UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
BACHARELADO EM ENGENHARIA ELÉTRICA

CARLOS DA CONCEIÇÃO CASTILHO NETO

TÉCNICAS ADAPTATIVAS DE CONTROLE APLICADAS A UM MOTOR BLDC BASEADAS EM LÓGICA FUZZY E SUA OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA
2020

CARLOS DA CONCEIÇÃO CASTILHO NETO

TÉCNICAS ADAPTATIVAS DE CONTROLE APLICADAS A UM MOTOR BLDC BASEADAS EM LÓGICA FUZZY E SUA OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS

Trabalho de Conclusão de Curso apresentado(a) como requisito parcial à obtenção do título de Bacharel em Engenharia Elétrica, do Departamento Acadêmico de Eletrônica, da Universidade Tecnológica Federal do Paraná.

Orientador(a): Prof(a). Dr(a). Fernanda Cristina Corrêa

PONTA GROSSA
2020

Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEP. ACADEMICO DE ELETRONICA-PG

TERMO DE APROVAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO - TCC

TÉCNICAS ADAPTATIVAS DE CONTROLE APLICADAS A UM MOTOR BLDC BASEADAS EM LÓGICA FUZZY E SUA OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS

Por

Carlos da Conceição Castilho Neto

Monografia apresentada às .14 horas 00 min. do dia 20 de agosto de 2020 como requisito parcial, para conclusão do Curso de Engenharia Elétrica da Universidade Tecnológica Federal do Paraná, Câmpus Ponta Grossa. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação e conferidas, bem como achadas conforme, as alterações indicadas pela Banca Examinadora, o trabalho de conclusão de curso foi considerado APROVADO.

Banca examinadora:

Prof. Drª. Cristhiane Gonçalves	Membro
Prof. Drª. Virgínia Helena Varotto Baroncini	Membro
Prof. Drª Fernanda Cristina Côrrea	Orientador
Prof. Dr. Josmar Ivanqui	Professor(a) responsável TCCII

Documento assinado eletronicamente por **FERNANDA CRISTINA CORREA**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 20/08/2020, às 15:41, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **CRISTHIANE GONCALVES**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 20/08/2020, às 15:54, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **VIRGINIA HELENA VAROTTO BARONCINI**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 20/08/2020, às 16:00, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **JOSMAR IVANQUI**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 20/08/2020, às 16:44, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.utfpr.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1566095** e o código CRC **476F7AB3**.

Referência: Processo nº 23064.021957/2020-01 SEI nº 1566095

Dedico este trabalho a minha família e aos meus amigos, pelos momentos de ausência.

AGRADECIMENTOS

Este trabalho não poderia ser terminado sem a ajuda de diversas pessoas e instituições, às quais presto minha homenagem. Certamente esses parágrafos não irão atender a todas as pessoas que fizeram parte dessa importante fase de minha vida. Portanto, desde já peço desculpas àquelas que não estão presentes entre estas palavras, mas elas podem estar certas que fazem parte do meu pensamento e de minha gratidão.

Em primeiro lugar a Deus por me fortalecer e me proporcionar todas as coisas necessárias para atingir esta meta.

A minha esposa Monique, meus pais e avós, pelo carinho, incentivo e total apoio em todos os momentos da minha vida que foram fundamentais para a realização deste sonho.

A minha orientadora, Professora Fernanda, que me mostrou os caminhos a serem seguidos, pela confiança depositada e pelo real desejo de me ajudar a compreender todo o conteúdo abordado neste trabalho.

A todos os professores e colegas do departamento, que ajudaram de forma direta e indireta na conclusão deste trabalho, dentre eles Diego Solak Castanho, Lenon Diniz Seixas e Rafael Schmidt Baumel.

A Universidade Tecnológica Federal do Paraná (UTFPR) por disponibilizar uma ótima estrutura organizacional, laboratórios e ferramentas necessárias para realizar este estudo.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pelo apoio financeiro.

Enfim, a todos os que de alguma forma contribuíram para a realização deste trabalho.

Reconheça o que há de bom nos outros,
não as manchas. Às vezes, as manchas
precisam da devida atenção para ser
limpas, mas sempre edifique sobre as
virtudes da pessoa.
(SCOTT, Richard, 2013)

RESUMO

CASTILHO NETO, Carlos da Conceição. **Técnicas adaptativas de controle aplicadas a um motor BLDC baseadas em lógica Fuzzy e sua otimização por enxame de partículas**. 2020. 67 f. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Elétrica) – Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2020.

O uso de motores de corrente contínua sem escovas (BLDC) é cada vez mais recorrente em aplicações industriais como o setor automobilístico, aviação e robótica. Em tais aplicações o motor BLDC é exposto a muitos tipos de distúrbios de carga o que faz com que os métodos de controles convencionais, como o controlador proporcional-integral-derivativo PID, não alcancem suas variáveis com precisão em casos de perturbação súbita e variação de parâmetros. Desta maneira, o controlador PID pode ter seu desempenho melhorado com a aplicação de técnicas adaptativas que coletam dados do ambiente de operação do sistema e realizam ajustes baseadas na condição em que o mesmo se encontra, assim minimizando falhas no sistema. Uma dessas técnicas é a da lógica difusa ou comumente dita lógica Fuzzy, entretanto, a obtenção dos parâmetros dos sistemas Fuzzy demanda de conhecimento por parte do projetista, de forma que ao final do projeto não se tem a certeza de que os valores ótimos foram atingidos. Estes parâmetros podem ser obtidos através do uso de técnicas de otimização, como o Particle Swarm Optimization (PSO), garantindo assim um melhor desempenho do sistema de controle. Tem-se como objetivo realizar neste trabalho a comparação entre diferentes técnicas de controle, como PID, Híbrido Fuzzy-PID e Híbrido Fuzzy-PID otimizado pelo PSO para o controle de velocidade de um motor BLDC, por meio de simulações realizadas no software Simulink e a sua implementação prática em um microcontrolador ESP32.

Palavras-chave: *F*uzzy. PID. Otimização por Enxame de Partículas. BLDC.

ABSTRACT

CASTILHO NETO, Carlos da Conceição. **Adaptive techniques of control applied to BLDC motor based on fuzzy logic and it's particle swarm optimization**. 2020. 67 p. Final Coursework (Bachelor's Degree in Course Name) – Federal University of Technology – Paraná. Ponta Grossa, 2020.

The usage of *Brushless Direct Current Electric Motor* (BLDC) motors is each time more frequent in industrial appliances such as the automobilistic, aviation and robotics segment. In such applications the BLDC motor is exposed to many types of charge disturbance which makes the conventional control methods, such as proportional–integral–derivative controller (PID), not reaching its variables with precision in cases of sudden perturbation and variation of the parameters. Thus, the PID controler might have its performance improved with the application of adaptative techniques which collect data from the operation system environment and make adjusts based in the condition where it is found, therefore minimizing flaws in the system. One of those techniques is the Fuzzy logic, however, the achievement of the parameters of the Fuzzy systems demands knowledge from the designer, this way at the end of the project there is no certainty that the optimum values have been reached. These parameters can be obtained through the usage of optimization techniques, such as the *Particle Swarm Optimization* (PSO), ensuring a better perfomance of the control system. This paper aims to accoplish the comparison between different control techniques, such as PID, Hibrid Fuzzy-PID e Hibrid Fuzzy-PID optimized by the PSO for the speed control in a BLDC motor, through simulations realizend on the Simulink software and its practical implementation in a ESP32 microcontroller.

Keywords: Fuzzy. PID. Particle Swarm Optimization. BLDC.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

SIGLAS

BLDC	*Brushless Direct Current Electric Motor*
CC	Corrente Contínua
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
ESC	*Electronic Speed Controller*
PI	Proporcional-Integral
PID	Proporcional-Integral-Derivativo
PSO	*Particle Swarm Optimization*
PWM	*Pulse Width Modulation*
RPM	Rotações Por Minuto
UTFPR	Universidade Tecnológica Federal do Paraná
ZOH	*Zero-Order Hold*

LISTA DE SÍMBOLOS

LETRAS LATINAS

$c1$	Parâmetro cognitivo	
de	Derivada do sinal de erro do controlador	
E	Sinal de erro do controlador	
$e_(k-1)$	Erro da amostra passada	
e_k	Erro a amostra atual	
$G(s)$	Função de transferência da planta no plano S	
$K(z)$	Função de transferência do controlador	
$P(z)$	Função de transferência da planta no plano Z	
$U(z)$	Equação a Diferenças	
$u_(k-1)$	Esforço de controle passado	
u_k	Esforço de controle	
$c2$	Parâmetro social	
$e(t)$	Sinal de erro	
$gBest$	Parâmetro população	
K_d	Ganho derivativo	
K_i	Ganho integral	
K_p	Ganho proporcional	
$pBest$	Parâmetro cognitivo	
PSS	Percentual de sobressinal	
$R\%$	Resolução	$[\%]$
T_e	Tempo de estabilização	[s]
T_s	Período de amostragem	[s]
$u(t)$	Sinal de controle	

LETRAS GREGAS

α	Zero do controlador PI discreto	
ω	Constante de inércia	
ω_d	Frequência natural Amortecida	[rad/s]
ω_n	Frequência natural	[rad/s]
π	Pi (constante circular)	[rad]
θ_k	Fase do controlador	
θ_n	Fase do numerador do controlador	
θ_p	Fase da planta	
ζ	Fator de amortecimento	

SUMÁRIO

1 INTRODUÇÃO

Com o início da segunda revolução industrial, onde o carvão começa a ser substituído pela eletricidade, a energia a vapor pelo petróleo e o aço pelo ferro as máquinas elétricas começam a ganhar espaço na indústria.

Para o desenvolvimento dos modelos existentes hoje em dia foi exigido uma contribuição de diversos cientistas e pesquisadores, dentre eles pode-se destacar Morit Hermann von Jacobi que em 1839 desenvolveu um motor de 1000 watts de potência e utilizou-o em um barco, onde o mesmo era alimentado por baterias de zinco-platina que pesavam juntas mais de 200 Quilos (DOPPELBAUER, 1822).

Nesta época, já ficava evidente que o uso de baterias seria umas das barreiras que impediam a substituição dos motores a vapor por motores elétricos em em diversas aplicações devido ao seu alto custo e baixo rendimento.

Desta maneira, podemos destacar Werner von Siemens que em 1866 construiu uma máquina elétrica sem ímã permanente com uma potência de aproximadamente 30 watts, esta máquina poderia funcionar como um gerador utilizando o efeito da autoexcitação e como um motor desde que fosse aplicado em seus terminais uma Corrente Contínua (CC) (DOPPELBAUER, 1822).

Em 1889 e com os avanços dos estudos sobre a eletricidade o engenheiro eletricista russo Michael von Dolivo Dobrowolsky desenvolve um motor trifásico com rotor de gaiola, um motor que requisitava menos manutenção, silencioso e com rendimento aproximado de 80% (DOPPELBAUER, 1822).

Em paralelo se desenvolviam técnicas de controle desde o controle de vazão para regular um relógio d'água ao controle de velocidade da pedra de moagem em um moinho movido pelo vento (FRANKLIN; POWELL; EMAMI-NAEINI, 2013).

Com o decorrer dos anos várias técnicas de controle foram desenvolvidas e utilizadas na indústria com o propósito de obter máximo desempenho e eficiência de motores elétricos.

