmetros do motor em condições ideais.

Figura 25: Conexão esquemática para o controle do motor cc



Fonte: Autoria própria.

7. RESULTADOS E DISCUSSÕES

Nesta seção são apresentados e analisados os resultados obtidos a partir dos ensaios de controle de velocidade angular de um motor de corrente contínua utilizando os controladores PI e *Fuzzy*. Inicialmente, são descritos os procedimentos adotados para a calibração do sensor de velocidade e para a identificação dos parâmetros do motor. Em seguida, são apresentados os resultados obtidos em simulação e em bancada, bem como a análise comparativa do desempenho dos controladores.

7.1 Considerações Iniciais

Antes da aplicação dos controladores propriamente ditos, realizou-se um procedimento de calibração do *encoder* acoplado ao motor, com o objetivo de garantir a correta conversão dos pulsos gerados em valores de velocidade angular. Para esse fim, foi realizado um ensaio preliminar utilizando apenas o motor e o microcontrolador Arduino, no qual contou-se o número de pulsos fornecidos pelo *encoder* durante uma rotação completa do eixo do motor.

A partir desse procedimento, foi determinado o número de pulsos por volta (*Pulses Per Revolution* – PPR) do *encoder*. A velocidade angular do motor foi então calculada em rotações por minuto (rpm) por meio da Equação (42):

$$\text{rpm} = \frac{60 \cdot p}{\text{PPR} \cdot \Delta t}, \tag{42}$$

em que p representa o número de pulsos contados em um intervalo de tempo Δt, e PPR corresponde ao número de pulsos por volta do *encoder*.

Para a determinação do valor de PPR, foi desenvolvido um código na plataforma Arduino, no qual os pulsos do *encoder* foram contabilizados durante uma rotação completa do motor. Com base nos resultados obtidos experimentalmente, verificou-se que o *encoder* utilizado apresenta PPR igual a 341 pulsos por volta.

Em função da indisponibilidade de um motor de maior porte e da presença de uma carga mecânica acoplada permanentemente ao eixo do motor utilizado, não foi possível realizar ensaios experimentais completos a vazio para obtenção direta de todos os parâmetros elétricos e mecânicos necessários à modelagem do motor no ambiente Simulink. Dessa forma, inicialmente foram obtidos apenas alguns parâmetros experimentais básicos, tais como a resistência da armadura, medida como $R_a = 2{,}9\ \Omega$, a tensão média aplicada ao motor de $V_{motor} = 9{,}15$ V para um valor de PWM igual a 80, a corrente em vazio $I_0 = 0{,}28$ A e a velocidade de rotação aproximada de 219,84 rpm.

A partir desses valores, estimou-se a velocidade angular do motor, bem como as constantes eletromecânicas associadas, assumindo-se a igualdade entre as constantes de Torque e de força contraeletromotriz ($k_e = k_t$). Contudo, visando obter um modelo mais preciso e representativo do comportamento dinâmico do sistema, optou-se pela utilização do MATLAB para a identificação do modelo do motor. O código desenvolvido para esse procedimento é apresentado no Apêndice A.

Para isso, o motor foi acionado em diferentes condições de operação, sendo registrados, por meio do Arduino, dados experimentais de tempo, velocidade (rpm) e valor de PWM aplicado. Esses dados foram organizados em uma planilha e posteriormente importados para o MATLAB, onde foi realizada a identificação do sistema. O procedimento permitiu a obtenção de modelos matemáticos de primeira e de segunda ordem, bem como a análise do erro associado a cada modelo e a geração de gráficos comparativos entre as respostas experimentais e os modelos identificados.

Com base nesse procedimento, obteve-se o modelo dinâmico de primeira ordem que representa o comportamento do motor de corrente contínua. A função de transferência identificada é expressa pela Equação (43):

$$G_1(s) = \frac{2{,}917}{0{,}7s+1}, \tag{43}$$

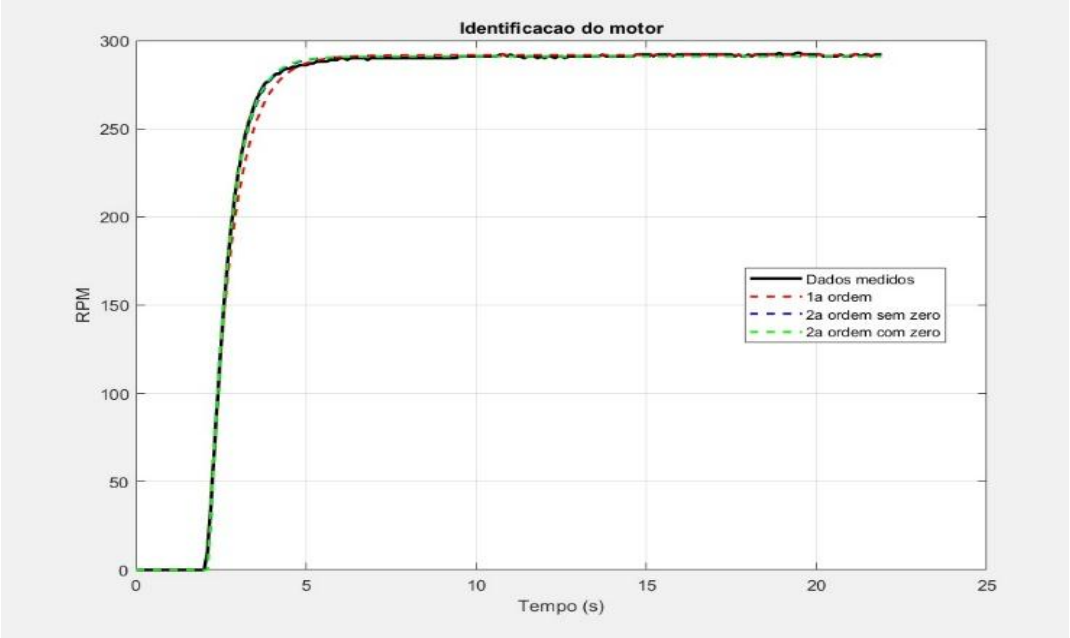
em que o numerador representa o ganho estático do sistema e o denominador caracteriza a constante de tempo associada à dinâmica do motor.

De forma análoga, foi identificado um modelo dinâmico de segunda ordem sem a presença de zeros (sem nenhum termo s no numerador), cuja função de transferência é apresentada na Equação (44):

$$G_2(s) = \frac{127{,}4}{s^2+27{,}42s+43{,}76} \quad . \tag{44}$$

Na Figura 27 apresenta-se a comparação entre os dados experimentais obtidos para a velocidade do motor e as respostas dos modelos identificados de primeira ordem, segunda ordem sem zero e segunda ordem com zero.

Figura 26:Gráfico comparativo entre as equações do motor CC



Fonte: Autoria própria.

O modelo de primeira ordem apresenta maior discrepância em relação aos dados medidos, principalmente na região inicial de aceleração, enquanto os modelos de segunda ordem fornecem um ajuste mais próximo da resposta experimental. Entre estes, o modelo de segunda ordem sem zero apresentou melhor compromisso entre precisão e simplicidade, reproduzindo adequadamente a dinâmica observada sem introduzir características adicionais que não foram verificadas experimentalmente. Dessa forma, o modelo de segunda ordem sem zero foi selecionado para representar o motor neste trabalho.

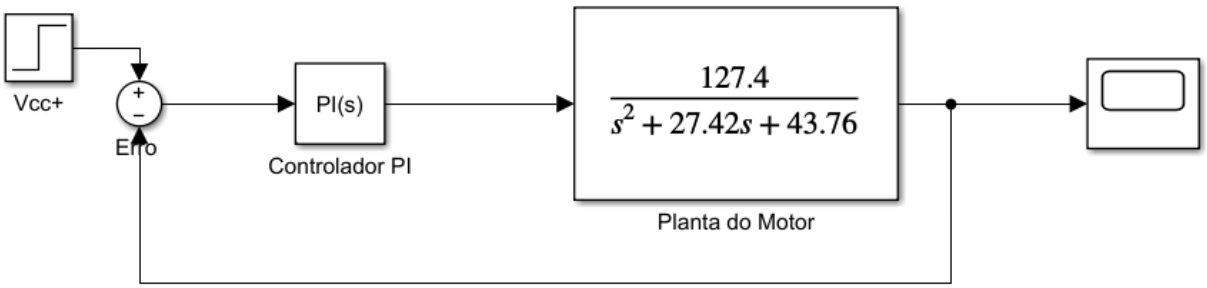
7.2 Simulação Do Modelo Encontrado

Após a obtenção do modelo dinâmico do sistema, foram feitas às simulações utilizando os controladores PI e *Fuzzy*.

7.2.1 Simulação do controlador pi

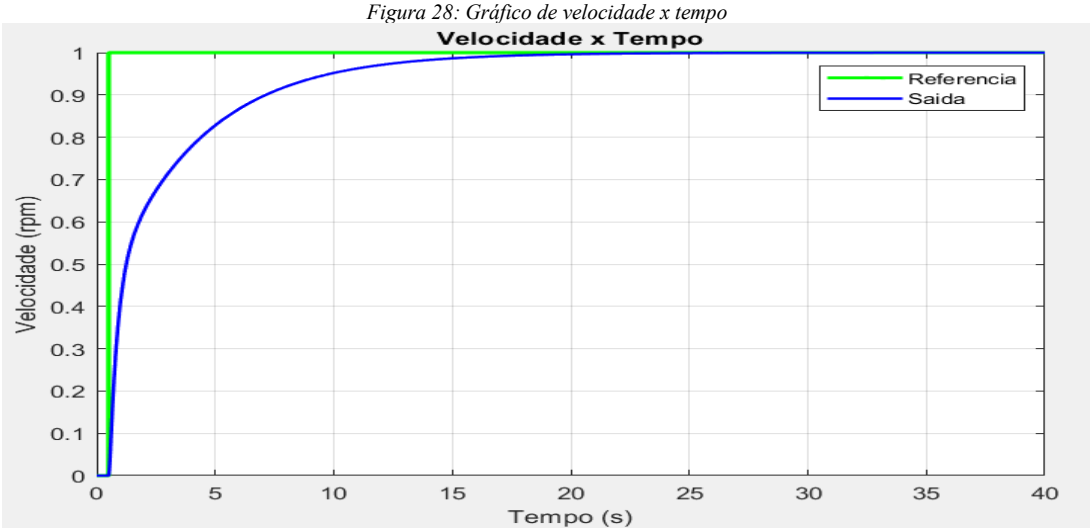
Utilizando o Matlab/Simulink e o modelo matemático do motor estudado, dado pela Equação (44), foi criado um bloco de controle PI, conforme ilustrado na Figura 23. Esse bloco possui scopes destinados ao acompanhamento da velocidade do motor e do esforço do controlador. A primeira simulação apresentada foi usando os ganhos KP=0,3 e KI= 0,15 (BOSCOLO,2019), aos quais teve resultados satisfatórios e será utilizado na parte prática.

Figura 27: Diagrama de bloco do motor CC com o controlador PI



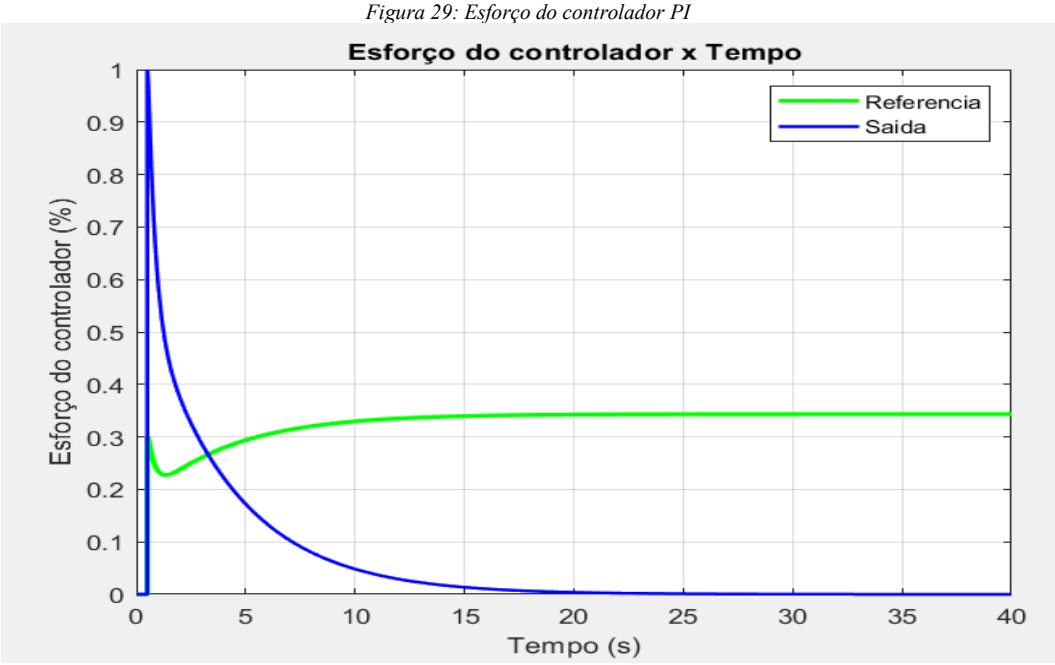
Fonte: Autoria própria

Figura 28: Gráfico de velocidade x tempo



Fonte: Autoria própria

Figura 29: Esforço do controlador PI



Fonte: autoria própria

Utilizando o Matlab, foi desenvolvido o código apresentado no apêndice A, responsável pelo cálculo interno dos parâmetros de desempenho do sistema, conforme definidos pelas

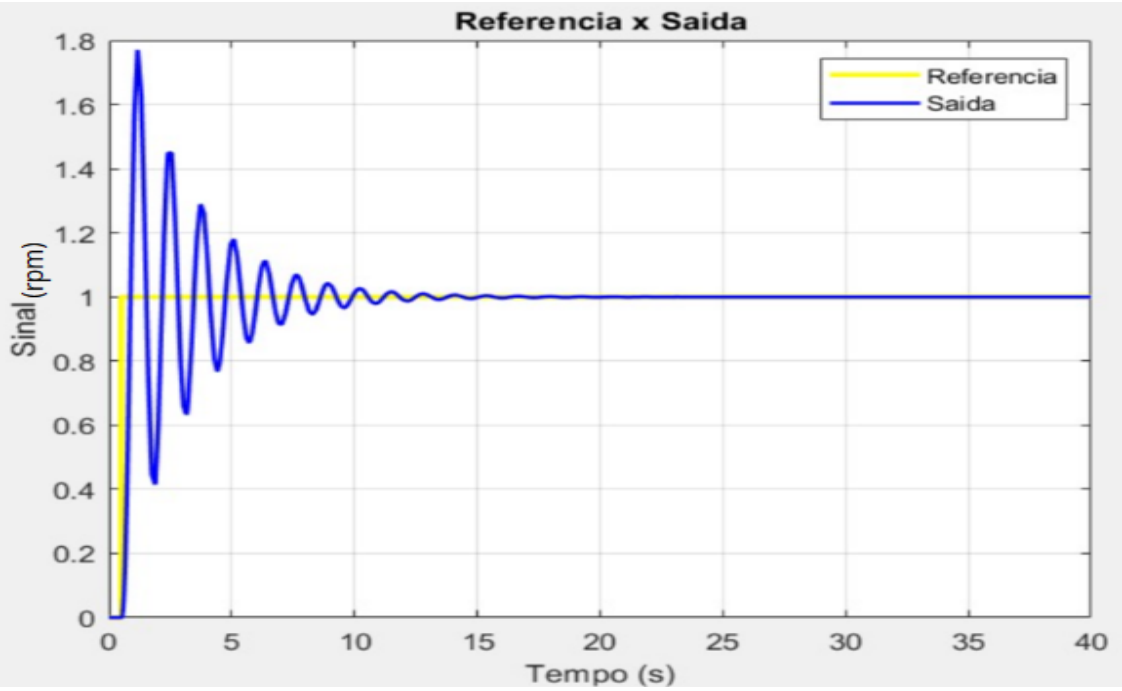
Equações (26) a (29), bem como pela determinação do erro em regime permanente. A partir desse procedimento, foram obtidos os seguintes resultados: sobressinal (*overshoot*) nulo, igual a 0,000%, tempo de subida de 6,515 s, tempo de acomodação de 13,443 s, tempo de pico de 40,000 s, valor de pico unitário igual a 1,000 e erro em regime permanente de 0,000023, indicando um desempenho satisfatório do controlador para os critérios analisados.