Dentre a variedade de técnicas de controle em malhas industriais para o controle de acionamentos de motores de corrente contínua (CC) o método Ziegler-Nichols é o mais utilizado por possuir robustez e ajuste de forma fácil e empírica (KUMAR; SWAIN; NEOGI, 2017).

No entanto, com o avanço da tecnologia a substituição de motores CC por motores de corrente contínua sem escovas (BLDC) em determinadas aplicações é cada vez mais frequente (GHANY; SHAMSELDIN; GHANY, 2017).

Maior vida útil, menor emissão de ruídos e alta resposta dinâmica são alguns atributos que fazem os motores BLDC serem atrativos para as áreas de veículos elétricos, robótica e aviação. Em tais aplicações o motor BLDC é exposto a muitos tipos de distúrbios de carga o que faz com que os métodos de controles convencionais, como o controlador Proporcional-Integral-Derivativo (PID), não alcancem o desempenho desejado com precisão em casos de perturbação súbita e variação de parâmetros (GHANY; SHAMSELDIN; GHANY, 2017).

No entanto, o controlador PID pode ter seu desempenho melhorado com a aplicação de técnicas adaptativas como a lógica *Fuzzy*, necessária para sistemas dinâmicos em ambientes instáveis. Desta maneira a união de um controlador PID a um controlador *Fuzzy* resulta em um controlador conhecido como controlador híbrido *Fuzzy*-PID. Nesta estrutura híbrida, o controlador *Fuzzy*-PID integra a vantagem de ambas estruturas de controle possibilitando um aprimoramento no controle PID mesmo quando os parâmetros da planta variam ou uma pertubação ocorre (GOSWAMI; JOSHI, 2018).

De acordo com Simões e Shaw (2007) a sintonia de um controlador *Fuzzy* exige do projetista pleno conhecimento do sistema no qual será implementado, o que pode fazer com que as escolhas que norteiem o desenvolvimento do mesmo não sejam as ideais.

Uma das alternativas para se evitar o empirismo característico de sistemas *Fuzzy* seria a utilização de algoritmos de otimização, como a otimização por enxame de partículas (PSO), que cria um universo de possíveis resultados e em algumas iterações pode conseguir os melhores valores para as funções de pertinência e seus respectivos graus, bem como a organização da base de regras.

1.1 DELIMITAÇÃO DO TEMA

Os veículos elétricos começaram a ser fabricados por volta de 1835, atingindo seu auge em 1912, onde 33% da frota americana possuía motores elétricos. Entretanto, em 1920 com a descoberta de grandes reservas petrolíferas, incentivo governa-

mental e a produção em massa criada por Henry Ford, os veículos de combustão interna possuíam valores mais competitivos no mercado, além de possuírem uma maior autonomia, o que fez com que a popularidade dos veículos elétricos diminuísse, pois eram mais lentos, não passavam de 20 km/h, possuíam baixa autonomia e custavam duas vezes mais que os veículos a combustão interna, (MATULKA, 2014).

Com o passar dos anos o uso de derivados do petróleo resultaram em uma grande emissão de poluentes ao meio ambiente, como também o desenvolvimento de uma alta dependência de uma fonte de energia não renovável. Para tal adversidade, a sustentabilidade tem se tornado o foco da sociedade, buscando alternativas que ajudem o planeta, principalmente relacionados à mobilidade urbana. Dessa maneira e com o avanço da tecnologia o mercado de veículos elétricos e híbridos vem aumentando nos últimos anos. Neste aspecto, a indústria automobilística investe de maneira pesada em desenvolvimento de tecnologias que façam com que a autonomia dos veículos elétricos aumente (IZO, 2018).

Para que tal objetivo seja alcançado, é necessário que o veículo possua baterias de alta densidade energética por massa, técnicas de controle de alto desempenho, um sistema de gerenciamento de energia eficaz e um motor que possua alta resposta dinâmica, alta eficiência e uma maior relação torque-peso, características essas que compõem um motor BLDC (ECKERT et al., 2018). Assim, o estudo de técnicas de controle voltadas para o controle do motor BLDC e o gerenciamento de outros sistemas do veículo são cada vez mais importantes.

1.2 OBJETIVOS

1.2.1 Objetivos gerais

Realizar a comparação entre diferentes técnicas de controle, como o PID, o híbrido *Fuzzy*-PID e o híbrido *Fuzzy*-PID otimizado pelo PSO (otimização por enxame de partículas), aplicadas ao controle de velocidade de um motor BLDC.

1.2.2 Objetivos específicos

- Realizar uma revisão da literatura, denotando os principais pontos assim como as principais características de cada controlador;
- Desenvolver a bancada de testes experimental;
- Realizar as comparações entre o PID e o Fuzzy-PID em simulações realizadas no software *S*imulink;
- Desenvolver um controlador PID e um controlador híbrido *Fuzzy*-PID e implementá-lo no microcontrolador ESP32;
- Otimizar o controlador híbrido *Fuzzy*-PID por meio do PSO, realizar a simulação no *Simulink* e implementá-lo no microcontrolador ESP32;
- Comparar os resultados obtidos na implementação prática com os da simulação.

1.3 ESTRUTURA DO TRABALHO

Este trabalho está dividido da seguinte forma:

No Capítulo 2 são descritas as estruturas do motor CC e motor BLDC com seus respectivos funcionamentos e diferenças, os fundamentos básicos de controle e o detalhamento do controlador PID, *Fuzzy* , controlador híbrido *Fuzzy*-PID e o algoritmo de otimização por enxame de partículas.

No Capítulo 3 são discutidas as justificativas para a escolha dos componentes da bancada de testes, a metodologia de projeto de cada técnica de controle utilizada, o método de otimização empregado e, por fim, a realização das simulações com auxílio do *software MatLab* e do *software Simulink*.

O Capítulo 4 aborda a comparação das simulações de cada controlador proposto e também apresenta os resultados obtidos na bancada de testes, onde foram realizados testes com carga e sem carga e a comparação entre os resultados obtidos na simulação e no experimental.

No Capítulo 5 apresentam-se as conclusões obtidas com o desenvolvimento desse trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 ESTRUTURA DO MOTOR CC

De acordo com Toro (1994) o motor de corrente contínua (CC) é constituído por quatro elementos: o estator, o rotor, anel comutador e escovas. O estator é constituído de um material ferromagnético envolto por um conjunto de espiras onde uma força magnetomotriz é produzida, já o rotor, cujo o objetivo é possibilitar a passagem do fluxo magnético produzido pelo estator, é um eletroímã onde a ação motora mecânica ocorre no qual é constituído por um núcleo, geralmente de aço-silício, envolto por um conjunto de bobinas ligadas ao anel comutador, que por sua vez transmite corrente elétrica no momento em que seus terminais entram em contato com as escovas (KOSOW, 1993). Nesse processo de comutação, faíscas são geradas entre o anel comutador e as escovas, além de um desgaste em ambos componentes gerando assim a necessidade de uma manutenção frequente (TORO, 1994).

2.2 ESTRUTURA DO MOTOR BLDC

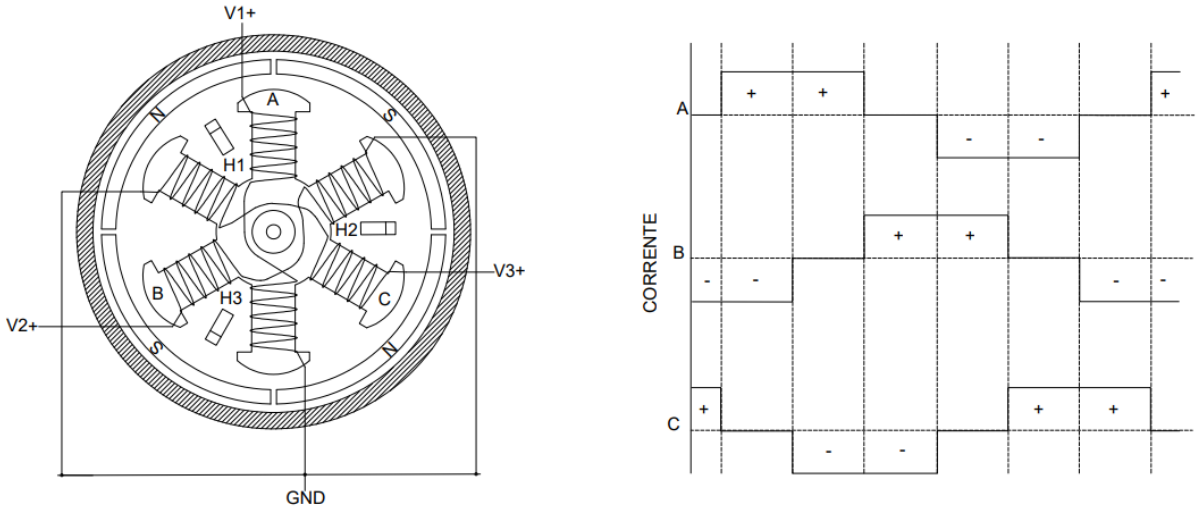
O motor de corrente contínua sem escovas (BLDC) é basicamente composto por um estator trifásico e um rotor com ímãs permanentes em sua superfície, cujo princípio de funcionamento é semelhante a de um motor de corrente alternada síncrono, porém a sua fonte de alimentação é de corrente contínua, (EL-SAMAHY; SHAMSELDIN, 2018).

Uma das alternativas de se aumentar a eficiência do motor é ligar em série duas bobinas opostas duplicando a força de atração e de repulsão, desta maneira, para que o rotor consiga realizar um giro completo é necessário seis intervalos de acordo com a Figura 1.

Nota-se que no mesmo instante a corrente é positiva no ponto A, é negativa no Ponto B e é nula no ponto C, desta maneira é possível utilizar a mesma corrente para energizar duas fases diferentes ao mesmo tempo, realizando uma ligação estrela.

Conforme dito anteriormente são necessários no mínimo seis intervalos para que o motor realize uma volta completa, para que tal fenômeno ocorra umas das so-

Figura 1 – Motor BLDC



Fonte: Autoria própria.

luções seria a utilização de um *Electronic Speed Controller* (ESC), controlador de velocidade eletrônico, no qual utiliza um arranjo de transistores que trabalham de modo sincronizado, acionando as bobinas que devem atrair e repelir o rotor de acordo com sua posição, que pode ser obtida pelo uso de sensores Hall ou o uso da força contra-eletromotriz (NARMADA; AROUNASSALAME, 2014).

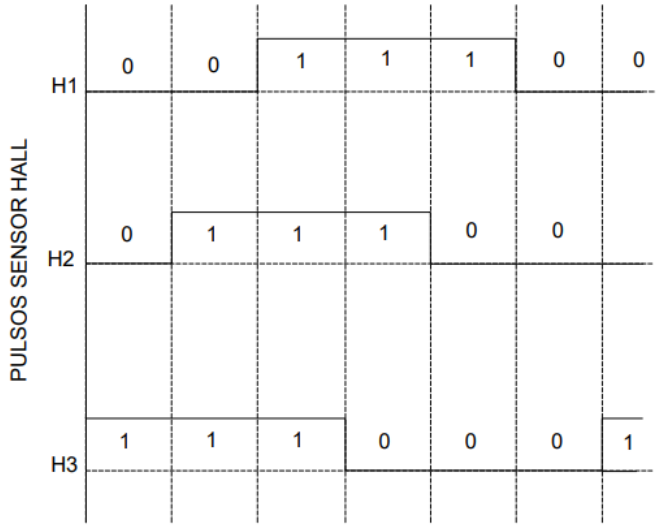
No método que utiliza os sensores Hall para a localização do rotor, os sensores são geralmente arranjados a cada 120° ou a cada 60°, para que no instante em que o campo magnético oriundo do rotor se aproxima de um sensor o mesmo emita um sinal de nível lógico alto para um polo e um sinal de nível lógico baixo para o polo oposto conforme a Figura 2.

Por meio da força contra eletromotriz uma corrente no sentido oposto é gerada nas bobinas que não estão energizadas em determinado instante e como consequência uma tensão induzida é gerada, que por sua vez é identificada pelo controlador presente no ESC no qual realiza cálculos para prever em quais bobinas ele deve ligar ou desligar, conforme a Figura 3.

2.3 MOTOR CC X MOTOR BLDC

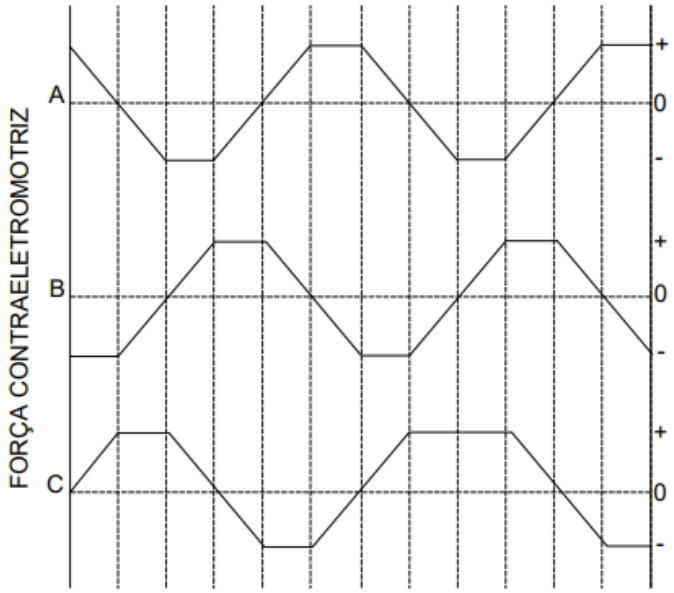
O motor de corrente contínua (CC) por possuir escovas possui efeitos indesejáveis como a projeção de faíscas e partículas de carbono provenientes da mesma, bem como a geração de ruído acústico (VARGHESE; ROY; THIRUNAVUKKARASU,

Figura 2 – Pulsos dos sensores Hall presentes no motor BLDC.



Fonte: Autoria própria.

Figura 3 – Sinal força contraeletromotriz



Fonte: Autoria própria.

2014).

Apesar de sua confiabilidade limitada ligada à necessidade de uma manutenção constante de suas escovas devido ao desgaste de operação e a necessidade de comutadores, o motor CC ainda sim possui suas vantagens tais como: boa eficácia e comportamento linear, descartando a necessidade do uso de técnicas de controle complexas (ARIS et al., 2016).

Diferente dos motores CC, os motores BLDC possuem muitas vantagens, tais

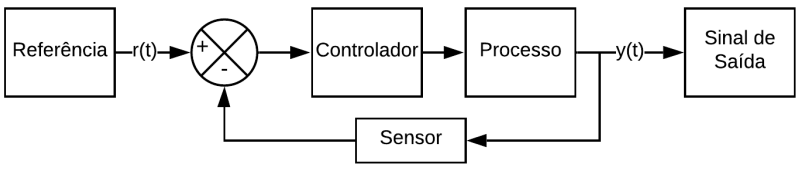
como: vida operacional longa, alta resposta dinâmica, alta eficiência, melhor velocidade vs características de torque, maior faixa de velocidade e maior relação torque-peso, (EL-SAMAHY; SHAMSELDIN, 2018).

Desta maneira, o motor BLDC exige um número reduzido de manutenção e consegue manter a mesma potência de um motor CC ocupando um menor volume. Estas e outras características anteriormente citadas fazem com que a substituição dos motores CC pelos motores BLDC seja mais atrativa e empregada em diversas aplicações industriais (KUMPANYA; THAIPARNAT; PUANGDOWNREONG, 2015).

2.4 CONTROLADOR PID

Os controladores são subsistemas que atuam sobre determinada planta para que resultados pré-estabelecidos possam ser atingidos. Quando dimensionados de maneira correta, os controladores, podem aumentar a eficiência do sistema no qual atua e quando utilizados em malha fechada (Figura 4), onde ocorre a realimentação do sistema com o sinal de saída, a redução do erro da saída em relação ao sinal de entrada pode ocorrer (OGATA, 2011).

Figura 4 – Sistema em malha fechada.



Fonte: Autoria própria.

Por ser simples, robusto e possuir poucos parâmetros de ajuste, 90% das malhas industriais aplicam os controladores PID (CHOPRA; SINGLA; DEWAN, 2014).

Conforme (NISE; SILVA, 2002), o sinal de controle fornecido pelo controlador PID depende de três parâmetros, no qual é dado pela Equação 1:

$$u(t) \quad = \quad K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt} \tag{1}$$

Onde:

$u(t)$ representa o sinal de controle;

$e(t)$ representa o sinal de erro que é a diferença entre o sinal de referência $r(t)$ e a saída do sistema $y(t)$;

K_p representa o ganho proporcional;

K_i representa o ganho integral;

K_d representa o ganho derivativo.

Cada parâmetro, K_p, K_d e K_i, permite alterar o comportamento do controlador aplicado à planta, o uso de altos ganhos podem fazer com que o controlador possa atuar com mudanças rápidas no sinal de saída, em contrapartida baixos ganhos resultam em um controlador com uma característica mais passiva, obtendo pouca influência sobre o sistema (OGATA, 2011). O conhecimento da influência de cada parâmetro no desempenho do sistema, como por exemplo o Percentual de Sobressinal (PSS) que representa quanto o valor máximo de pico ultrapassa o valor final, é importante para que quando for necessário efetuar algum ajuste mais preciso, o mesmo possa operar da maneira esperada. Desta maneira, de acordo com o Quadro 1, pode-se verificar o efeito causado na planta em relação à variação do ganho de cada parâmetro do controlador (KUMAR; SWAIN; NEOGI, 2017).

Quadro 1 – Relação Controlador x Planta (Modificado).

Parâmetro	**Tempo de Resposta**	**PSS**	**Erro**
Proporcional (Kp)	Pouco aumento	Grande	Pequeno
Integral (Ki)	Diminui	Aumenta	Zero
Derivativo (Kd)	Aumenta	Diminui	Pouco aumento

Fonte: Kumar, Swain e Neogi (2017)

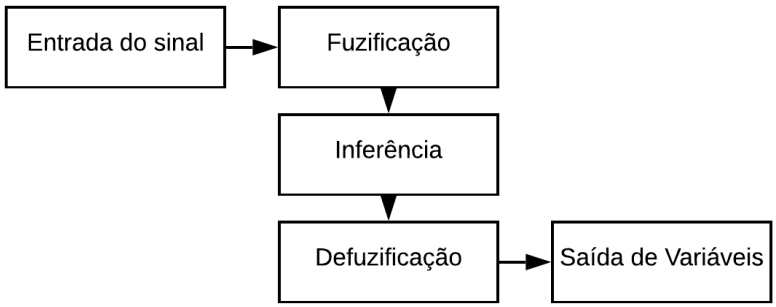
2.5 LÓGICA *FUZZY*

Com o propósito de auxiliar na busca da solução de problemas específicos, a lógica difusa ou comumente conhecida como lógica *Fuzzy* procura se aproximar do pensamento humano, saindo da lógica booleana e buscando respostas lidando com o conceito de verdade parcial. Isto ocorre, pois a base dos sistemas Fuzzy é a teoria dos conjuntos Fuzzy no qual seus elementos possuem um grau de pertinência associado que é determinado pela análise das funções de pertinência (AZEVEDO; BRASIL; OLIVEIRA, 2000).

Uma das principais vantagens de se utilizar o controlador *Fuzzy* é a de que o projetista pode realizar o controle de um sistema com base no comportamento que o mesmo possui, sem a necessidade de se levantar o seu modelo matemático (FENG et al., 2002). Porém, para que isso ocorra é necessário que o projetista tenha um pleno conhecimento do funcionamento do sistema de maneira que consiga ajustar os parâmetros de forma correta fazendo com que o esforço de controle trate de maneira específica cada entrada com o propósito de atingir os objetivos esperados.

Para que ocorra o processamento das variáveis numéricas emitidas ao controlador, por exemplo um sinal enviado por um sensor, é necessária a execução de um processo que consiste em transformar estes valores numéricos em variáveis linguísticas para realizar a tomada de decisão com base em regras pré-estabelecidas, regras estas que estão associadas a um valor numérico necessário para efetuar o controle da planta. Estas etapas podem ser definidas como: fuzzificação, inferência e defuzzificação (CHOI et al., 2005), conforme representado Figura 5.

Figura 5 – Estrutura de um controlador de lógica *Fuzzy*.



Fonte: Autoria própria.

A etapa de fuzzificação consiste em transformar os dados de entrada, que são variáveis numéricas, em variáveis linguísticas, onde ocorre um pré-processamento de categorias com a finalidade de reduzir o número de processos. Já na inferência ocorrem as decisões com base no condicional Se-Então, em inglês *If-Then*, definidas por uma base de regras previamente estabelecidas definido as ações a serem tomadas em determina ocasião. A última etapa do processamento do sinal, a defuzzificação, assegura a interpretação exata das variáveis linguísticas obtidas na fase da inferência em valores numéricos (CHOI et al., 2005).

Existem dois modelos que usualmente são utilizados nos sistemas *Fuzzy*, os clássicos e os de interpolação. Os clássicos possuem para cada regra um termo ne-

buloso dentro de um conjunto fixo de termos convexos que podem ser representados graficamente por funções triangulares, trapezoidais e funções de sino. Dentre os modelos mais comuns vale destacar o modelo de Larsen e o modelo de Mamdani, (SILVA; DATTA; BHATTACHARYYA, 2002).

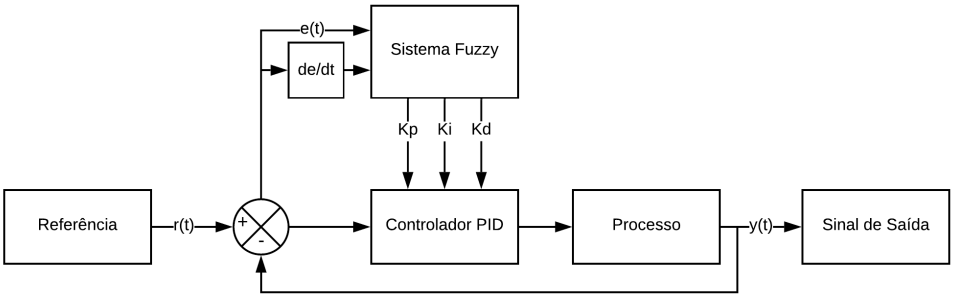
Já os modelos de interpolação apresentam normalmente uma conclusão diferente para cada regra através de uma função estreitamente monotônica, sendo os mais comuns o modelo Takagi-Sugeno e o modelo Tsukamoto. Nesses modelos os valores obtidos por cada regra para cada variável de controle são únicos, onde uma ação de controle global é obtida por meio de uma média ponderada dos valores individuais obtidos, (FERNANDES et al., 2005).