Apesar dos resultados obtidos serem satisfatórios, foram realizadas simulações adicionais com o objetivo de analisar o efeito da variação dos ganhos do controlador.

Considerando um ganho proporcional nulo e um ganho integral elevado, observa-se que o sistema apresenta um aumento significativo do sobressinal (*overshoot*), enquanto o erro em regime permanente se mantém bastante reduzido. Esse comportamento é característico da predominância da ação integral, a qual atua de forma eficiente na eliminação do erro estacionário, porém tende a piorar o desempenho transitório do sistema.

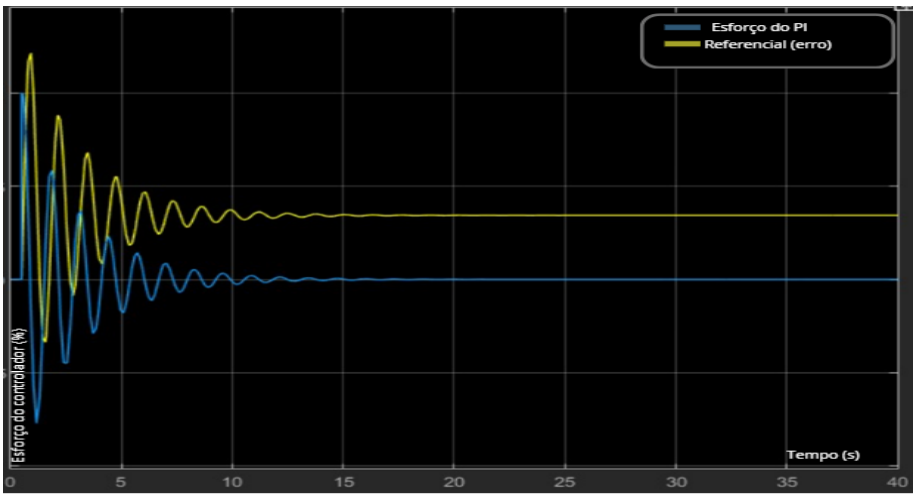
Na Figura 30 apresenta-se a resposta da velocidade do motor sob essa condição de controle, evidenciando um pico acentuado acima do valor de referência. Enquanto na Figura 31 apresenta-se o esforço do controlador PI, no qual se observa uma atuação intensa da ação integral como consequência do elevado valor do ganho K_i.

Figura 30: Gráfico de velocidade do motor CC, com o ganho Ki alto



Fonte: Autoria própria

Figura 31: Gráfico do esforço do controlador PI quando Ki é alto

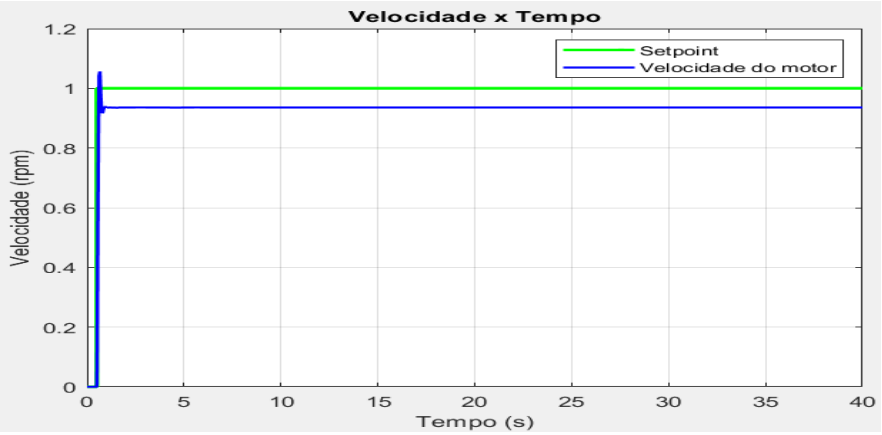


Fonte: Autoria própria

Para esse caso, foram obtidos os seguintes resultados de desempenho: sobressinal igual a 77,094%, tempo de subida de 0,230 s, tempo de acomodação de 10,866 s, tempo de pico de 1,166 s, valor de pico igual a 1,771 e erro em regime permanente de 0,000149. Esses resultados evidenciam que, embora a ação integral seja eficaz na redução do erro estacionário, valores elevados de K_i comprometem significativamente o comportamento transitório do sistema, tornando essa configuração inadequada para aplicações que exigem resposta suave e sem grandes oscilações.

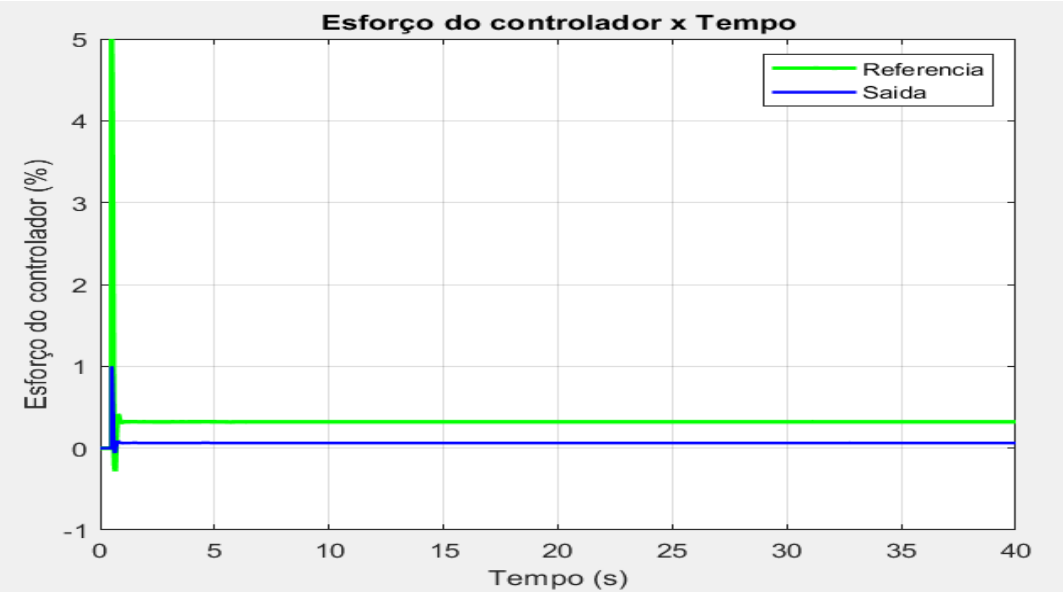
Com o intuito de ampliar a análise comparativa, simulou-se também o cenário inverso, no qual se adota um ganho proporcional elevado e um ganho integral reduzido Na Figura 32 apresenta-se a resposta da velocidade do motor de corrente contínua sob essa condição, enquanto na Figura 33 apresenta-se o esforço do controlador PI correspondente.

Figura 32: Velocidade do motor CC com Kp alto



Fonte: Autoria própria

Figura 33: Gráfico do esforço do controlador PI com Kp alto



Fonte: autoria própria

Para essa configuração de controle, o sistema apresentou um sobressinal de 25,814%, tempo de subida de 0,041 s, tempo de acomodação de 0,773 s, tempo de pico de 0,584 s, valor de pico igual a 1,216 e erro em regime permanente de 0,033246. Observa-se que o aumento do ganho proporcional proporciona uma resposta mais rápida, refletida nos menores tempos de subida, pico e acomodação, entretanto resulta em erro de regime permanente mais elevado, evidenciando a limitação da ação proporcional na eliminação desse erro.

Comparando-se os três cenários simulados, verifica-se que valores elevados do ganho integral tendem a aumentar significativamente o sobressinal e tornar a resposta do sistema mais lenta, apesar de reduzirem de forma eficaz o erro em regime permanente. Por outro lado, um ganho proporcional elevado também ocasiona aumento do sobressinal e do erro estacionário, porém promove uma resposta consideravelmente mais rápida, com tempos de acomodação e de pico muito inferiores quando comparados ao caso de ganho integral elevado, diferença está bastante expressiva. Essa análise evidencia a necessidade de um compromisso adequado entre as ações proporcional e integral para alcançar um desempenho satisfatório tanto em regime transitório quanto em regime permanente.

7.2.2 Simulação do controlador fuzzy

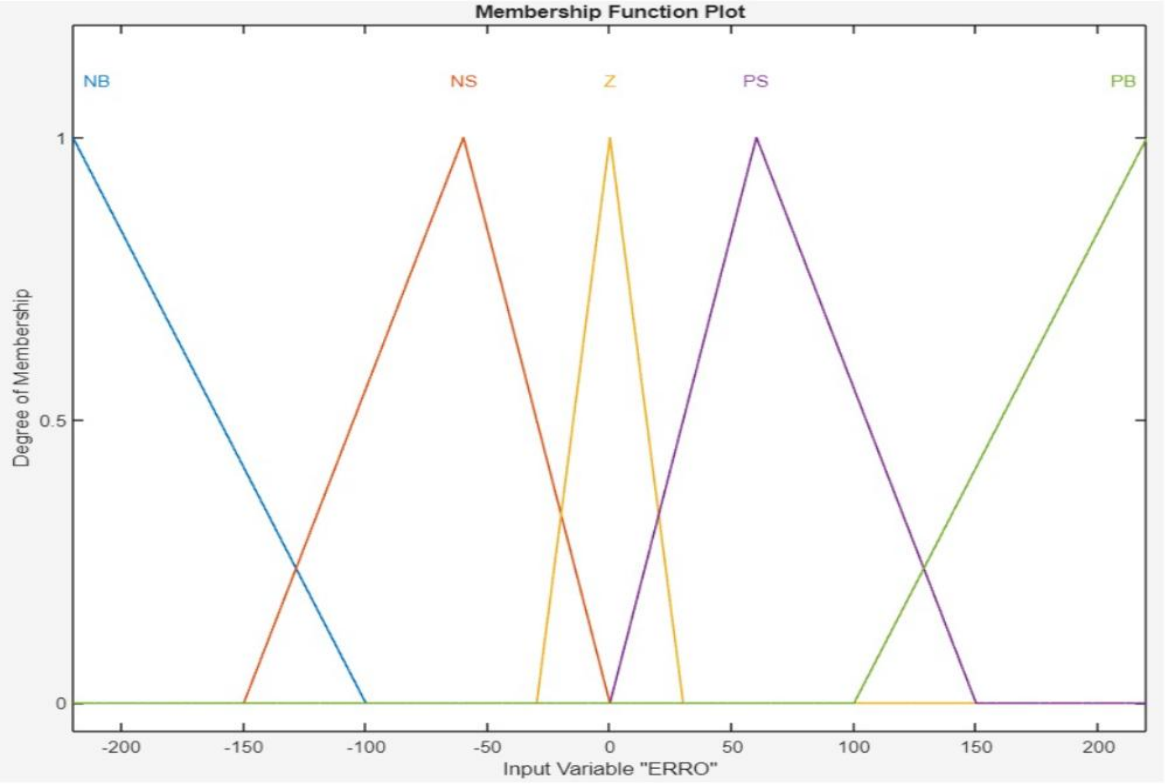
O controlador *Fuzzy* foi desenvolvido no ambiente Matlab por meio da ferramenta própria *Fuzzy Logic Designer*. Para esse controlador, foi proposto um sistema *Fuzzy* com duas entradas e uma saída. As entradas correspondem ao erro entre a velocidade de referência e a

velocidade medida do motor (setpoint – rpm) e à derivada do erro, utilizada para avaliar a dinâmica da variação do erro ao longo do tempo. Como saída do sistema, adotou-se o sinal PWM aplicado ao motor, responsável pela atuação no controle da velocidade angular.

Cada variável do sistema *Fuzzy* foi definida com cinco termos linguísticos. Na Figura 35 são apresentadas as funções de pertinência associadas à variável de entrada erro, composta pelos conjuntos linguísticos NB (*Negative Big*), NS (*Negative Small*), Z (Zero), PS (*Positive Small*) e PB (Positive Big).

Para a construção dessas funções de pertinência, foram utilizadas funções triangulares, escolhidas devido à sua simplicidade de implementação e à boa capacidade de representação do comportamento do sistema. Os intervalos de atuação de cada termo linguístico foram definidos de maneira a garantir uma sobreposição adequada entre os conjuntos, possibilitando transições suaves na ação de controle.

Figura 34: Funções de pertinência da variável de entrada erro do controlador Fuzzy



Fonte: Autoria própria.

Os intervalos associados a cada termo linguístico da variável de entrada *erro* encontram-se detalhados na Tabela 1, ao qual foi definido de acordo com o comportamento do motor cc no ambiente de simulação.

Tabela 1: Parâmetros das funções de pertinência da variável de entrada erro

Termo linguístico	Tipo de função	Intervalo
NB	Triangular	[−220 −220 −100]
NS	Triangular	[−150 −60 0]
Z	Triangular	[−30 0 30]
PS	Triangular	[0 60 150]
PB	Triangular	[100 220 220]

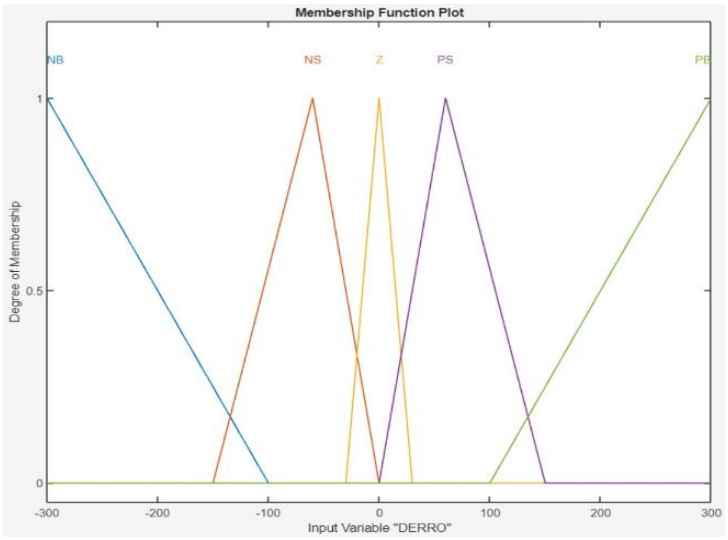
Fonte: Autoria própria

A segunda variável de entrada do controlador *Fuzzy* corresponde à derivada do erro (*derro*), a qual representa a taxa de variação do erro ao longo do tempo. Essa variável fornece ao sistema informações dinâmicas sobre o comportamento do erro, permitindo identificar se ele está aumentando, diminuindo ou se mantendo próximo de zero.

Na Figura 36 são apresentas as funções de pertinência associadas à variável do erro, definidas por cinco termos linguísticos: NB (*Negative Big*), NS (*Negative Small*), Z (Zero), PS (*Positive Small*) e PB (Positive Big).

Para a composição dessas funções, foram utilizadas funções triangulares, distribuídas ao longo de um intervalo de atuação compreendido entre –300 e 300. Os parâmetros numéricos associados a cada termo linguístico encontram-se detalhados na Tabela 2, sendo definidos de forma a garantir uma sobreposição adequada entre os conjuntos e proporcionar transições suaves na ação de controle.

Figura 35: Funções de pertinência da variável de entrada derro do controlador Fuzzy



Fonte: Autoria própria.

Tabela 2: Parâmetros das funções de pertinência da variável de entrada derro

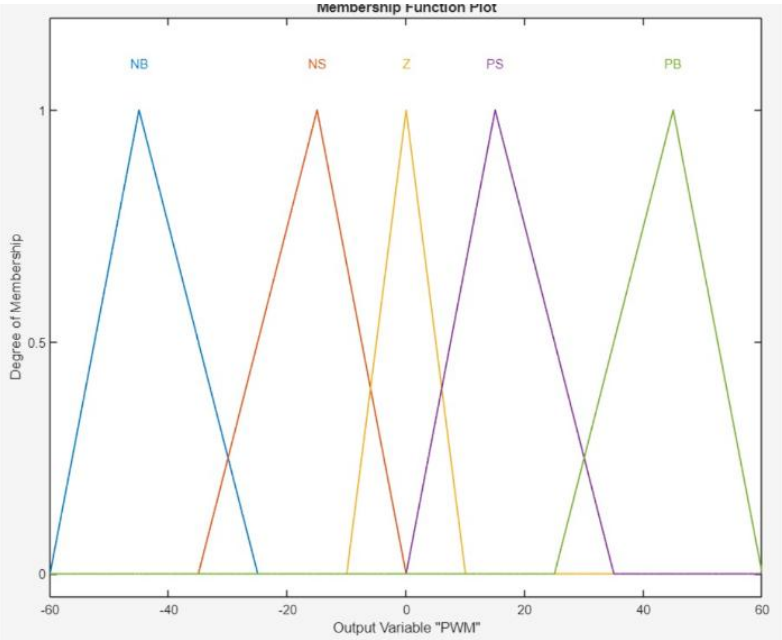
Termo linguístico	Tipo de função	Intervalo
NB	Triangular	[−300 −300 −100]
NS	Triangular	[−150 −60 0]
Z	Triangular	[−30 0 30]
PS	Triangular	[0 60 150]
PB	Triangular	[100 300 300]

Fonte: Autoria própria.

A variável de saída do controlador Fuzzy corresponde ao sinal de controle PWM, responsável pela atuação direta na velocidade angular do motor de corrente contínua. Essa variável define o ajuste aplicado à tensão média de alimentação do motor, possibilitando o aumento, a redução ou a manutenção da velocidade em torno do valor de referência.

As funções de pertinência que representam à variável de saída PWM, são definidas por cinco termos linguísticos do tipo triangular: NB (*Negative Big*), NS (*Negative Small*), Z (Zero), PS (*Positive Small*) e PB (Positive Big), conforme mostrado na Figura 37, distribuídas ao longo do intervalo de atuação compreendido entre –60 e 60. Os parâmetros numéricos correspondentes a cada termo linguístico são apresentados na Tabela 3.

Figura 36:Funções de pertinência da variável de saída PWM do controlador Fuzzy



Fonte: Autoria própria.

Tabela 3: Parâmetros das funções de pertinência da variável de saída PWM

Termo linguístico	Tipo de função	Intervalo
NB	Triangular	[−60 −45 −25]
NS	Triangular	[−35 −15 0]
Z	Triangular	[−10 0 10]
PS	Triangular	[0 15 35]
PB	Triangular	[25 45 60]

Fonte: Autoria própria.

A base de regras do controlador *Fuzzy* foi elaborada considerando as duas variáveis de entrada, erro e derivada do erro (derro), e a variável de saída PWM. O sistema é composto por um total de 25 regras do tipo SE–ENTÃO, resultantes da combinação de cinco termos linguísticos para cada variável de entrada.

Todas as regras foram definidas com peso unitário, garantindo que nenhuma regra tenha maior influência que as demais. A base de regras, mostrada na Tabela 4, foi construída de forma a proporcionar uma ação de controle progressiva e coerente, em que erros elevados resultam em atuações mais intensas no sinal PWM, enquanto valores próximos de zero promovem ações mais suaves. Essa estrutura assegura um equilíbrio adequado entre resposta dinâmica e estabilidade do sistema controlado.

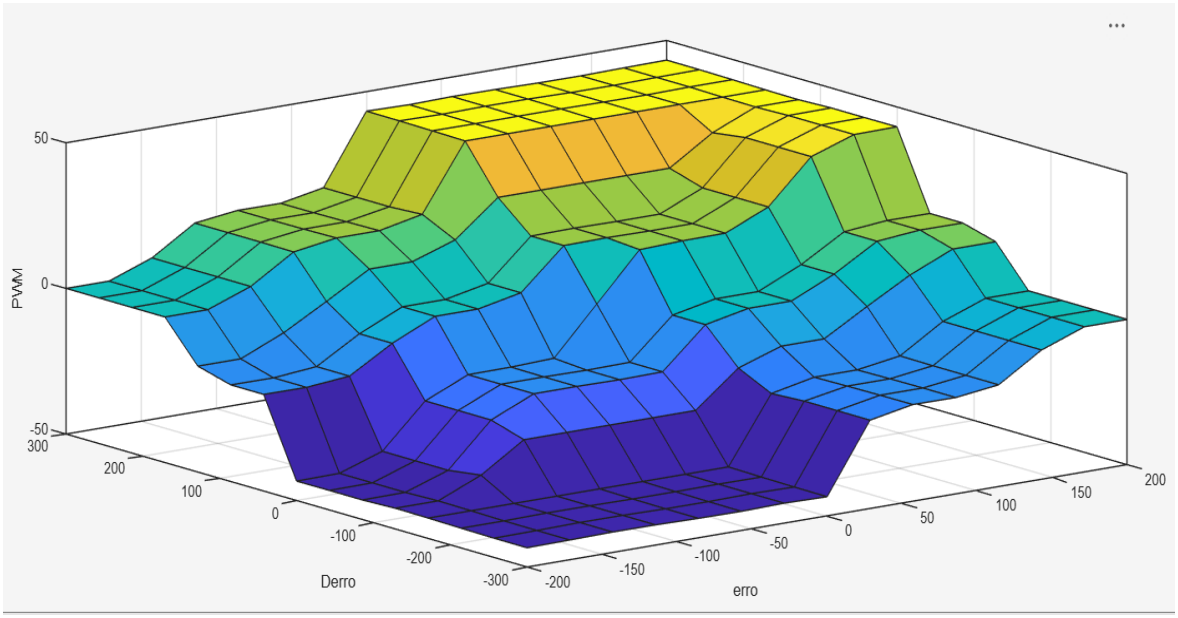
Tabela 4: Base de regras do controlador Fuzzy

Erro \ Derro	NB	NS	Z	PS	PB
NB	NB	NB	NB	NS	Z
NS	NB	NS	NS	Z	PS
Z	NB	NS	Z	PS	PB
PS	NS	Z	PS	PS	PB
PB	Z	PS	PB	PB	PB

Fonte: Autoria própria.

Na Figura 38 apresenta a superfície de inferência do controlador *Fuzzy*, a qual ilustra o comportamento da variável de saída PWM em função das variáveis de entrada erro e derivada do erro (derro). Essa superfície representa graficamente a aplicação conjunta das funções de pertinência, da base de regras *Fuzzy* e do processo de defuzzificação adotado.

Figura 37: Superfície de ingerência do controlador Fuzzy



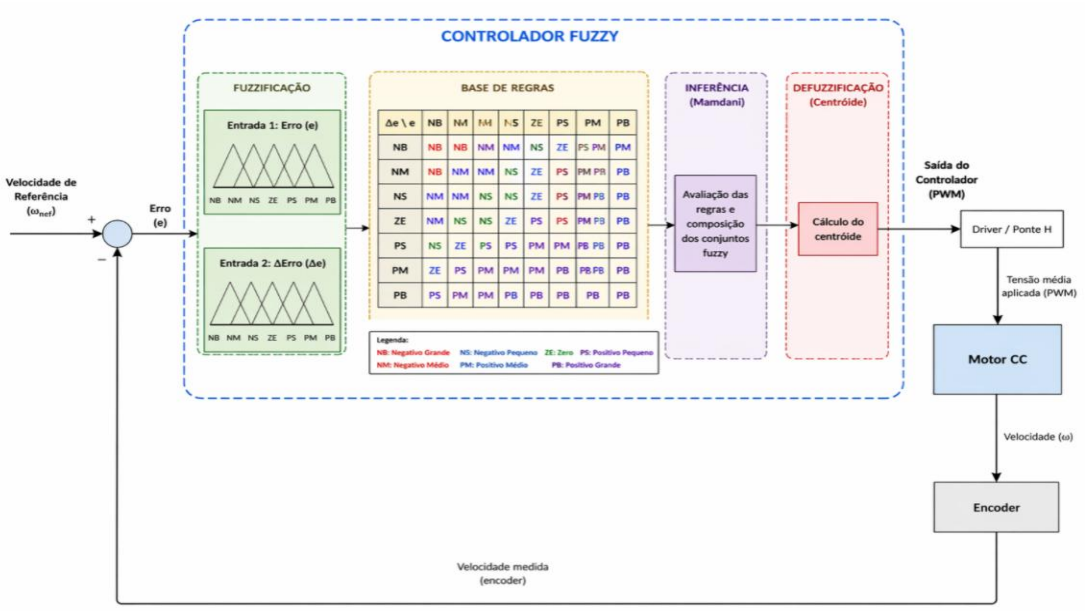
Fonte: Autoria própria.

Observa-se que, para valores elevados e positivos do erro, combinados com derivadas positivas, o controlador fornece uma ação de controle mais intensa, refletida em valores elevados de PWM. De forma análoga, para erros negativos significativos, a saída do controlador assume valores reduzidos ou negativos, promovendo a diminuição do sinal aplicado ao motor. Nas regiões próximas ao zero, a superfície apresenta transições suaves, indicando uma atuação moderada do controlador.

A continuidade da superfície evidencia que o controlador *Fuzzy* proporciona uma transição gradual entre as diferentes regiões de operação, evitando mudanças abruptas no sinal de controle. Essa característica contribui para uma resposta mais estável e suave do sistema, sendo uma das principais vantagens do controle *Fuzzy* quando comparado a estratégias de controle convencionais.

Com o sistema *Fuzzy* implementado, constrói-se um circuito esquemático do controlador *Fuzzy*, mostrado na Figura 38.

Figura 38: Esquemático do controlador Fuzzy

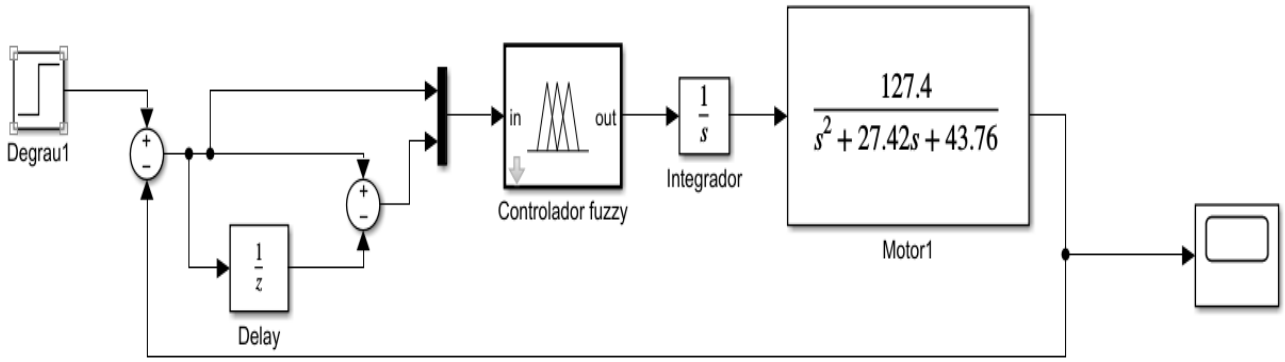


Fonte: Autoria própria.

No ambiente Simulink, a implementação do controlador *Fuzzy* foi realizada de forma direta, bastando substituir o bloco do controlador PI pelo bloco correspondente ao controlador Fuzzy. Ressalta-se que, para fins de simplificação da simulação, o controlador *Fuzzy* foi configurado para gerar o incremento do sinal de controle (ΔPWM). Dessa forma, tornou-se necessária a inserção de um bloco integrador após o controlador, a fim de obter o sinal de controle efetivamente aplicado ao motor.