2.6 CONTROLADOR HÍBRIDO *FUZZY*-PID

Apesar de ser robusto e de fácil aplicação o controlador PID possui algumas limitações em determinadas aplicações, como por exemplo em sistemas que não se comportam de maneira linear ou em situações em que a dinâmica da planta varie constantemente, tais situações podem impactar no tempo de resposta do controlador. Uma das alternativas para tal situação seria a união do controlador PID e a lógica *Fuzzy*.

Neste controlador, a aplicação da lógica *Fuzzy* é acoplada a um controlador PID para ajustar os seus parâmetros automaticamente em um processo on-line (Figura 6), ou seja, se houver modificações na dinâmica da planta, como por exemplo variações de carga, os ganhos do controlador PID são ajustados por meio da lógica *Fuzzy* para se adaptarem à essa mudança em tempo real (GOSWAMI; JOSHI, 2018).

Figura 6 – Estrutura básica de um controlador híbrido *Fuzzy*-PID.



Fonte: Autoria própria.

Esta união pode tornar o sistema estável mais rapidamente que um controla-

dor PID clássico, resultando uma diminuição do tempo de acomodação para se atingir o estado estacionário (GEETHA; THANGAVEL, 2016).

2.7 OTIMIZAÇÃO POR ENXAME EM PARTÍCULAS

A otimização por enxame de partículas (PSO), foi originalmente proposta em 1995 por James Kennedy e Russell Eberhart, no qual procuraram descrever o comportamento coletivo de grupos de animais de forma matemática, onde é analisado o comportamento individual de cada membro que compõe o grupo e o impacto social que ele tem sobre seus vizinhos (KENNEDY; EBERHART, 1995).

Por ser baseado em modelos biológicos, o algoritmo utiliza regras básicas que podem gerar um comportamento competitivo e/ou cooperativo entre os indivíduos com o propósito de encontrar a melhor solução para dado problema (GARCIA-GONZALO; FERNANDEZ-MARTINEZ, 2012).

Com o decorrer do tempo foi buscado por diversos pesquisadores maneiras de aumentar o desempenho do PSO, dentre as melhorias vale destacar a análise de estabilidade e a compreensão da dinâmica do enxame de partículas no qual se baseia na influência que o grupo tem sobre a partícula.

Esta adaptação cultural pode ser resumida em três princípios: A percepção individual e coletiva da partícula; a comparação entre os indivíduos e a imitação das melhores partículas (EBERHART; SHI; KENNEDY, 2001).

Desta maneira, por ser simples e robusto este método foi aplicado com sucesso em diversas áreas da engenharia com o propósito de encontrar soluções para variados problemas (GARCIA-GONZALO; FERNANDEZ-MARTINEZ, 2012).

2.7.1 Otimização por meio do PSO

Para a otimização de um problema é necessário a inicialização de uma população composta por N indivíduos ou partículas que são distribuídas com posições x_i e velocidades v_i randômicas. Essas partículas são representadas por um vetor cujo a dimensão é o domínio da função *fitness* que é avaliada por cada partícula em cada iteração resultando na alteração da posição e velocidade de cada indivíduo. Onde o número de iterações e a quantidade de indivíduos influencia na quantidade de proces-

samento utilizado e na precisão do resultado obtido.

Portanto, a decisão tomada por um determinado indivíduo esta ligada com o seu desempenho no passado em conjunto com o desempenho de seus vizinhos.

Desta maneira, para uma melhor compreensão pode-se separar o processo de otimização em sete etapas:

1. Gerar população inicial onde cada população possui uma posição x_i que é o valor que a partícula possui no momento e uma velocidade v_i que é o valor que altera a resposta do indivíduo a cada iteração;

2. Calcular a função de *fitness* ou aptidão retornando como resultando o quão distante cada partícula está do objetivo;

3. Achar o (*pBest*) que é a melhor posição até o momento de cada partícula;

4. Achar o (*gBest*) que é a comparação entre as partículas, cujo o objetivo é de encontrar a melhor partícula da população no momento;

5. Atualizar a velocidade de cada partícula, cujo a equação pode ser representada pela Equação 2, onde r_1 e r_2 são valores aleatórios entre 0 e 1 e c_1 e c_2 valores arbitrários entre 0 e 4. Nesta etapa pode-se adicionar uma variável que pode ser determinante na condição de estabilidade do algoritmo, o fator de inércia (ω) que tem como propósito realizar em um momento de execução do algoritmo uma fase exploratória e com o decorrer das iterações diminuir o seu valor chegando na parte de especialização, em outras palavras, procurar encontrar um balanço entre as habilidades locais e globais do PSO;

6. Determinar a posição de cada partícula conforme apresentado na Equação 3;

7. Avaliar cada partícula encontrando o pBest e o gBest.

$$V_i(t+1) \quad = \quad \omega v_i + c_1 r_1 (pBest - x_i) + c_2 r_2 (gBest - x_i) \tag{2}$$

$$x_i(t+1) \quad = \quad x_i(t) + V_i(t+1) \tag{3}$$

Onde:

ω representa a constante de inércia;

v_i representa a velocidade anterior;

c_1 representa o parâmetro cognitivo;

r_1 representa um numérico aleatório;

$pBest$ representa a melhor posição da partícula;

c_2 representa o parâmetro social;

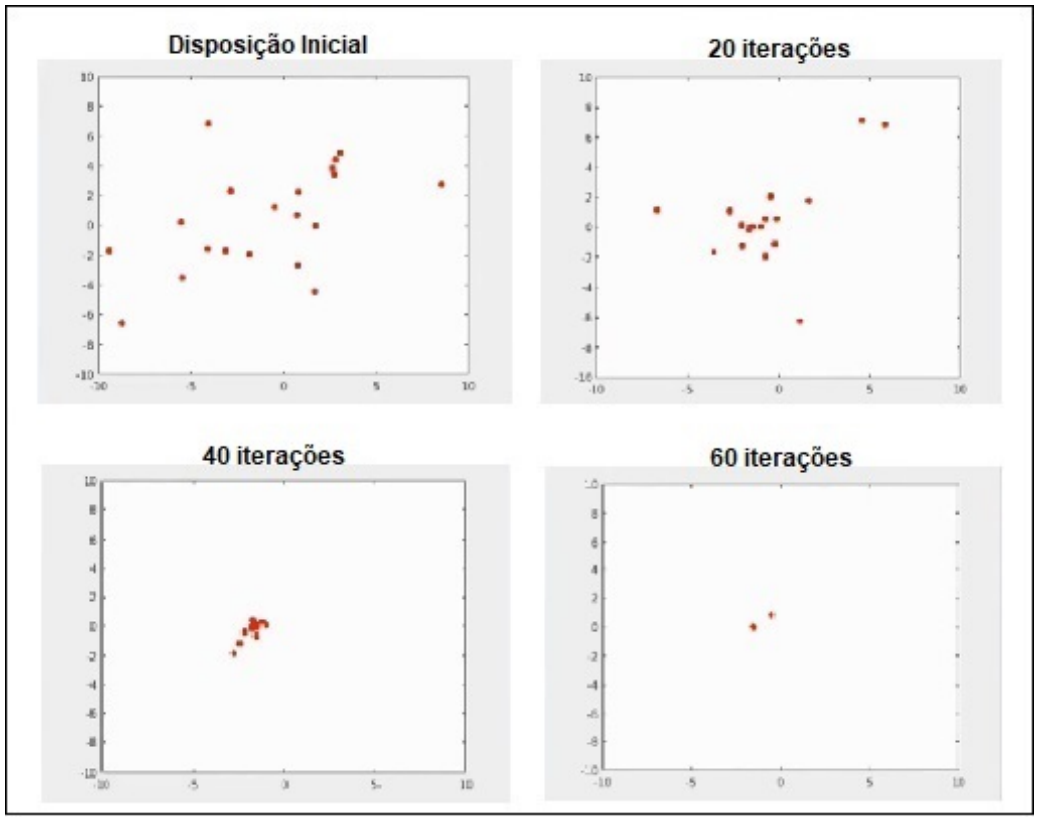
r_2 representa um numérico aleatório;

$gBest$ representa a melhor posição global da população;

x_i representa a posição anterior da partícula.

Para melhor compreensão a Figura 7 ilustra o comportamento das partículas a cada iteração para a resolução de um problema ao qual foram geradas 30 partículas com 60 iterações.

Figura 7 – Exemplo de disposição das partículas do PSO.



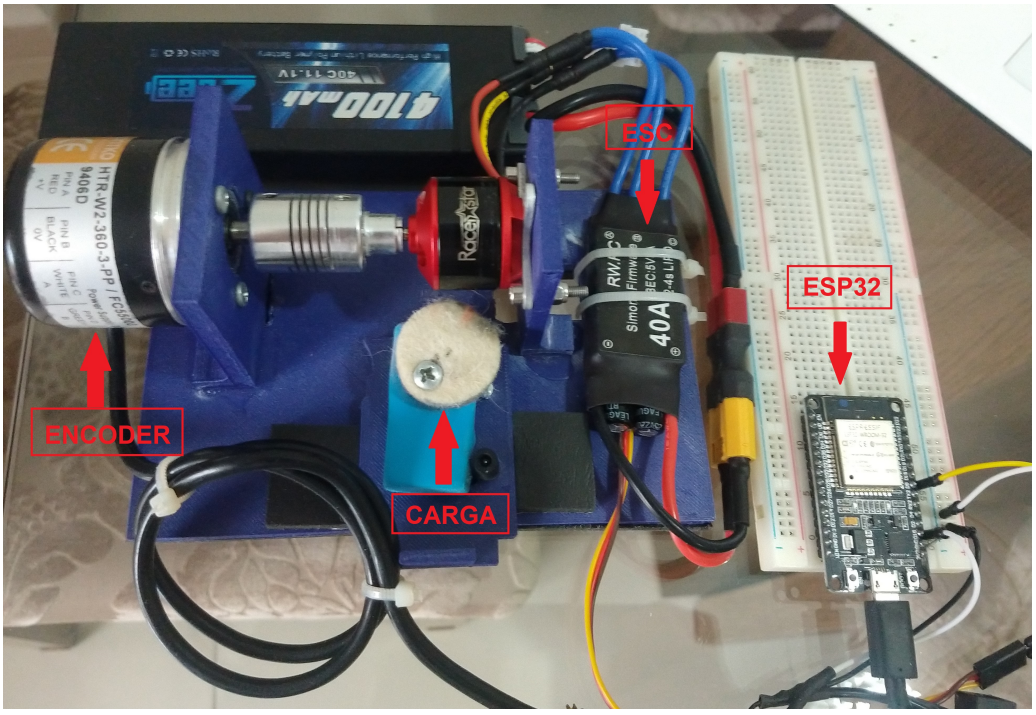
Fonte: Autoria Própia

3 DESENVOLVIMENTO TEÓRICO E EXPERIMENTAL

Para o desenvolvimento do sistema de controle, optou-se em utilizar o motor BLDC Racerstar BR2212 1800KV, um controlador eletrônico de velocidade (ESC) de 40 Ampéres, o *encoder* HTR-W2-360-3PP para fechar a malha de controle e que que emite 360 pulsos por rotação e para efetuar o controle e monitorar esses pulsos foi escolhido o microcontrolador ESP32.

A fim de produzir a variação de carga, necessária nos testes experimentais, foi utilizado um disco de feltro direcionado ao rotor do motor conforme a Figura 8:

Figura 8 – Bancada de Testes.