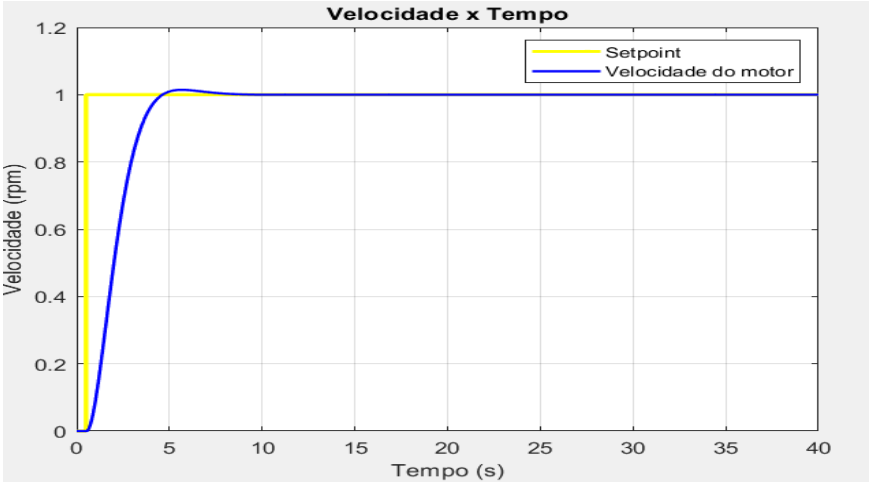
A Figura 39 ilustra o diagrama de blocos do sistema em malha fechada com o controlador *Fuzzy,* evidenciando a substituição do controlador PI, a presença do integrador e o modelo dinâmico do motor utilizado nas simulações.

Figura 39: Diagrama de blocos do controlador Fuzzy



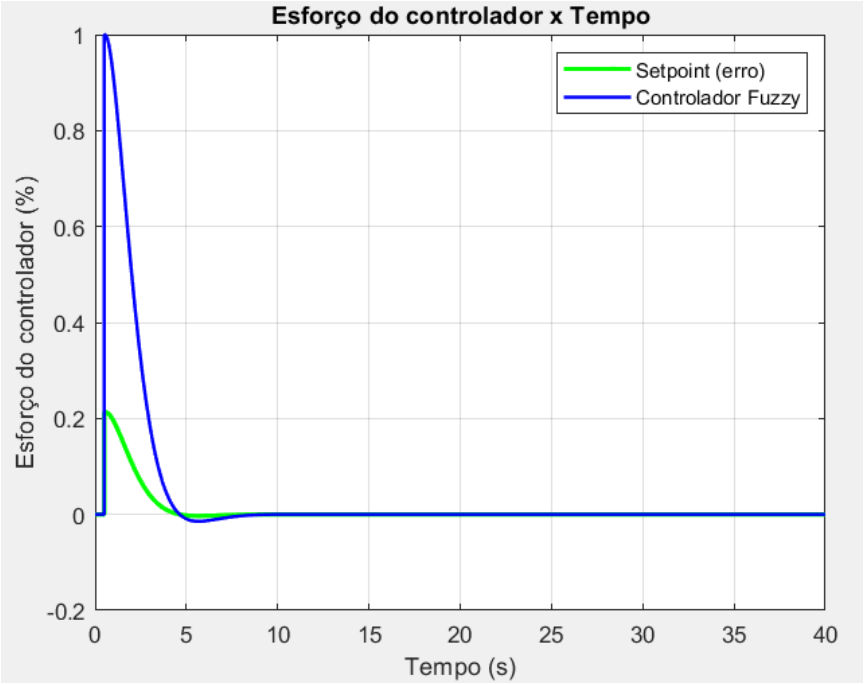
Fonte: Autoria própria.

Figura 40: Gráfico da velocidade do motor com o controlador Fuzzy



Fonte: Autoria própria.

Figura 41: Gráfico do esforço do controlador Fuzzy



Fonte: Autoria própria

O controlador *Fuzzy* apresentou os seguintes resultados: sobressinal de 1,441%, tempo de subida de 2,417 s, tempo de acomodação de 4,217 s, tempo de pico de 5,600 s, valor de pico igual a 1,014 e erro em regime permanente da ordem de 1×10^{-6}. Observa-se que o controlador *Fuzzy* apresenta um desempenho bastante satisfatório, com baixo sobressinal e erro em regime permanente praticamente nulo, indicando elevada precisão no seguimento da referência.

Além disso, os tempos de subida e de acomodação são reduzidos quando comparados a configurações conservadoras do controlador PI, evidenciando uma resposta mais rápida sem

a introdução de oscilações significativas. Esse comportamento pode ser atribuído à capacidade do controlador *Fuzzy* em adaptar sua ação de controle de forma não linear, utilizando simultaneamente informações do erro e da sua derivada, o que resulta em uma atuação mais suave e eficiente ao longo do regime transitório e permanente.

5.3 Programas Do Arduino E Supervisório

A etapa prática do experimento foi implementada utilizando a plataforma Arduino, em conjunto com o *Software* supervisório Elipse E3, por meio de comunicação serial baseada no protocolo Modbus RTU. Nessa arquitetura, o Elipse E3 foi configurado como mestre Modbus, responsável pelo envio de comandos e parâmetros ao sistema, enquanto o Arduino atuou como escravo Modbus, executando localmente os algoritmos de controle e retornando as variáveis medidas ao supervisório.

No Elipse E3, foi desenvolvido uma interface gráfica que serve como ponte de comunicação entre o motor e o Arduino (Figura 42). Essa interface conta com quatro telas principais para a operação e supervisão do sistema. A primeira corresponde à tela inicial, na qual são apresentados o nome do projeto e seus autores.

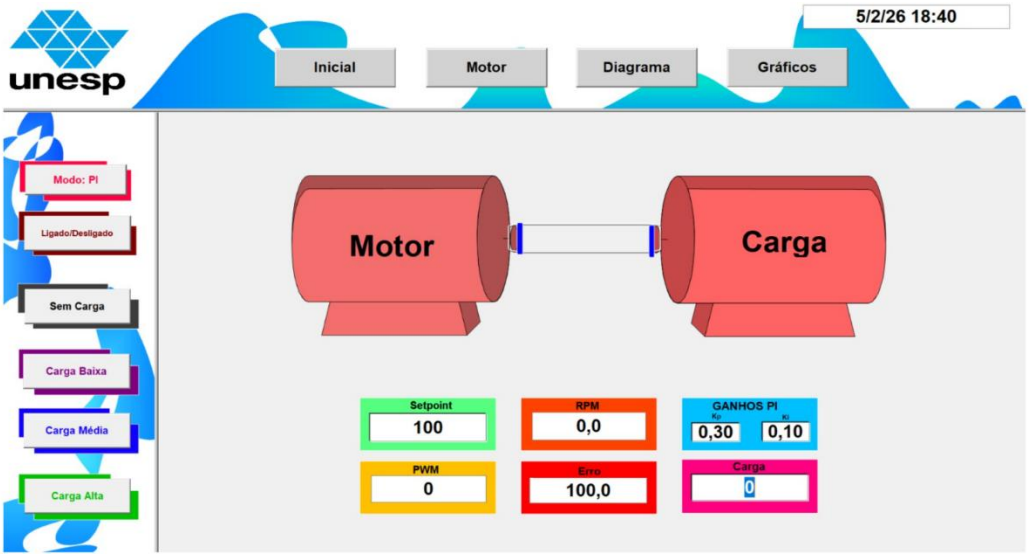
Figura 42: Primeira camada da interface

Fonte: Autoria própria

A segunda tela desenvolvida (Figura 43) é dedicada à supervisão do motor, apresentando uma representação gráfica do sistema físico montado em bancada, bem como a visualização em tempo real das principais variáveis do processo. Nessa interface, assim como nas telas subsequentes, encontra-se um botão de Liga/Desliga, responsável por acionar o motor

principal, no qual são aplicadas as estratégias de controle PI e *Fuzzy*. Além disso, são disponibilizados quatro botões destinados ao controle da carga mecânica, permitindo ligar ou desligar o motor de carga e aplicar níveis de carga previamente definidos como baixa (48%), média (72%) e alta (92%). A interface também contempla um campo de setpoint de carga, possibilitando ao usuário inserir valores específicos diferentes daqueles previamente programados.

Figura 43: Segunda camada da interface

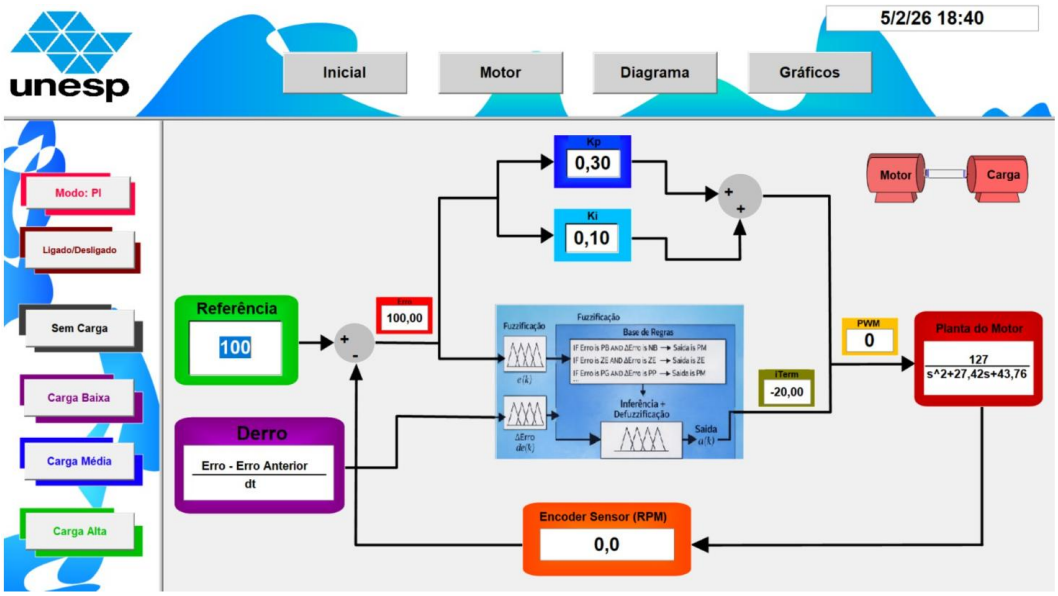


Fonte: Autoria própria

Complementarmente, a tela apresenta displays responsáveis pela indicação da velocidade do motor, do erro de controle e do valor do sinal PWM aplicado, bem como campos editáveis para ajuste do setpoint de velocidade e dos ganhos dos controladores, permitindo a supervisão e a interação do usuário com o sistema durante os ensaios experimentais.

A terceira tela (Figura 44) contempla a interface dos controladores, na qual foi redesenhado o diagrama de blocos do sistema de controle, permitindo o acompanhamento dos sinais de entrada e saída durante a operação.

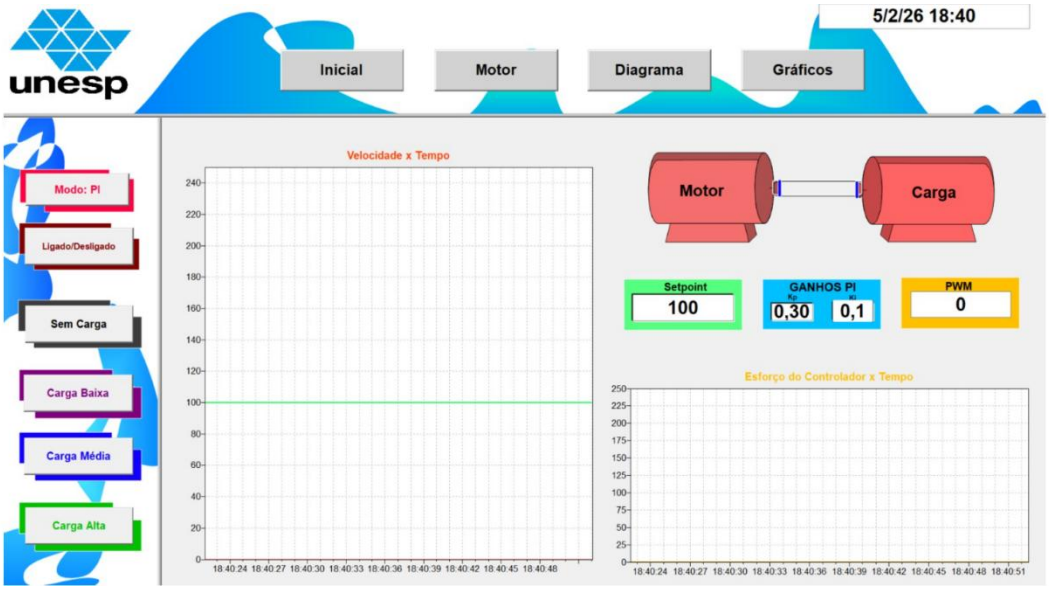
Figura 44: Terceira camada da interface



Fonte: Autoria própria

Por fim, a quarta tela (Figura 45) é destinada à apresentação gráfica dos resultados experimentais, possibilitando a análise do comportamento do sistema ao longo do tempo.

Figura 45: Quarta camada da interface=



Fonte: Autoria própria

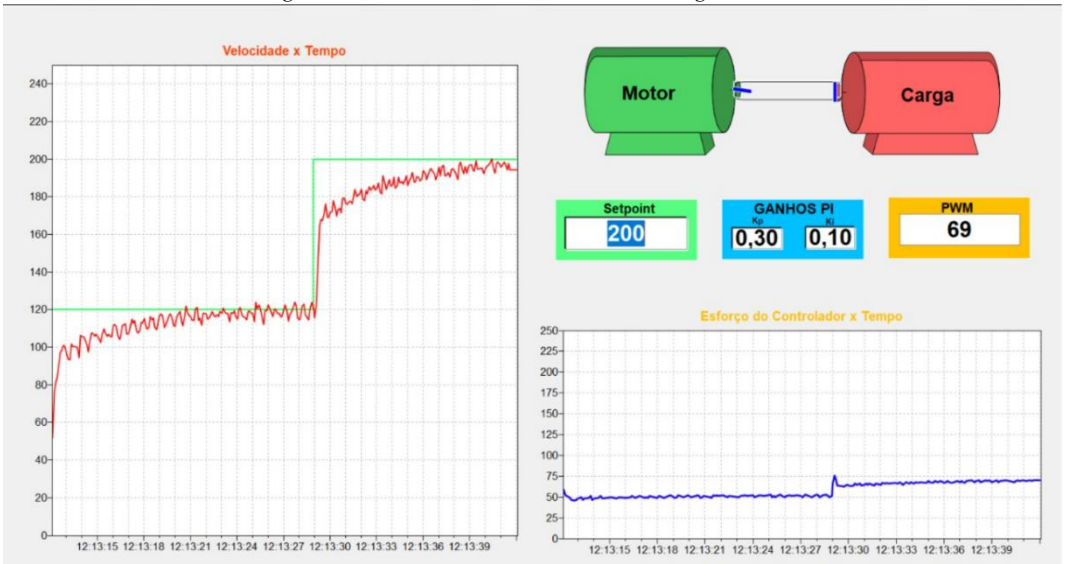
5.4 Aplicando Cargas E Ligando O Controlador Pi

Para o início das análises, foi considerado que para cada tipo de carga, o controle deve responder de forma a estabilizar o sistema e fazer a velocidade se manter constante, portanto, antes do acréscimo da carga, inicia-se com o motor a vazio e implementando alguns valores de ganhos proporcional e integrador, após essa implementação coloca-se as cargas.

5.4.1 Controlador PI

O primeiro controle foi efetuado com o motor trabalhando a vazio. Na Figura 46 é ilustrada a curva de velocidade vs tempo para o controle pi sem carga.

Figura 46: Motor CC com controlador PI sem carga adicional



Fonte: Autoria própria

Analisando a Figura 41, tem- se que, o sistema opera com um setpoint em torno de 120 rpm, sendo posteriormente aplicado um degrau para 200 rpm. Observa-se que a velocidade do motor acompanha a mudança de referência, convergindo para o novo valor desejado após um período transitório.

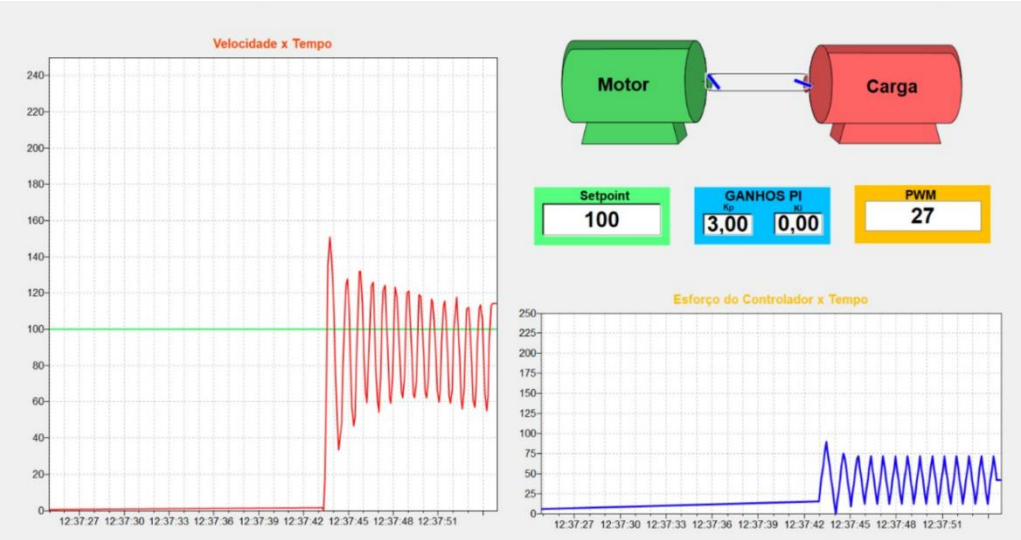
Após o degrau, o sistema apresenta um tempo de subida reduzido, com a velocidade do motor aumentando rapidamente em direção aos 200 rpm pedido. Esse comportamento evidencia a capacidade do controlador PI em responder prontamente a variações na referência. Além disso, verifica-se que o sistema não apresenta *overshoot*, uma vez que a velocidade do motor não ultrapassa o valor de referência estabelecido, indicando uma resposta bem amortecida e uma sintonia adequada dos ganhos do controlador.

Observa-se ainda a presença de pequenas oscilações na resposta da velocidade, atribuídas às características do sistema físico, como ruídos de medição do *encoder*, vibrações mecânicas e variações de carga. Apesar disso, o controlador PI mantém o sistema estável e com erro em regime permanente reduzido. Na Figura em questão também é possível observar o gráfico do esforço do controlador que evidencia um aumento do sinal PWM após aplicar o degrau, seguido de sua estabilização em um novo patamar, indicando uma atuação adequada do controlador para compensar a dinâmica do sistema e manter a velocidade desejada. Esses

resultados demonstram que o controlador PI é capaz de regular a velocidade do motor de corrente contínua de forma satisfatória em condições reais de operação.

Ademais, foi realizado o experimento referente à modificação dos ganhos para analisar o comportamento do controlador com diferentes ganhos, conforme Figuras 47-49.

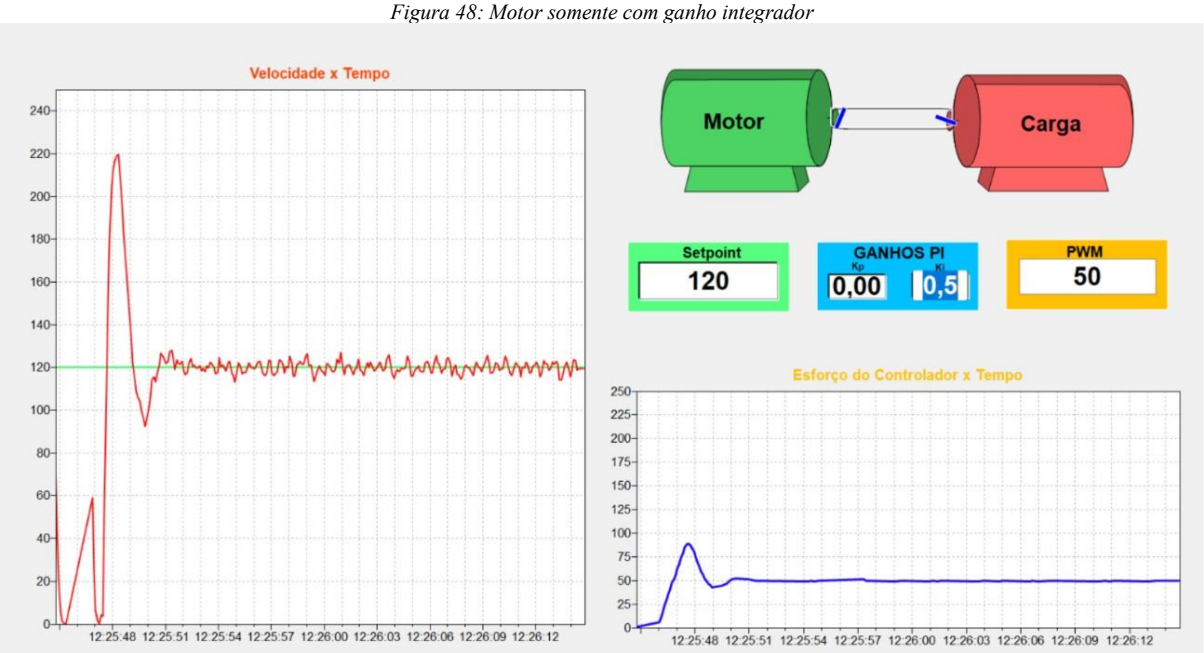
Figura 47: Motor CC com controle PI tendo apenas Kp



Fonte: Autoria própria

A partir da resposta apresentada na Figura 47, observa-se que a utilização de um controlador apenas proporcional, com ganho $K_p = 3$ (alto) e ausência da ação integral, resulta em um sobressinal elevado e em oscilações persistentes na velocidade do motor. Nota-se ainda que, para valores de K_p inferiores a 3, a velocidade não se aproxima adequadamente do valor de referência, evidenciando um erro em regime permanente significativo. Esses resultados indicam que o controle puramente proporcional não é suficiente para garantir precisão e estabilidade adequadas no controle de velocidade do motor, reforçando a necessidade da inclusão da ação integral.

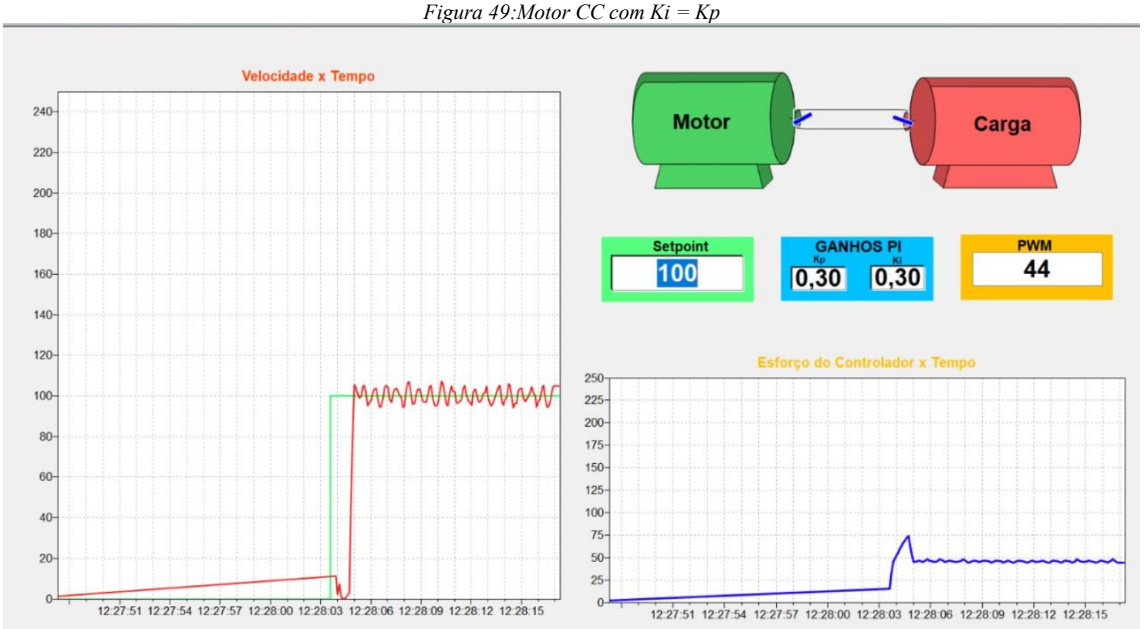
Figura 48: Motor somente com ganho integrador



Fonte: Autoria própria

A partir da resposta apresentada na Figura 48, observa-se que, embora o sistema apresente um *overshoot* elevado devido à predominância da ação integral ($K_i = 0{,}5$ e $K_p = 0$), a velocidade do motor se estabiliza rapidamente em torno do valor de referência. Esse comportamento indica que a ação integral é eficaz na eliminação do erro em regime permanente, garantindo a convergência da velocidade para o setpoint. No entanto, o alto sobressinal inicial compromete o desempenho transitório do sistema, evidenciando que o uso isolado da ação integral não é adequado para aplicações que exigem resposta suave, reforçando a necessidade da combinação das ações proporcional e integral.

Figura 49:Motor CC com Ki = Kp



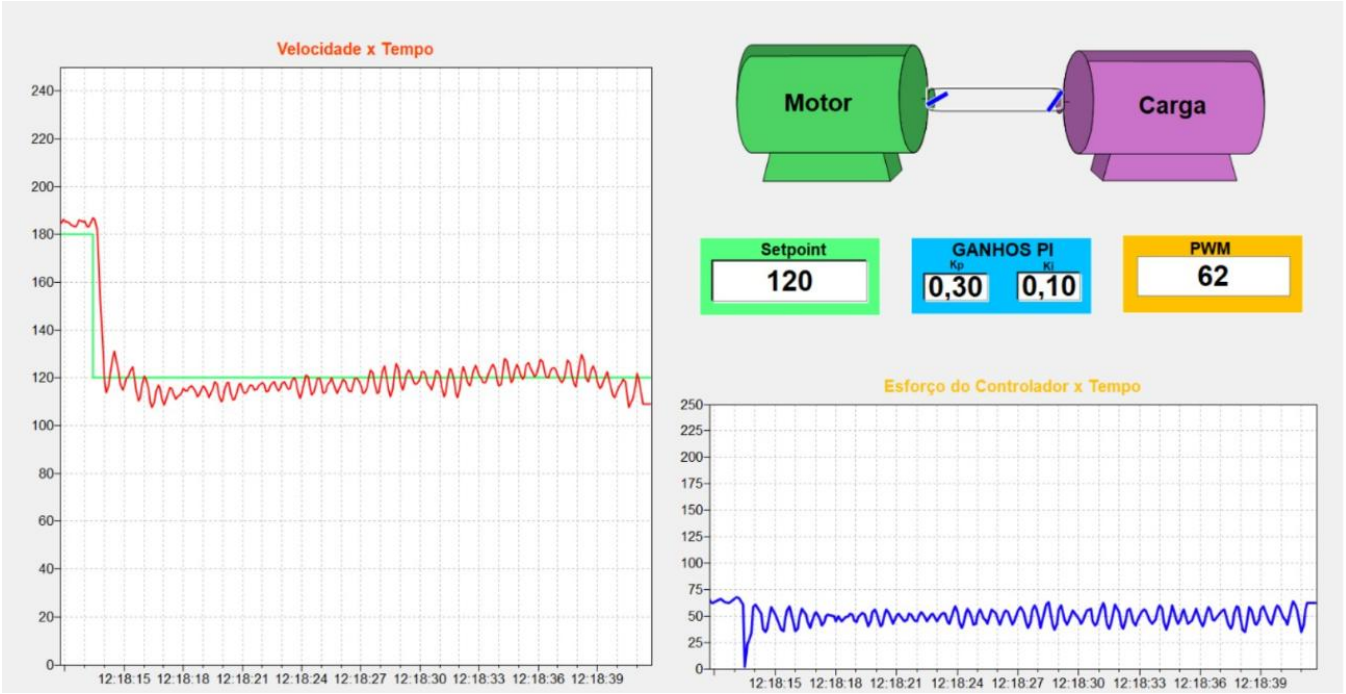
Fonte: Autoria própria

A resposta apresentada na Figura 49, evidencia que a utilização conjunta das ações proporcional e integral, com $K_p = 0{,}30$ e $K_i = 0{,}30$, resulta em um desempenho satisfatório do sistema, caracterizado por tempo de subida reduzido, ausência de *overshoot* significativo e boa estabilização da velocidade em torno do valor de referência. O erro em regime permanente é praticamente nulo, indicando a eficácia da ação integral, enquanto o esforço do controlador se mantém estável após o regime transitório. Esses resultados demonstram que a correta combinação dos ganhos proporcional e integral proporciona um equilíbrio adequado entre rapidez, estabilidade e precisão no controle de velocidade do motor.