Fonte: Autoria Própria

3.1 OBTENÇÃO DO MODELO MATEMÁTICO DA PLANTA

Para o desenvolvimento do sistema de controle proposto neste trabalho, o primeiro passo é obter o modelo matemático da planta. Como não é possível conhecer os valores de todos os parâmetros construtivos relacionados ao motor BLDC, torna-se necessária a realização de ensaios experimentais no qual se pode levantar uma representação matemática que se aproxime da realidade no qual o mesmo opera (NISE; SILVA, 2002).

Inicialmente, foram realizados testes em malha aberta, aplicando-se 11,1 volts na entrada do motor afim de se obter a velocidade em regime estacionário e por consequência a quantidade máxima de pulsos emitidos pelo *encoder* no período de amostragem T_s.

Com o propósito de encontrar o período de amostragem T_s mais adequado ao sistema, foram realizadas 5 medidas, onde o motor gira em velocidade máxima a cada trinta segundos para cada T_s sendo eles: 1 s, 500 ms, 275 ms, 200 ms, 100 ms e 50 ms. E destas foi retirada a média aritmética simples com suas respectivas resoluções $R\%$ (Equação 4), conforme a Tabela 1.

$$R\% = 100\frac{1}{Pulsos} \tag{4}$$

Tabela 1 – Máxima contagem de pulsos e resolução por período de amostragem

T_s **(ms)**	**Média de Pulsos/**T_s	**Resolução %**
1000	121000	0,000826
500	60500	0,001653
275	33275	0,003005
200	24200	0,004132
100	12100	0,008264
50	6050	0,016529

Fonte: Autoria Própia.

Com o propósito de validar os dados obtidos e averiguar se a velocidade obtida por meio do *encoder* retrata a realidade, foram realizados os seguintes cálculos:

- Sabendo que a cada volta o *enconder* emite 360 pulsos e que no período T_s de 1000 ms foram obtidos 121000 pulsos tem-se a Equação 5.

$$Voltas = \tfrac{121000}{360} \quad \approx \quad 336 \tag{5}$$

Passando para Rotações Por Minuto (RPM):

$$Velocidade = 336 * 60 segundos \quad \approx \quad 20167 rpm \tag{6}$$

Desta maneira, conforme apresentado pelo fabricante no *datasheet* do motor BLDC Racerstar BR2212 presente na Figura Figura 9 a cada volt aplicado o motor gira 1800 rpm, em outras palavras, na sua tensão nominal de 11,1 volts a velocidade máxima é de aproximadamente 20000 rpm.

Figura 9 – *Datasheet* motor BLDC Racerstar BR2212.

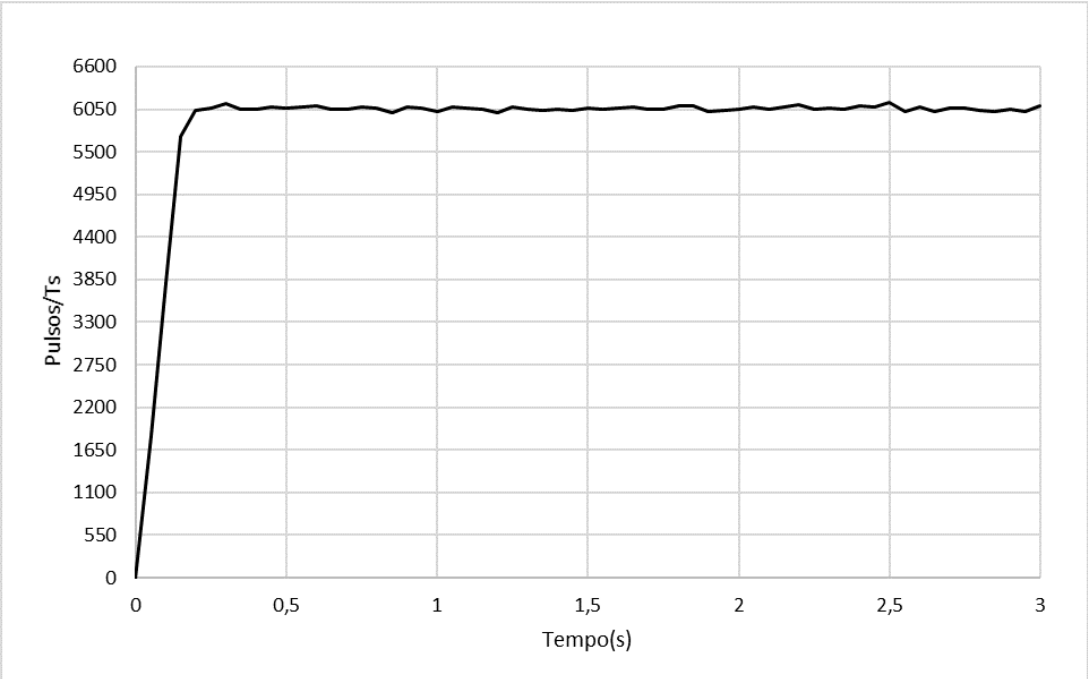
Racerstar BR2212 1800KV 2-4S Brushless Motor For RC Models

Descrição:
Nome da Marca: Racerstar
Nome do Item: BR2212 brushless motor
KV: 1800
Tensão de Operação: 2-4S
Peso: 52g
Hélice Recomendada: 8060

Fonte: Site do Fabricante (Modificado)

Com base nos resultados obtidos experimentalmente (Tabela 1), foi adotado o T_s de 50 ms por ter a melhor resolução e que apresenta valor de 0,016529% e 6050 pulsos por T_s. Resultando no gráfico da Figura 10, onde tem-se a curva de velocidade do motor BLDC em malha aberta, nota-se que a resposta do motor se comporta como a de um sistema de primeira ordem. Assim, os parâmetros necessários a serem determinados são os valores de ganho estático do motor BLDC A e a constante de tempo τ.

Figura 10 – Curva em malha aberta do motor BLDC para um T_s de 50ms



Fonte: Autoria própria

Sistemas de primeira ordem podem ter sua constante de tempo mensurada no momento em que a resposta do sistema leva para atingir 63% de sua velocidade

máxima,(NISE; SILVA, 2002).

Logo,

$$Velocidade = 6050 * 0{,}63 \quad \approx \quad 3812 pulsos \tag{7}$$

Desta maneira, o valor de τ estaria entre 0,05 e 0,1 segundos. Optou-se pelo valor de 0,05 segundos e aplicando os valores conforme apresentado na Equação 8 obteve-se a seguinte curva conforme a Figura 11.

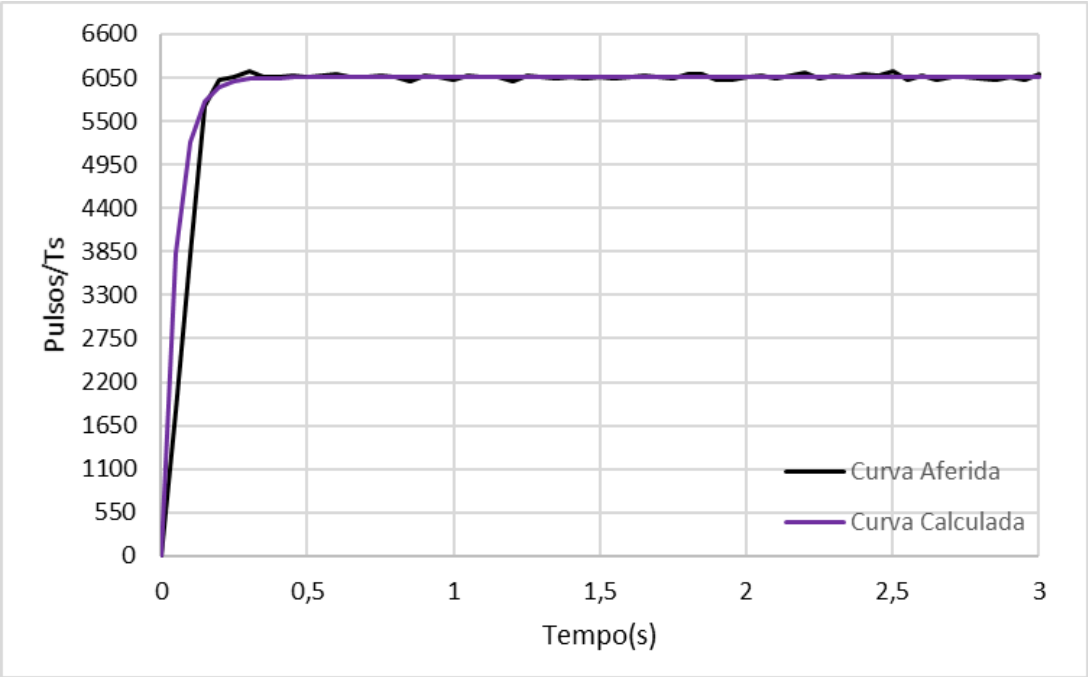
$$y(t) = A(1 - e^{\frac{-t}{\tau}}) \tag{8}$$

Onde: $y(t)$ representa a velocidade do motor no instante t.

Assim,

$$y(t) = 6050(1 - e^{\frac{-t}{0{,}05}}) \tag{9}$$

Figura 11 – Curva calculada e curva aferida da velocidade do motor para um T_s de 50ms



Fonte: Autoria própria

Portanto, o modelo matemático do motor BLDC, ou seja, a função de transferência da planta pode ser dada pela Equação 10.

$$G(s) = \frac{6050}{0{,}05s + 1} \tag{10}$$

E sabendo que o comportamento da planta foi obtido a partir de um degrau de 11,1 V de amplitude, deve-se dividir a Equação 10 pelo degrau, obtendo a Equação 11.

$$G(s) = \frac{6050}{0{,}555s + 11{,}1} \tag{11}$$

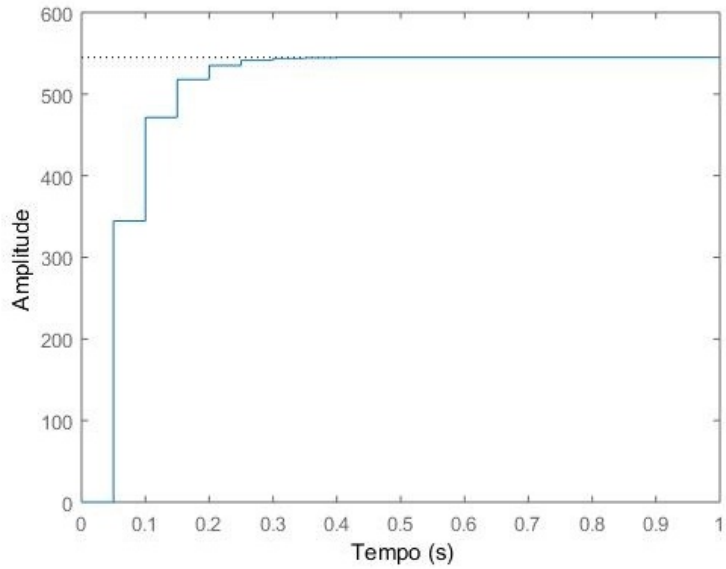
3.2 DISCRETIZAÇÃO DA PLANTA

Como o controle do sistema é embarcado em um microcontrolador no qual é composto por um sistema de controle discreto, torna-se necessário realizar o projeto do controlador no domínio discreto (plano z), onde o modelo é obtido através do método de discretização *Zero-Order Hold* (ZOH), que é o modelo mais adequado para sistemas digitais.