5.4.2 Controlador pi com carga

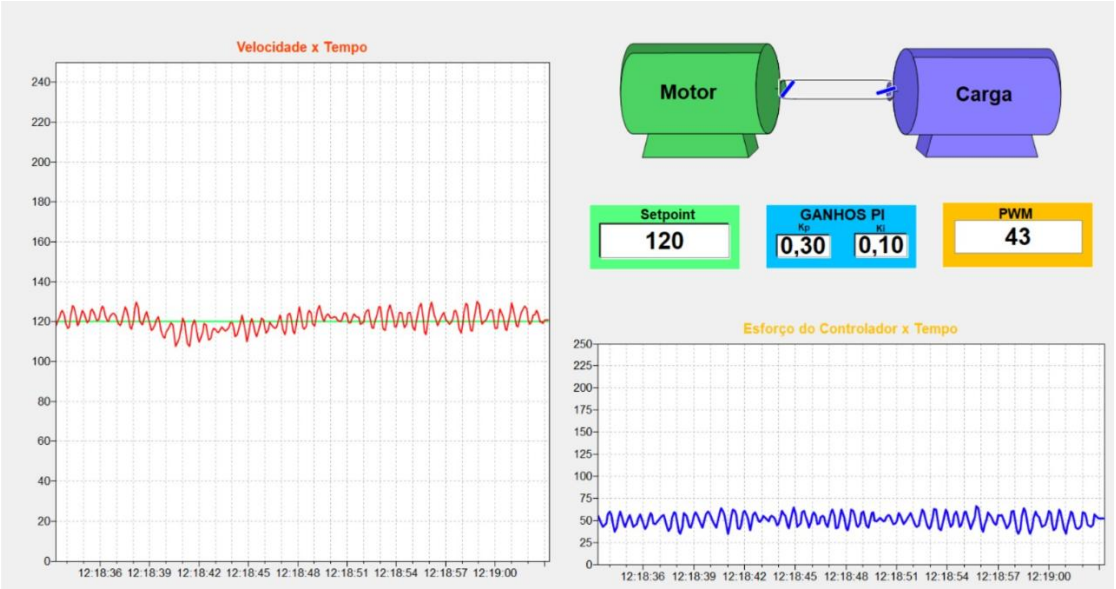
Após a análise dos ganhos dos controladores, observa-se o comportamento do controlador com o acréscimo de diferentes valores de carga, para esse caso nas Figuras 45,46 e 47 apresentam-se a resposta experimental do sistema de controle de velocidade utilizando o controlador PI sob diferentes condições de carga mecânica.

Figura 50: Motor CC com controlador PI e carga baixa



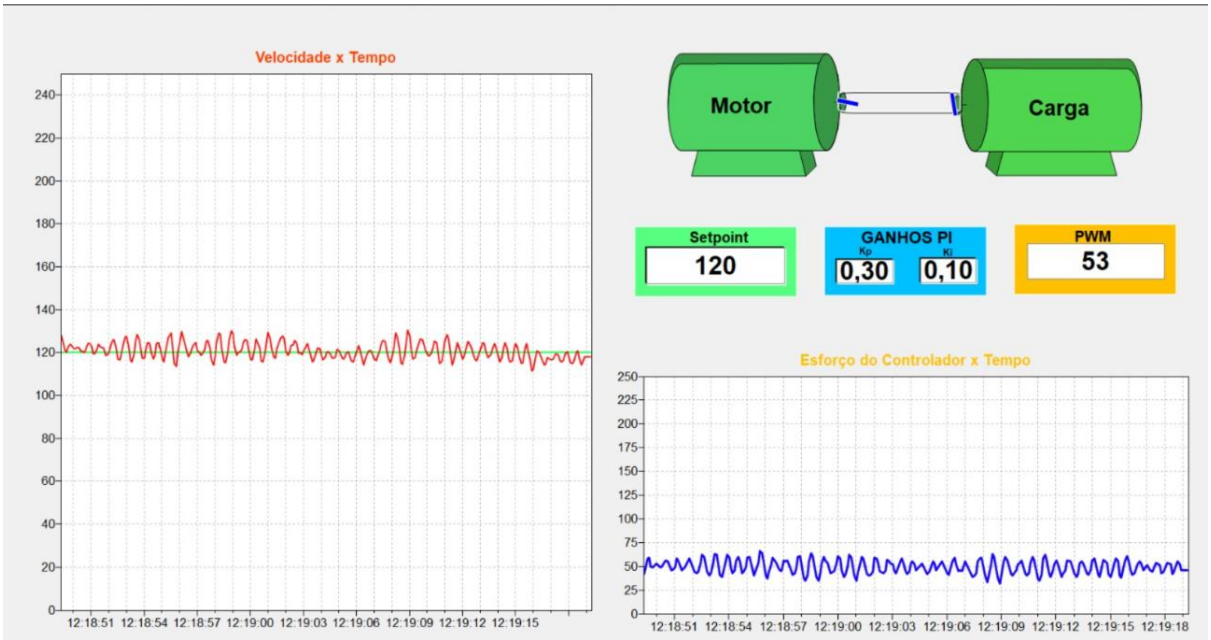
Fonte: Autoria própria

Figura 51:Motor CC com controlador PI e carga média



Fonte: Autoria própria

Figura 52: Motor CC com controlador PI e carga alta



Fonte: Autoria própria

Observa-se que, mesmo com o acréscimo de carga ao eixo do motor, a velocidade permanece próxima ao valor de referência estabelecido, não sendo observadas variações significativas no regime permanente.

Nota-se que a atuação do controlador ocorre principalmente por meio do ajuste do esforço de controle, evidenciado pela variação do sinal PWM aplicado ao motor. Esse comportamento indica que o controlador PI é capaz de rejeitar perturbações de carga,

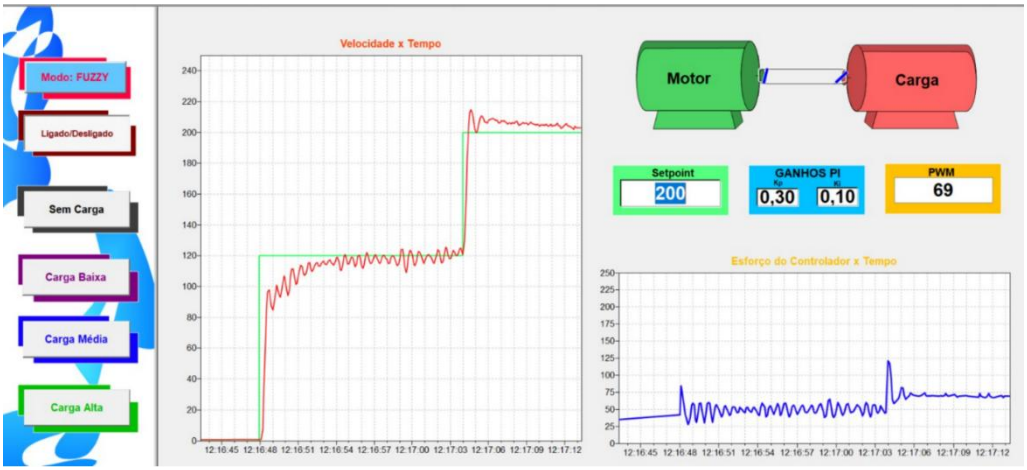
compensando o aumento do Torque resistente sem comprometer a velocidade do eixo. As pequenas oscilações observadas na resposta são atribuídas às características do sistema físico e ao ruído de medição, não afetando o desempenho global do controle. Esses resultados demonstram a robustez do controlador PI frente a variações de carga nas condições experimentais analisadas.

5.5 Controlador *Fuzzy*

Para o teste do controlador *Fuzzy*, verificou-se seu comportamento apenas com o motor a vazio e com o acréscimo de cargas.

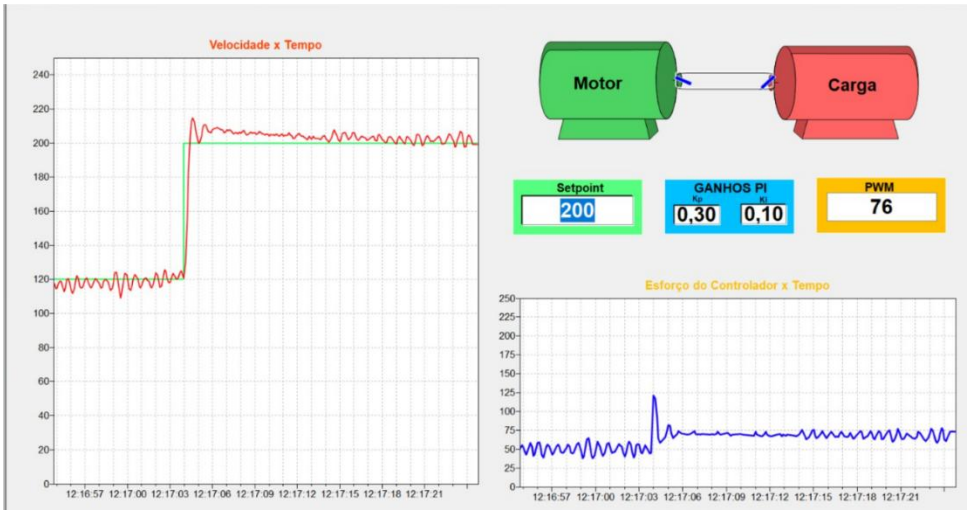
Na Figura 53 apresenta-se a resposta experimental do sistema considerando apenas a variação do valor de referência e sem aplicação de carga mecânica. Observa-se que a velocidade do motor acompanha a mudança de setpoint, passando de 120 rpm para 200 rpm.

Figura 53: Motor CC com controlador Fuzzy



Fonte: Autoria própria

Figura 54: Motor CC com controlador Fuzzy com a velocidade estabilizada



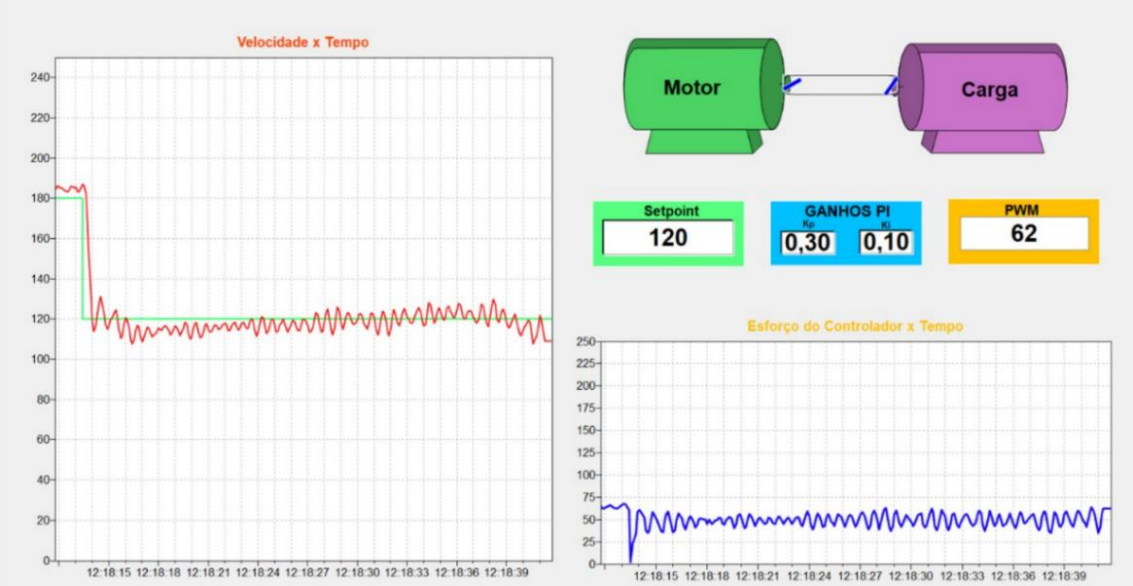
Fonte: Autoria própria.

Analisando as Figuras 53 e 54, nota-se que o controlador Fuzzy apresenta tempo de subida reduzido, com *overshoot* limitado e rápida estabilização da velocidade em torno do valor de referência. As oscilações observadas após a estabilização possuem baixa amplitude e estão associadas às características do sistema físico e ao ruído de medição. O esforço do controlador evidencia uma atuação progressiva do sinal PWM, sem saturações prolongadas, indicando uma ação de controle suave e eficiente. Esses resultados demonstram a capacidade do controlador Fuzzy em regular a velocidade do motor de corrente contínua de forma adequada em condições reais de operação.

5.5.1 Controlador *fuzzy* com acréscimo de carga

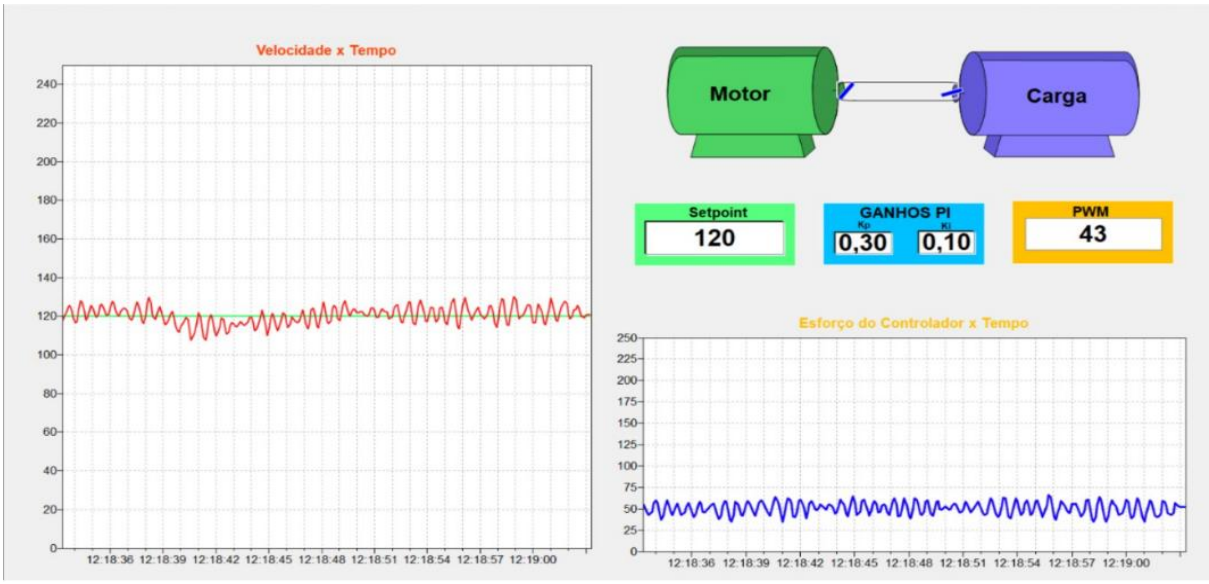
Nas Figuras 55, 56 e 57 são apresentas a resposta experimental do sistema de controle de velocidade utilizando o controlador *Fuzzy* sob diferentes níveis de carga mecânica, correspondentes às condições de carga baixa, média e alta. Observa-se que, independentemente do nível de carga aplicado ao eixo do motor, a velocidade permanece próxima ao valor de referência estabelecido, não sendo identificadas variações significativas no regime permanente.

Figura 55:Motor CC com controlador Fuzzy e carga baixa



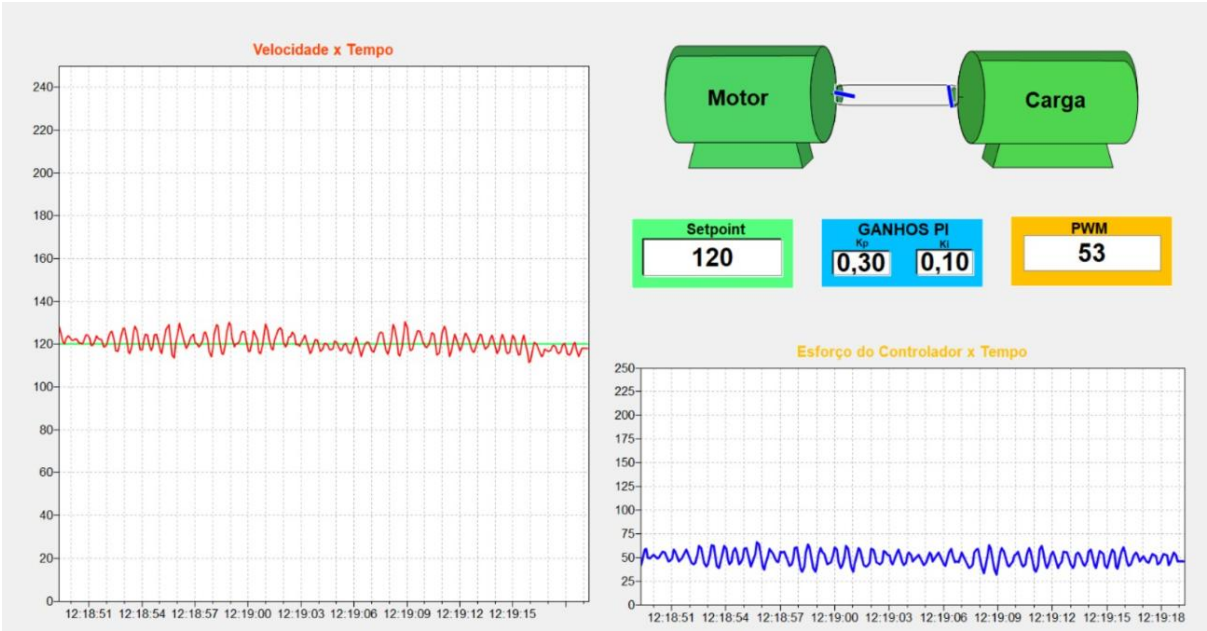
Fonte: Autoria própria.

Figura 56: Motor CC com controlador Fuzzy e carga média



Fonte: Autoria própria.

Figura 57: Motor CC com controlador Fuzzy e carga alta



Fonte: Autoria própria.

Com a análise das Figuras, nota-se que a atuação do controlador *Fuzzy* ocorre predominantemente por meio do ajuste do esforço de controle, evidenciado pela modulação do sinal PWM aplicado ao motor, o qual se adapta conforme o aumento do Torque resistente. Esse comportamento indica que o controlador *Fuzzy* é capaz de rejeitar perturbações de carga de forma eficiente, compensando as variações impostas ao sistema sem comprometer a estabilidade ou a precisão da velocidade.

As pequenas oscilações observadas na resposta da velocidade são atribuídas às características do sistema físico e ao ruído de medição, não afetando o desempenho global do controle. Esses resultados evidenciam a robustez e a suavidade de atuação do controlador *Fuzzy* frente a variações de carga nas condições experimentais analisadas.

8. CONCLUSÃO

Este trabalho apresentou uma análise comparativa entre os controladores Proporcional–Integral (PI) e *Fuzzy* aplicados ao controle de velocidade angular de um motor de corrente contínua. A metodologia envolveu a modelagem e identificação experimental do motor, simulações no ambiente MATLAB/Simulink e a implementação prática em bancada utilizando Arduino Uno, *encoder* incremental, ponte H e supervisão via Elipse E3.

Os resultados demonstram que o controlador PI é capaz de regular a velocidade do motor de forma eficiente, apresentando boa estabilidade, erro em regime permanente reduzido e capacidade de rejeição a perturbações de carga, desde que corretamente sintonizado. Contudo, seu desempenho mostrou-se sensível à variação dos ganhos, exigindo ajustes cuidadosos para evitar sobressinal excessivo ou respostas lentas.

O controlador *Fuzzy* apresentou desempenho superior em termos de robustez e suavidade da resposta, com baixos valores de sobressinal, tempos de acomodação reduzidos e erro em regime permanente praticamente nulo, mesmo sob variações de carga. Sua atuação não linear proporcionou maior adaptabilidade às incertezas do sistema, reduzindo a dependência de um modelo matemático preciso.

Dessa forma, conclui-se que ambos os controladores são viáveis para o controle de velocidade de motores de corrente contínua. O controlador PI destaca-se pela simplicidade e ampla aplicação industrial, enquanto o controlador *Fuzzy* se mostra mais robusto e eficiente em sistemas sujeitos a não linearidades e variações de carga, sendo uma alternativa promissora para aplicações em automação e sistemas embarcados. Assim, a escolha entre as estratégias de controle deve considerar não apenas o desempenho desejado, mas também a complexidade de implementação, os recursos computacionais disponíveis e as características da aplicação final.

Como trabalhos futuros, sugere-se a ampliação do estudo para controladores híbridos, como PI-Fuzzy, a aplicação da metodologia em motores de maior porte e a análise do desempenho sob condições mais severas de perturbação, bem como a implementação de técnicas automáticas de ajuste de parâmetros.

9. REFERÊNCIAS BIBLIOGRÁFICAS

ARDUINO. Documentação de Referência da Linguagem Arduino. 2018. Disponível em https://www.arduino.CC/reference/pt/. Acesso: 03/05/2026.

BARBOSA, João Pedro Pereira; ALVES, Dartiu Rafael Dantas da Silva; ALEXANDRE, Geronimo Barbosa. Análise comparativa dos controladores PID e PID-Fuzzy aplicados na regulação de velocidade de um motor CC. Revista Eletrônica de Engenharia Elétrica e Engenharia Mecânica (R4EM), Mossoró, v. 8, n. 1, p. 95–105, 2026. DOI: https://doi.org/10.21708/issn27635325.v8n1.a115467.2026.

BERTULUCCI, Cristiano**.** *Motor CC: saiba como funciona e de que forma especificar*. Citisystems, 2018. Disponível em: https://www.citisystems.com.br/motor-cc/.

BOSCOLO, Filipe Urban. Controle proporcional-integral da velocidade angular de um motor de corrente contínua. 2019. Trabalho de Graduação (Engenharia Elétrica) – Faculdade de Engenharia de Ilha Solteira, Universidade Estadual Paulista (UNESP), Ilha Solteira, 2019

CHAPMAN, J., S. Fundamentos de Máquinas Elétricas, 5 edições, Porto Alegre, Editora McGrawHill, 2013

FILIPEFLOP, encontrado em https://www.filipeflop.com/. Acesso: 05/10/2018.

Freitas, Carlos Márcio (2014), 'Controle pid em sistemas embarcados', Embarcados. URL: https://embarcados.com.br/controle-pid-em-sistemas-embarcados/

FUENTES, C., R. Apostila de automação. Universidade Federal de Santa Maria, Colégio Técnico Industrial de Santa Maria, 2005.

GREGÓRIO, Lucas Vinícius de Oliveira. Controle Fuzzy de Velocidade de Motor DC em Arduino. 2023.

JAFELICE, R. S. M.; BARROS, L. C.; BASSANEZI, R. C. Usando a teoria dos conjuntos Fuzzy na modelagem de fenômenos biológicos. Natal, 2012.

KOSOW, I, I, Máquinas Elétricas e Transformadores, 15 Edição, UNESP, Ilha Solteira, Editora Globo S.A, 2005.

MULTILÓGICA SHOP, Arduino Guia Iniciante, versão 1.0. Disponível https://edisciplinas.usp.br/pluginfile.php/4110478/mod_resource/content/ 1/Guia_Arduino_Iniciante_Multilógica_Shop.pdf. Acesso: 03/05/2026.

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

PATANÉ, E. J. Implementação de controle de velocidade em malha fechada para motor de corrente contínua utilizando sistema de controle de dados. Dissertação de Mestrado, Escola de Engenharia Mauá do Centro Universitário do Instituto Mauá de Tecnologia, São Caetano do Sul, 2008.

TEIXEIRA C. M, M e Assunção, E, Controle Linear 1, Parte A-sistema contínuos no tempo, Faculdade de Engenharia, Universidade Estadual Paulista "Julio de Mesquita Filho", 2016

FITZGERALD, Arthur Eugene; KINGSLEY JUNIOR, Charles; UMANS, Stephen D. **Máquinas elétricas:** com introdução à eletrônica de potência. 6. ed. Porto Alegre, RS: Bookman, 2006. 648 p. ISBN 9788560031047.2014

Zadeh, Lotfi Asker, George J Klir & Bo Yuan (1996), Fuzzy sets, Fuzzy logic, and Fuzzy systems: selected papers, Vol. 6, World scientific.

APÊNDICE A

Este apêndice apresenta os códigos desenvolvido em ambiente MATLAB.

O código a seguir foi utilizado para a identificação do modelo dinâmico do motor de corrente contínua a partir de dados experimentais. O código realiza a leitura dos dados de entrada e saída do sistema, o pré-processamento dos sinais, a detecção do degrau de entrada, a remoção de offset e a identificação de modelos matemáticos de primeira e segunda ordem. Além disso, são calculados os erros médios quadráticos entre os modelos identificados e os dados reais, bem como gerados gráficos comparativos entre as respostas simuladas e medidas experimentalmente.

```
% Chamando o arquivo com os dados do motor
arquivo = 'teste.csv.xlsx';
tempoEmMs = true;
% Leitura dos dados experimentais
dados = readmatrix(arquivo);
t = dados(:,1);
u = dados(:,2);
y = dados(:,3);
idxValid = ~isnan(t) & ~isnan(u) & ~isnan(y);
t = t(idxValid);
u = u(idxValid);
y = y(idxValid);
if tempoEmMs
    t = t/1000;
end
% Garantindo o tempo crescente e único
[t, idxUnique] = unique(t, 'stable');
u = u(idxUnique);
y = y(idxUnique);
% Criação de uma malha de tempo uniforme
Ts = median(diff(t));
t_uniforme = (t(1):Ts:t(end))';
u_uniforme = interp1(t, u, t_uniforme, 'previous', 'extrap');
y_uniforme = interp1(t, y, t_uniforme, 'linear', 'extrap');
t = t_uniforme;
u = u_uniforme;
y = y_uniforme;
```

```
% Detecção do degrau de entrada
idx_step = find(u > 0, 1, 'first');
if isempty(idx_step)
    error('Nao foi detectado degrau de entrada.');
end
t_step = t(idx_step);
u0 = u(idx_step);
fprintf('Degrau encontrado em t = %.4f s\n', t_step);
fprintf('Amplitude do degrau = %.4f PWM\n', u0);
% Remoção do offset
idx_pre = 1:max(1, idx_step-1);
u_pre = mean(u(idx_pre));
y_pre = mean(y(idx_pre));
u_corr = u - u_pre;
y_corr = y - y_pre;
% Identificação do modelo de 1ª ordem
t_pos = t(idx_step:end) - t_step;
y_pos = y_corr(idx_step:end);
n = length(y_pos);
nTail = max(10, round(0.15*n));
y_final = mean(y_pos(end-nTail+1:end));
K1 = y_final / u0;
y_tau = 0.632 * y_final;

idx_tau = find(y_pos >= y_tau, 1, 'first');
if isempty(idx_tau)
    error('Nao foi possivel detectado o ponto de 63,2%%.');
end
tau1 = t_pos(idx_tau);
G1 = tf(K1, [tau1 1]);
y1 = zeros(size(t));
for k = 1:length(t)
    if t(k) >= t_step
        y1(k) = K1*u0*(1 - exp(-(t(k)-t_step)/tau1));
    end
end
erro1 = mean((y_corr - y1).^2);
% Identificação via System Identification Toolbox
data_id = iddata(y_corr, u_corr, Ts);
```

```
% Modelo de 2ª ordem sem zero
sys2_sem_zero = tfest(data_id, 2, 0);
y2_sem_zero = lsim(sys2_sem_zero, u_corr, t);
erro2_sem_zero = mean((y_corr - y2_sem_zero).^2);
% Modelo de 2ª ordem com zero
sys2_com_zero = tfest(data_id, 2, 1);
y2_com_zero = lsim(sys2_com_zero, u_corr, t);
erro2_com_zero = mean((y_corr - y2_com_zero).^2);
% Exibição dos resultados
disp('Modelo de 1ª ordem:');
G1
disp('Modelo de 2ª ordem sem zero:');
sys2_sem_zero
disp('Modelo de 2ª ordem com zero:');
sys2_com_zero
fprintf('\nErro modelo 1ª ordem        = %.6f\n', erro1);
fprintf('Erro modelo 2ª ordem s/zero = %.6f\n', erro2_sem_zero);
fprintf('Erro modelo 2ª ordem c/zero = %.6f\n', erro2_com_zero);
% Geração dos gráficos
figure;
plot(t, y_corr, 'k', 'LineWidth', 2); hold on;
plot(t, y1, 'r--', 'LineWidth', 1.5);
plot(t, y2_sem_zero, 'b--', 'LineWidth', 1.5);
plot(t, y2_com_zero, 'g--', 'LineWidth', 1.5);
grid on;
xlabel('Tempo (s)');
ylabel('RPM');
legend('Dados medidos', '1ª ordem', '2ª ordem sem zero', '2ª ordem com
zero', ...
       'Location', 'best');
title('Identificação do modelo do motor CC');
```

Posteriormente, foi desenvolvido um código Matlab para identificar os parâmetros do sistema de controle mostrados das equações (26), (27), (28), (29) bem como o erro em regime permanente.

```
% Leitura dos dados provenientes do Scope do Simulink
dados = out.ScopeData1.signals.values;
```

```
ref = dados(:,1);   % Sinal de referência
y   = dados(:,2);   % Sinal de saída
t = out.ScopeData1.time;  % Vetor de tempo
% Cálculo dos índices de desempenho
info = stepinfo(y, t);
erro_ss = abs(ref(end) - y(end));
% Exibição dos resultados no console
fprintf('\n===== ANALISE DO CONTROLADOR =====\n');
fprintf('Overshoot: %.3f %%\n', info.Overshoot);
fprintf('Tempo de subida: %.3f s\n', info.RiseTime);
fprintf('Tempo de acomodacao: %.3f s\n', info.SettlingTime);
fprintf('Tempo de pico: %.3f s\n', info.PeakTime);
fprintf('Valor de pico: %.3f\n', info.Peak);
fprintf('Erro em regime permanente: %.6f\n', erro_ss);
% Geração do gráfico de referência e saída
figure
plot(t, ref, 'r', 'LineWidth', 2); hold on
plot(t, y, 'b', 'LineWidth', 1.5);
grid on
xlabel('Tempo (s)');
ylabel('Sinal');
title('Referencia x Saida');
legend('Referencia', 'Saida');
```

APÊNDICE B

Este apêndice apresenta o código-fonte desenvolvido e utilizado na plataforma Arduino para a implementação prática dos controladores Proporcional-Integral (PI) e Fuzzy, bem como para a comunicação serial com o *Software* supervisório Elipse E3 por meio do protocolo Modbus RTU.