Para que a discretização da planta fosse realizada optou-se pela utilização do *software Matlab* resultando na Equação12, que por sua vez pode ter a sua curva representada conforme a Figura 12.

$$P(z) = \frac{344{,}5}{z - 0{,}3679} \tag{12}$$

Figura 12 – Resposta ao degrau unitário da planta discreta em malha aberta



Fonte: Autoria própria

3.3 PROJETO DOS CONTROLADORES

3.3.1 Projeto do controlador proporcional-integral (PI)

Com o propósito de controlar a velocidade do motor BLDC, optou-se por realizar o projeto de um controlador Proporcional-Integral (PI), sem a necessidade do uso de um ganho derivativo (K_d) por se tratar de um sistema de primeira ordem. Assim com o ganho integral (K_i) é possível corrigir o erro estacionário, e com o ganho proporcional (K_p) obter uma melhoria na velocidade de resposta.

Desta maneira, os parâmetros de desempenho adotados para para o desenvolvimento do controlador foram o percentual de sobressinal (PSS) de no máximo 1,3% e um tempo de estabilização (Te) de 0,5 segundos.

Por meio do percentual de sobressinal e do tempo de estabilização é possível se obter o coeficiente de amortecimento (ζ), conforme representado pela Equação 13.

$$\zeta = \frac{log(\frac{100}{PSS})}{\sqrt{\pi^2 + log(\frac{100}{PSS})^2}} = 0{,}5147 \tag{13}$$

Desta maneira consegue-se efetuar o cálculo da frequência natural (ω_n) de acordo com a Equação 14 e da frequência natural amortecida (ω_d) representada na Equação 15.

$$\omega_n = \frac{4}{Te\zeta} = 15{,}5425 \tag{14}$$

$$\omega_d = \omega_n\sqrt{1-\zeta^2} = 13{,}3255 \tag{15}$$

Após os valores do coeficiente de amortecimento, da frequência natural do sistema e da frequência natural serem obtidos é possível obter o polo no domínio S de acordo com a Equação 16.

$$s = -\zeta\omega_n \pm j\omega_d = -8 + j13{,}3255 \tag{16}$$

Passando a Equação 16 para o domínio Z por intermédio da Equação 17 te-

mos:

$$z = e^{sT_s} = 0{,}5270 + j0{,}4143 \tag{17}$$

Para que o erro estacionário seja nulo foi adotado um polo em z=1 e um zero em α conforme representado na Equação 18, onde k representa o ganho do controlador.

$$K(z) = k\frac{z-\alpha}{z-1} \tag{18}$$

Desta maneira, é possível se obter a fase da planta (θ_p) com a substituição da Equação 17 na Equação 18. E sabendo que a fase do controlador (θ_k) somada a θ_p deve resultar em -π, pode-se obter (θ_k) conforme a Equação 19.

$$\theta_k = -\pi - \theta_p = -1{,}9374 \tag{19}$$

Sabendo que a fase do numerador do controlador (θ_n) é a soma da fase do controlador e a fase do denominador do controlador e que o denominador do controlador é fixo em (z-1), pode-se calcular α de acordo com a Equação 20.

$$\alpha = -\frac{z.tan(\theta_n)}{tan(\theta_n)} = -0{,}2594 \tag{20}$$

Com o valor de α, resta calcular o valor de k para obter o controlador, conforme a Equação 21.

$$k = \frac{1}{|P(z)|.|\frac{z-\alpha}{z-1}|} = 0{,}0009113 \tag{21}$$

Substituindo as Equações 20 e Equações 21 na Equação 18, obtém-se o controlador conforme a Equação 22.

$$K(z) = \frac{0{,}0009113z + 0{,}0002364}{z-1} \tag{22}$$

3.3.2 Projeto do controlador híbrido *Fuzzy*- PI

Conforme já mencionado, o desenvolvimento do sistema *Fuzzy* foi escolhido por ser o método mais simples para fazer o controle adaptativo, porém é empírico

exigindo do projetista ou especialista pleno conhecimento do funcionamento da planta a ser controlada. Dessa maneira, para que o mesmo seja implementado, algumas etapas precisam ser efetuadas, que são a definição das variáveis de entrada e saída, delimitação do universo de discurso de cada variável e a definição da base de regras.

3.3.2.1 Definição das variáveis de entrada e saída

Para estabelecer o sistema de controle *Fuzzy* utilizou-se duas variáveis de entrada. A primeira entrada foi o sinal do erro (E) que é a diferença entre a referência escolhida e o sinal de saída, já a segunda entrada é a derivada do erro (de) que indica a proximidade do sinal de saída com a referência ajudando no ajuste fino do controlador. Para a saída do sistema de controle *Fuzzy* foram adotados os ganhos do controlador PI, denominados K_p e K_i.

3.3.2.2 Delimitação do universo de discurso de cada variável

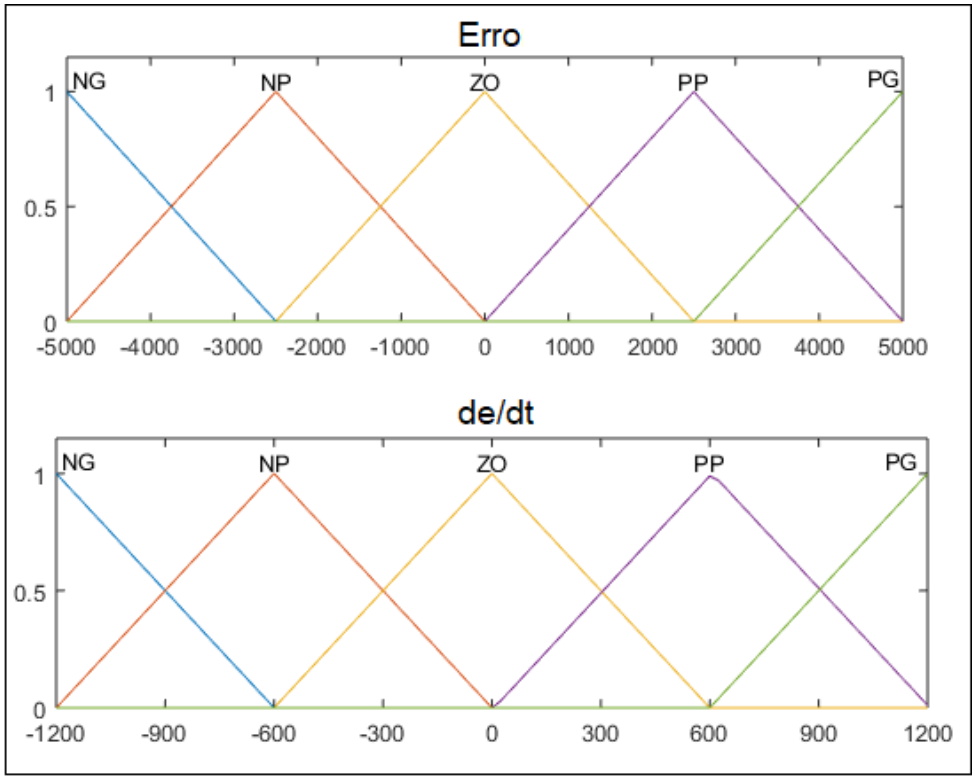
Nessa etapa se inicia o processo de fuzzificação, onde as variáveis numéricas são transformadas em variáveis linguísticas. Assim, por meio de testes em simulações, foram delimitados o universo de discurso para cada variável de entrada e saída, bem como suas funções de pertinência.

Conforme a Figura 13, optou-se para as entradas do erro e a derivada do erro utilizar cinco funções de pertinência do tipo triangular por ser a topologia mais fácil de se manipular e também por ser o tipo de função mais comum em diversas aplicações dos controladores *Fuzzy*. As funções de pertinência são denominadas como NG, NP, ZO, PP e PG significando negativo grande, negativo pequeno, zero, positivo pequeno e positivo grande, respectivamente. Sendo que o universo de discurso da variável E varia de -5000 a 5000 e o universo de discurso da variável de varia de -1200 a 1200.

Para as variáveis de saída foram usadas quatro funções de pertinência do tipo triangular denominadas como Z, P, M e G significando zero, pequeno, médio e grande, respectivamente. Conforme a Figura 14.

Sendo que o universo de discurso da variável K_p é de 0 a 3 e o universo de discurso da variável K_i é de 0 a 7.

Figura 13 – Função de pertinência das entradas (E) e (de).



Fonte: Autoria própria

3.3.2.3 Definição da base de regras

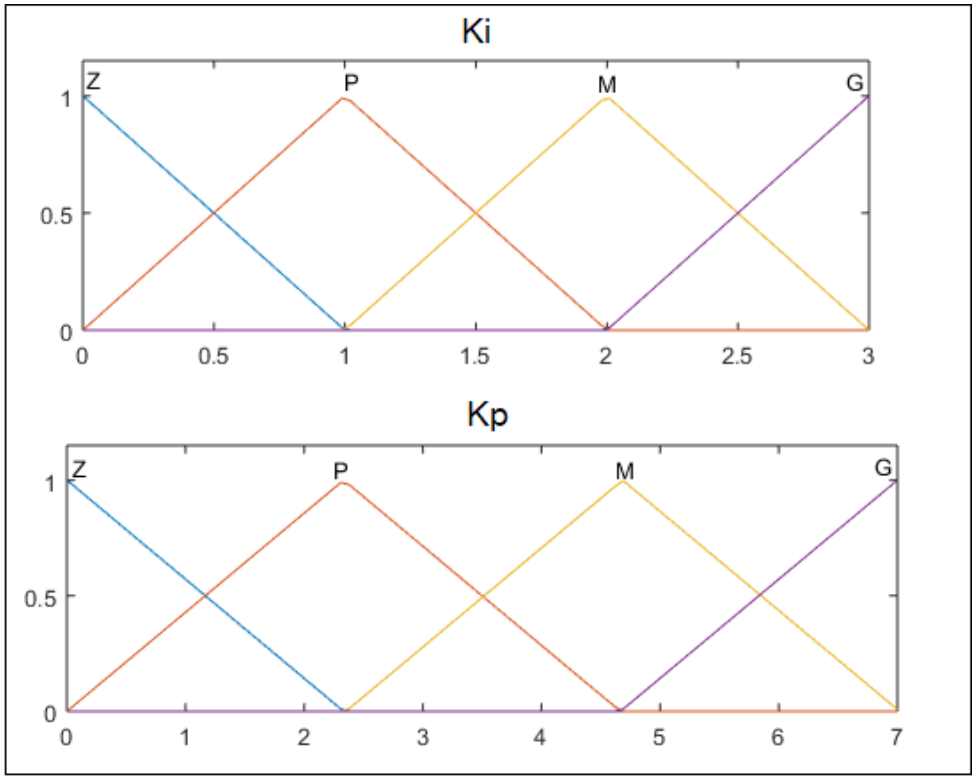
A base de regras tem forte influência sobre o comportamento do sistema de controle influenciando diretamente os parâmetros de desempenho, tais como o tempo de resposta e o percentual de sobressinal.

Para que a base de regras trouxesse o comportamento desejado ao sistema de controle, levou-se em consideração alguns pontos que são:

- Para um valor elevado do erro deve-se ter um ganho proporcional elevado o que ocasiona a correção necessária aproximando-se do valor de referência pré-estabelecido; e um valor moderado do ganho integral para evitar um percentual de sobressinal elevado e não ter um impacto significativo no tempo de resposta;
- Para a situação em que os valores do erro e a derivada são próximos de zero, os valores de ganho proporcional e integral devem ser próximos de zero, mantendo a estabilidade do sistema;

Desta forma, a Figura 15 representam as regras estabelecidas para K_p e K_i respectivamente.

Figura 14 – Função de pertinência das saídas K_p e K_i.



Fonte: Autoria própria

Figura 15 – Base de regras para K_p e K_i.

Kp

E \ de	NG	NP	ZO	PP	PG
NG	G	G	G	G	M
NP	M	G	P	P	P
ZO	M	G	Z	P	G
PP	P	P	P	P	P
PG	M	G	G	M	G

Ki

E \ de	NG	NP	ZO	PP	PG
NG	Z	Z	Z	Z	Z
NP	M	M	M	M	M
ZO	G	G	Z	G	G
PP	P	M	M	M	M
PG	Z	P	G	G	G

Fonte: Autoria própria

3.4 OTIMIZAÇÃO DO CONTROLADOR HÍBRIDO *FUZZY*- PI PELO PSO

Neste trabalho, afim de retirar o empirismo relacionada à lógica *Fuzzy*, realizou-se a otimização do controlador utilizando o PSO, obtendo-se assim um controlador híbrido *Fuzzy*-PI de maneira *offline* onde o PSO é previamente obtido por meio das simulações realizadas no *Simulink*, conforme a Figura 16 para a referência de 2900 pulsos.

Para gerar o PSO de forma adequada os seguintes passos foram seguidos:

1. Gerar a população inicial de 40 partículas com 40 iterações, sendo que cada partícula possui a posição e velocidade geradas de maneira randômica;

2. Calcular a função de *fitness*, em outras palavras, verificar o quão distante as partículas estão do objetivo;
3. Encontrar a melhor posição de cada partícula a cada iteração (*pBest*);
4. Encontrar a melhor partícula da população no momento (*gBest*);
5. Atualizar a velocidade de cada partícula, com base no *pBest* e *gBest* obtidos;
6. Determinar a posição de cada partícula em dado instante;
7. Ao término das 40 iterações, salvar o *gBest* obtido e realizar comparações entre as 35 simulações que serão realizadas e utilizar o melhor resultado entre elas para a implementação.

Vale destacar que no quinto passo é o momento em que a Equação 2 é aplicada, neste caso os valores escolhidos para os parâmetros c_1 e c_2 foram de 1 e 2,5, respectivamente. Dentre os testes realizados utilizou-se um coeficiente de inércia (ω) fixo com o valor de 0,5. Também vale salientar que a nomenclatura das funções de pertinência continuaram as mesmas que utilizadas no controlador *Fuzzy* para melhor comparação.

Com as simulações pode-se notar que o PSO continuou utilizando as 25 regras onde ocorreram diversas alterações conforme a Figura 16, as funções de pertinência continuaram sendo do tipo triangular porém o universo de discurso para cada variável sofreu alterações.

Figura 16 – Base de regras para K_p e K_i obtida pelo PSO.

Kp

E \ de	NG	NP	ZO	PP	PG
NG	G	Z	G	Z	M
NP	G	G	G	Z	G
ZO	P	Z	G	G	M
PP	Z	Z	Z	Z	Z
PG	M	Z	Z	M	Z

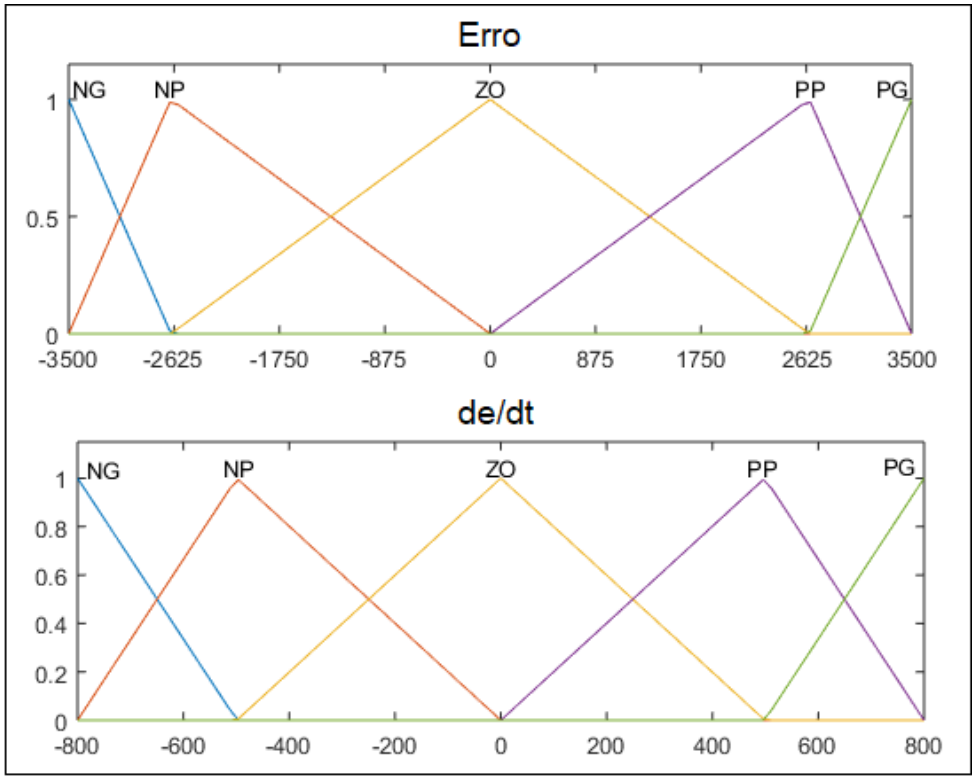
Ki

E \ de	NG	NP	ZO	PP	PG
NG	G	Z	G	Z	G
NP	G	G	Z	Z	G
ZO	G	Z	G	P	G
PP	Z	G	M	Z	G
PG	G	Z	G	Z	G

Fonte: Autoria própria

Agora E varia de -3500 a 3500 e de varia de -800 a 800, de acordo com a Figura 17, enquanto K_p varia de 0 a 3,5 e K_i varia de 0 a 10 conforme a Figura 18.

Figura 17 – Função de pertinência das entradas (E) e (de) obtidas pelo PSO.



Fonte: Autoria própria

3.5 IMPLEMENTAÇÃO DO CONTROLADOR

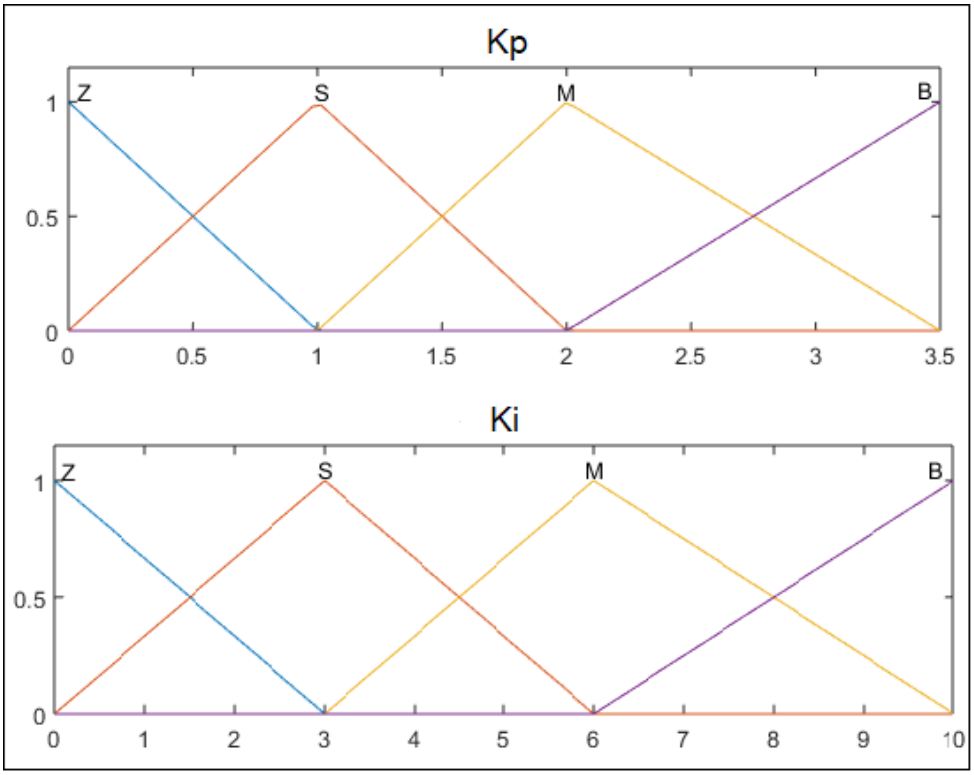
Conforme citado no início deste capítulo, foi utilizado o microcontrolador ESP32 para realizar o controle de velocidade do motor por meio dos sinais obtidos pelo *encoder*. Neste caso, optou-se em utilizar a IDE do Arduino para realizar a programação do controlador devido a familiaridade com a linguagem e a praticidade para o acompanhamento dos resultados obtidos. Desta forma, os seguintes passos foram realizados:

1. Realização da contagem dos pulsos emitidos pelo *encoder* ao ESP32 por meio de interrupções dentro do período estipulado de 50 ms, obtendo-se uma contagem confiável se aproximando da realidade;

2. Realização dos cálculos por meio das equações a diferenças obtidas;

3. Utilização do valor resultante do esforço de controle ($u_{(k)}$) para gerar um pulso *Pulse Width Modulation* (PWM) ao motor necessário naquele instante;

Figura 18 – Função de pertinência das saídas (K_p) e (K_i) obtidas pelo PSO.



Fonte: Autoria própria

3.5.1 Equação a Diferenças

Para que o microcontrolador consiga interpretar os controladores a serem implementados, uma das maneiras de se realizar é escrever o controlador na forma de equação a diferenças.

Assim, a Equação 23 pode ser escrita da seguinte maneira:

$$U(z) * (z-1) = E(z) * 0.0009113 * (z + 0.0002364) \tag{23}$$

Dessa forma, realizando-se a transformada Z inversa da Equação 23 obtém-se $u_{(k)}$ que é descrito pela Equação 24.

$$u_{(k)} = u_{(k-1)} + 0.0009113 * e_{(k)} + 0.0002364 * e_{(k-1)} \tag{24}$$

Onde: $u_{(k-1)}$ representa o esforço de controle passado;

$e_{(k)}$ representa o erro a amostra atual;

$e_{(k-1)}$ representa o erro da amostra passada.

Vale destacar que a Equação 15 representa a equação a diferenças do controlador PI, como o controlador híbrido *Fuzzy*- PI nada mais é do que a multiplicação das

saídas resultantes da lógica *Fuzzy* pelos ganhos do controlador PI, a equação a diferenças utilizada para a implementação deste controlador e do controlador otimizado pelo PSO pode ser dada pela Equação 25.

$$u_{(k)} = u_{(k-1)} + (0.0009113 * e_{(k)} * A) + (0.0002364 * e_{(k-1)} * B) \tag{25}$$

Onde:

A representa o valor da saída K_p obtido por meio da lógica *Fuzzy*;

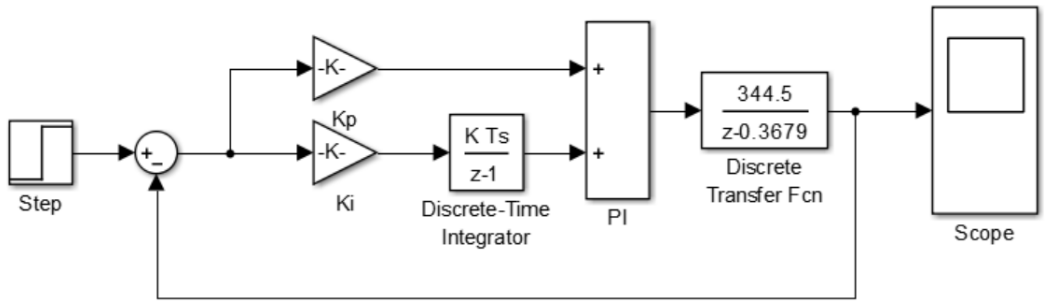
B representa o valor da saída K_i obtido por meio da lógica *Fuzzy*.