O código contempla a leitura dos sinais provenientes do *encoder* incremental, o cálculo da velocidade do motor, a execução dos algoritmos de controle selecionados e o envio e recebimento de dados entre o Arduino e o supervisório. Adicionalmente, são definidas as configurações de comunicação, a pinagem utilizada, a lógica de seleção do controlador e os registradores Modbus empregados durante os testes experimentais.

```
#include <math.h>
#include <ModbusRtu.h>
// DEFININDO CONSTANTES NO MODBUS
#define SLAVE_ID 1
#define BAUD_RATE 9600
#define NUM_REGS 12
uint16_t modbusData[NUM_REGS];
Modbus slave(SLAVE_ID, Serial, 0);
// PINAGEM REAL ENTRE OS MOTORES O ARDUINO E A PONTE H
// Motor A (principal): ENA=D5, IN1=D7, IN2=D6
// Motor B (carga):     ENB=D10, IN3=D8, IN4=D9
// Encoder: D3
const int ENA = 5;
const int IN1 = 7;
const int IN2 = 6;
const int ENB = 10;
const int IN3 = 8;
const int IN4 = 9;
const int ENC_PIN = 3;
// PARÂMETROS GERAIS PARA OS DOIS CONTROLADORES
const unsigned int PPR = 341;
const unsigned long TsControl_ms = 100;
volatile unsigned long pulsos = 0;
float rpm = 0.0f;
float rpm_f_PI = 0.0f;
float rpm_f_Fuzzy = 0.0f;
```

```
int pwmA = 0;
bool ligado = false;
int cargaPWM = 0;
int cargaPWM_anterior = 0;
unsigned long lastControl = 0;
int modoControle = 0; // 0 = PI | 1 = FUZZY
int modoControleAnterior = 0;
// PARÂMETROS DO CONTROLADOR PI
float Kp = 0.3f;
float Ki = 0.10f;
float integradorPI = 0.0f;
float pwm_base_PI = 10.0f;    // PWM base para vencer atrito/zona morta
static int pwmA_filtrado_PI = 0;
const int PWM_STEP_PI = 15;
// PARÂMETROS FUZZY
const int PWM_MIN_PARTIDA = 72;
const int RPM_LIMIAR_MOV = 8;
const int PWM_MAX_TESTE = 210;
// ESTADOS DO CONTROLE FUZZY
float erroAnteriorFuzzy = 0.0f;
float derro_f = 0.0f;
// Integral para eliminar o erro residual
float iTermFuzzy = 0.0f;
const float Ki_Fuzzy = 0.12f;
const float ITERM_MAX = 40.0f;
// Bandas mortas e zona livre
const float ERRO_DEADBAND = 1.0f;
const float DERRO_DEADBAND = 3.0f;
const float ERRO_PAZ = 1.2f;
const float DERRO_PAZ = 4.0f;
const int PWM_HISTERESIS = 1;
// Universo de saída Fuzzy (-60 a 60)
const int N_OUT = 81;
float y_out[N_OUT];
float pwm_hold = 0.0f;
int pwmAnteriorFuzzy = 0;
// BASE DE REGRAS FUZZY
const byte ruleBase[5][5] = {
  {0, 0, 0, 1, 2},
```

```
  {0, 1, 1, 2, 3},
  {0, 1, 2, 3, 4},
  {1, 2, 3, 3, 4},
  {2, 3, 4, 4, 4}
};
// PROTÓTIPOS
float max2(float a, float b);
float min2(float a, float b);
float limita(float x, float xmin, float xmax);
float trimf(float x, float a, float b, float c);
void inicializaUniversoSaida();
float pwmBaseFromSetpoint(float sp);
float FuzzyControlCorrecao(float erro, float derro);
void resetPI();
void resetFuzzy();
void resetTodosControladores();
int controlePI(float sp, float rpmAtual, float dt);
int controleFuzzy(float sp, float rpmAtual, float dt, bool cargaMudou);
void isr_encoder();
// SETUP
void setup() {
  Serial.begin(BAUD_RATE);
  while (!Serial);
  slave.start();
  // Valores iniciais no array único
  modbusData[0] = 100; // 40001 - Setpoint RPM
  modbusData[1] = 0;   // 40002 - Modo (0=PI)
  modbusData[2] = 0;   // 40003 - Carga PWM
  modbusData[3] = 30;  // 40004 - Kp x100
  modbusData[4] = 100; // 40005 - Ki x1000
  modbusData[5] = 0;   // 40006 - Start/Stop
  for (int i = 6; i < NUM_REGS; i++) {
    modbusData[i] = 0;
  }
  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN3, OUTPUT);
```

```
  pinMode(IN4, OUTPUT);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  pinMode(ENC_PIN, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(ENC_PIN), isr_encoder, RISING);
  analogWrite(ENA, 0);
  analogWrite(ENB, 0);
  inicializaUniversoSaida();
  resetTodosControladores();
}
// LOOP
void loop() {
  slave.poll(modbusData, NUM_REGS);
  // Verificando os dados do Elipse
  int setpointRPM = (int)modbusData[0];
  modoControle = (int)modbusData[1];
  cargaPWM = (int)modbusData[2];
  Kp = (float)modbusData[3] / 100.0f;
  Ki = (float)modbusData[4] / 1000.0f;
  ligado = (modbusData[5] > 0);
  if (modoControle != modoControleAnterior) {
    resetTodosControladores();
    modoControleAnterior = modoControle;
  }
  unsigned long now = millis();
  if (now - lastControl >= TsControl_ms) {
    lastControl = now;
    noInterrupts();
    unsigned long p = pulsos;
    pulsos = 0;
    interrupts();
    float dt = TsControl_ms / 1000.0f;
    float rpm_medida = (60.0f * (float)p) / ((float)PPR * dt);
    if (modoControle == 0) {
      rpm_f_PI = 0.2f * rpm_f_PI + 0.80f * rpm_medida;
      rpm = rpm_f_PI;
    } else {
```

```
      rpm_f_Fuzzy = 0.7f * rpm_f_Fuzzy + 0.3f * rpm_medida;
      rpm = rpm_f_Fuzzy;
    }
    analogWrite(ENB, cargaPWM);
    bool cargaMudou = (cargaPWM != cargaPWM_anterior);
    if (!ligado) {
      pwmA = 0;
      resetTodosControladores();
      analogWrite(ENA, 0);
    } else {
      if (modoControle == 0) {
        pwmA = controlePI((float)setpointRPM, rpm, dt);
      } else {
        pwmA = controleFuzzy((float)setpointRPM, rpm, dt, cargaMudou);
      }
      analogWrite(ENA, pwmA);
    }
    cargaPWM_anterior = cargaPWM;
    float erroAtual = (float)setpointRPM - rpm;
    // Atualiza registradores para o Elipse
    modbusData[6] = (uint16_t)(rpm * 10.0f);
    modbusData[7] = (uint16_t)pwmA;
    modbusData[8] = (uint16_t)((int16_t)(erroAtual * 10.0f));
    modbusData[9] = (uint16_t)modoControle;
    modbusData[10] = (uint16_t)cargaPWM;
    modbusData[11] = (uint16_t)((int16_t)(iTermFuzzy * 100.0f));
  }
}
// FUNÇÕES AUXILIARES
float max2(float a, float b) { return (a > b) ? a : b; }
float min2(float a, float b) { return (a < b) ? a : b; }
float limita(float x, float xmin, float xmax) {
  if (x < xmin) return xmin;
  if (x > xmax) return xmax;
  return x;
}
float trimf(float x, float a, float b, float c) {
  if (x <= a || x >= c) return 0.0f;
  if (x == b) return 1.0f;
```

```
  if (x < b) return (x - a) / (b - a);
  return (c - x) / (c - b);
}
float pwmBaseFromSetpoint(float sp) {
  if (sp <= 0.0f) return 0.0f;
  float pwmBase = 0.35f * sp;
  return limita(pwmBase, 0.0f, 255.0f);
}
// UNIVERSO DE SAÍDA (-60 a +60)
void inicializaUniversoSaida() {
  for (int k = 0; k < N_OUT; k++) {
    y_out[k] = -60.0f + k * (120.0f / (N_OUT - 1));
  }
}
// correção Fuzzy
float FuzzyControlCorrecao(float erro, float derro) {
  erro  = limita(erro,  -220.0f, 220.0f);
  derro = limita(derro, -300.0f, 300.0f);
  float muE[5];
  muE[0] = trimf(erro, -220.0f, -220.0f, -100.0f);
  muE[1] = trimf(erro, -150.0f, -60.0f, 0.0f);
  muE[2] = trimf(erro, -30.0f, 0.0f, 30.0f);
  muE[3] = trimf(erro, 0.0f, 60.0f, 150.0f);
  muE[4] = trimf(erro, 100.0f, 220.0f, 220.0f);
  float muDE[5];
  muDE[0] = trimf(derro, -300.0f, -300.0f, -100.0f);
  muDE[1] = trimf(derro, -150.0f, -60.0f, 0.0f);
  muDE[2] = trimf(derro, -30.0f, 0.0f, 30.0f);
  muDE[3] = trimf(derro, 0.0f, 60.0f, 150.0f);
  muDE[4] = trimf(derro, 100.0f, 300.0f, 300.0f);
  float numerador = 0.0f;
  float denominador = 0.0f;
  for (int k = 0; k < N_OUT; k++) {
    float y = y_out[k];
    float muAgregado = 0.0f;
    for (int i = 0; i < 5; i++) {
      for (int j = 0; j < 5; j++) {
        float firing = min2(muE[i], muDE[j]);
        if (firing <= 0.0f) continue;
```

```
        byte outSet = ruleBase[i][j];
        float muOut = 0.0f;
        switch (outSet) {
          case 0: muOut = trimf(y, -60.0f, -45.0f, -25.0f); break;
          case 1: muOut = trimf(y, -35.0f, -15.0f, 0.0f);   break;
          case 2: muOut = trimf(y, -10.0f, 0.0f, 10.0f);    break;
          case 3: muOut = trimf(y, 0.0f, 15.0f, 35.0f);     break;
          case 4: muOut = trimf(y, 25.0f, 45.0f, 60.0f);    break;
        }
        float regraImp = min2(firing, muOut);
        muAgregado = max2(muAgregado, regraImp);
      }
    }
    numerador += y * muAgregado;
    denominador += muAgregado;
  }
  if (denominador <= 0.001f) return 0.0f;
  return numerador / denominador;
}
// RESETs -- QUANDO TROCAR O TIPO DE CONTROLADOR DÁ UM RESET PRIMEIRO
void resetPI() {
  integradorPI = 0.0f;
  pwmA_filtrado_PI = 0;
  rpm_f_PI = 0.0f;
}
void resetFuzzy() {
  erroAnteriorFuzzy = 0.0f;
  derro_f = 0.0f;
  iTermFuzzy = 0.0f;
  pwm_hold = 0.0f;
  pwmAnteriorFuzzy = 0;
  rpm_f_Fuzzy = 0.0f;
}
void resetTodosControladores() {
  resetPI();
  resetFuzzy();
}
// CONTROLE PI
```

```
int controlePI(float sp, float rpmAtual, float dt) {
  float erro = sp - rpmAtual;
  integradorPI += erro * dt;
  if (integradorPI > 1200.0f) integradorPI = 1200.0f;
  if (integradorPI < -1200.0f) integradorPI = -1200.0f;
  float u = pwm_base_PI + Kp * erro + Ki * integradorPI;
  if (sp <= 0.0f) {
    u = 0.0f;
    integradorPI = 0.0f;
  }
  if (u > 255.0f) u = 255.0f;
  if (u < 0.0f) u = 0.0f;
  int pwm_desejado = (int)u;
  if (pwm_desejado > pwmA_filtrado_PI + PWM_STEP_PI) {
    pwmA_filtrado_PI += PWM_STEP_PI;
  } else if (pwm_desejado < pwmA_filtrado_PI - PWM_STEP_PI) {
    pwmA_filtrado_PI -= PWM_STEP_PI;
  } else {
    pwmA_filtrado_PI = pwm_desejado;
  }
  return pwmA_filtrado_PI;
}
// CONTROLE FUZZY - (lógica do loop)
int controleFuzzy(float sp, float rpmAtual, float dt, bool cargaMudou) {
  float erro = (float)sp - rpmAtual;
  if (fabs(erro) < ERRO_DEADBAND) erro = 0.0f;
  float derro = (erro - erroAnteriorFuzzy) / dt;
  derro_f = 0.7f * derro_f + 0.3f * derro;
  if (fabs(derro_f) < DERRO_DEADBAND) derro_f = 0.0f;
  // Integral ativa para erros de até 30 RPM
  if (fabs(erro) < 30.0f) {
    iTermFuzzy += Ki_Fuzzy * erro * dt;
    iTermFuzzy = limita(iTermFuzzy, -ITERM_MAX, ITERM_MAX);
  } else {
    iTermFuzzy *= 0.95f; // "Esfria" a integral se o erro for gigante
(anti-windup)
  }
  float pwm_base = pwmBaseFromSetpoint((float)sp);
  float correcao = FuzzyControlCorrecao(erro, derro_f);
  float pwm_Fuzzy = pwm_base + correcao + iTermFuzzy;
```

```
  pwm_Fuzzy = limita(pwm_Fuzzy, 0.0f, (float)PWM_MAX_TESTE);
  bool erroRelevante = (fabs(erro) > 2.0f);
  if (!erroRelevante && fabs(derro_f) < DERRO_PAZ) {
    pwm_Fuzzy = pwm_hold;
  } else {
    pwm_hold = pwm_Fuzzy;
  }
  pwmA = (int)pwm_Fuzzy;
  erroAnteriorFuzzy = erro;
  return pwmA;
}
// INTERRUPÇÃO DO ENCODER
void isr_encoder() {
  pulsos++;
}
```