Como o pulso PWM a ser enviado ao motor será multiplicado por $u_{(k)}$, realizou-se uma normalização do esforço de controle para que o mesmo sempre varie entre 0 e 1, logo em momentos que $u_{(k)}$ pode saturar, como no primeiro instante onde o motor precisa sair da inércia e $e_{(k)}$ é máximo, o valor que poderia passar de 1 vira 1 não estourando o valor máximo do PWM emitido pelo microcontrolador que é 255 (8 bits).

4 RESULTADOS E DISCUSSÃO

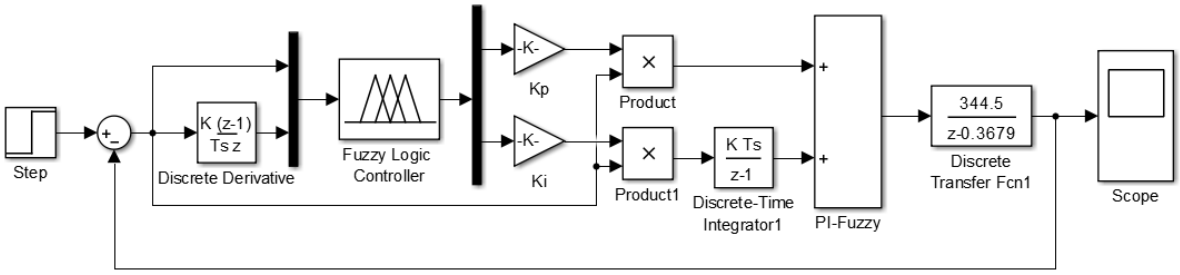
Neste capítulo são apresentados os resultados de simulação e resultados experimentais obtidos neste trabalho. Primeiramente, foram realizadas simulações em malha fechada para uma referência de 2900 pulsos/Ts usando o software *Simulink* para os três controladores abordados neste trabalho e considerando a situações sem carga carga conforme as Figura 19 e Figura 20. Tendo como resultado as curvas representadas na Figura 21.

Figura 19 – Simulação do controlador PI no *Simulink*



Fonte: Autoria própria

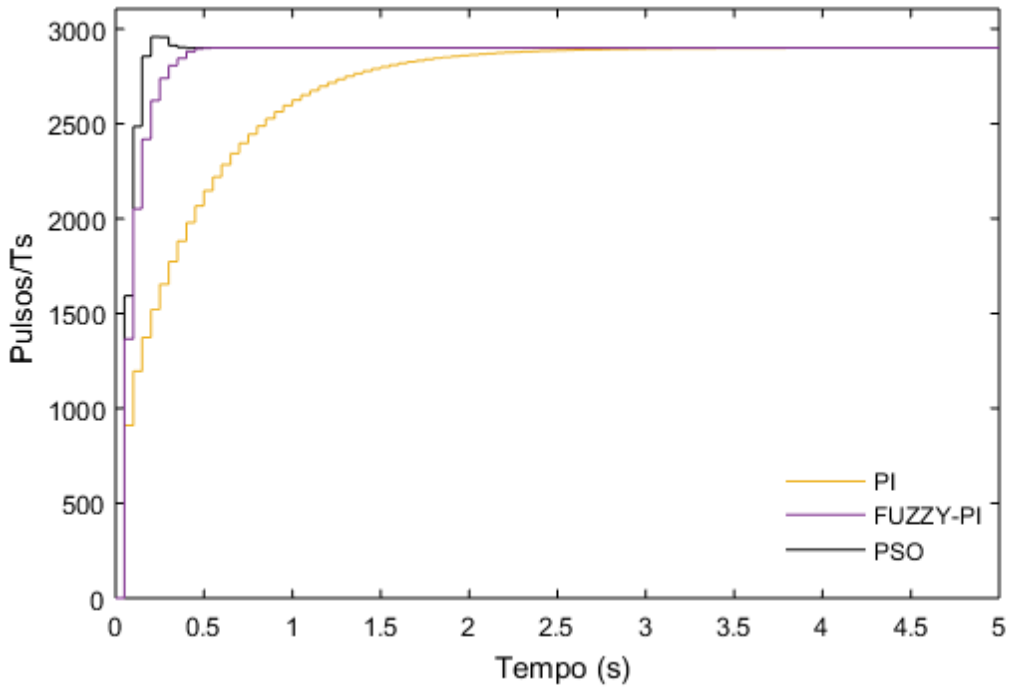
Figura 20 – Simulação do controlador Híbrido *Fuzzy*-PI no *Simulink*



Fonte: Autoria própria

Com base na Tabela 2 consegue-se notar que o controlador híbrido *Fuzzy*-PI otimizado pelo PSO obteve um menor tempo de subida cerca de 0,85 segundos e um tempo de estabilização de 1,63 segundos mais rápido que o controlador PI, porém com um *overshoot* de aproximadamente 2%. Nota-se também um bom desempenho do controlador híbrido *Fuzzy*-PI que teve um desempenho semelhante ao do controlador híbrido *Fuzzy*-PI otimizado pelo PSO com um *overshoot* de 0%.

Figura 21 – Curva dos três controladores simulados sem carga para uma referência de 2900 pulsos



Fonte: Autoria própria

Tabela 2 – Resultados obtidos das curvas de controle simuladas.

Seção	PI	*Fuzzy*-PI	PSO
Tempo de Subida (s)	0,9587	0,1864	0,1078
Tempo de Estabilização (s)	1,7738	0.3463	0,1482
Valor Mínimo	2624	2622	2855
Valor Máximo	2900	2900	2957
Percentual de Sobressinal (%)	0	0	1,9663
Pico	2900	2900	2957
Tempo em que o pico ocorre (s)	10	10	0,2

Fonte: Autoria Própria.

Realizadas as simulações com os três controladores analisados, a próxima etapa apresenta os testes sem carga na bancada experimental, no qual resultou na Figura 22.

Com os dados obtidos consegue-se notar que o controlador PI foi adequadamente projetado obtendo um desempenho melhor que o simulado com um tempo de subida de 0,3 segundos e um percentual de sobressinal semelhante de 1,3%. Outro dado interessante é que na implementação o controlador híbrido *Fuzzy*-PI otimizado pelo PSO não conseguiu se sair melhor que todos os controladores como apresentado na simulação, possuindo um percentual de sobressinal 2% maior e um tempo de subida 0,02 segundos mais lento que o controlador híbrido *Fuzzy*-PI ganhando somente no tempo de estabilização, conforme a Tabela 3.

Tabela 3 – Resultados obtidos das curvas de controle implementadas sem carga.

Seção	PI	*Fuzzy*-PI	PSO
Tempo de Subida (s)	0,3003	0,1180	0,1490
Tempo de Estabilização (s)	0,4532	0,8844	0,7
Valor Mínimo	2698	2765	2802
Valor Máximo	2940	3030	3086
Percentual de Sobressinal (%)	1,3793	4,4828	6,4138
Pico	2940	3030	3086
Tempo em que o pico ocorre (s)	1,35	0,5	0,3

Fonte: Autoria Própria.

Para investigar a robustez de cada técnica de controle utilizada foi inserido um distúrbio de carga onde um disco de feltro é aplicado diretamente ao rotor do motor BLDC e retirado repetidas vezes a cada 5 segundos, em outras palavras, a carga é colocada nos tempos de 5 e 15 segundos e retirada nos tempos de 10 e 20 segundos como pode ser verificado na Figura 23.

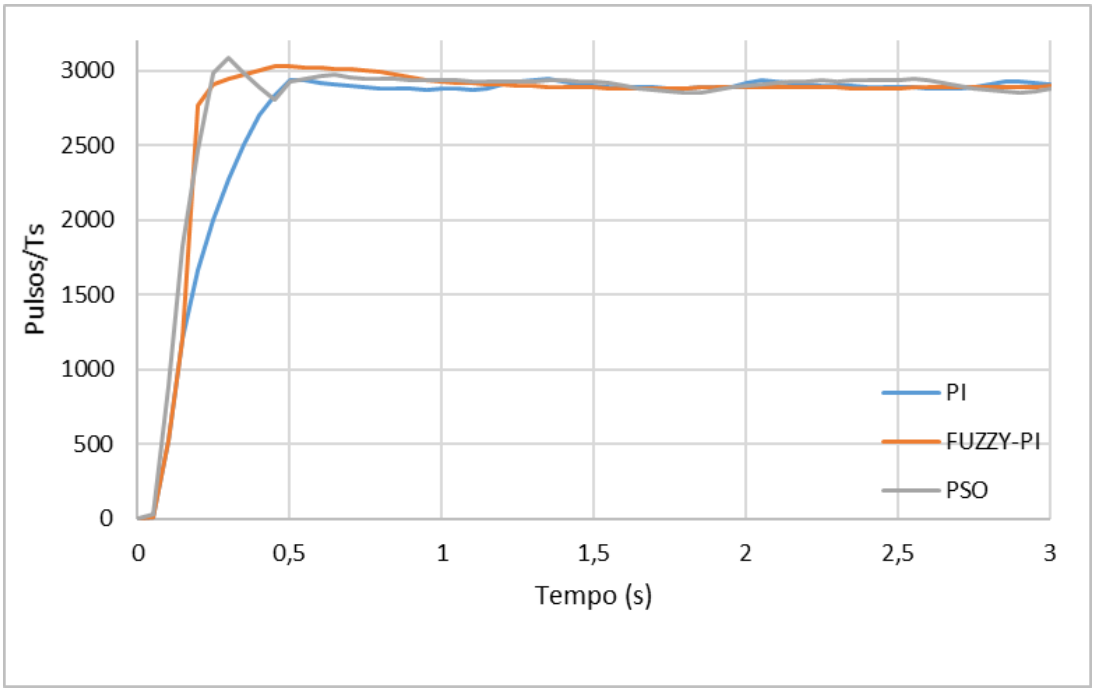
De acordo com o gráfico apresentado e com os dados aferidos presentes na Tabela 4 pode-se notar que o controlador híbrido *Fuzzy*-PI otimizado pelo PSO possui uma resposta mais rápida aos distúrbios de carga conseguindo manter o menor percentual de sobressinal entre os três controladores analisados. Além disso, também vale destacar a robustez do controlador PI que apesar de ter uma resposta lenta para a variação de carga, mesmo assim consegue se adaptar e atingir a referência estabelecida, tendo um tempo de estabilização menor que os outros dois controladores.

Tabela 4 – Resultados obtidos das curvas dos controladores implementados com carga.

Seção	PI	*Fuzzy*-PI	PSO
Tempo de Subida (s)	0,2970	0,1391	0,1512
Tempo de Estabilização (s)	21,8275	24,71	24,4115
Valor Mínimo	984	1131	2057
Valor Máximo	4957	4042	3983
Percentual de Sobressinal (%)	70,9310	39,3793	37,3448
Pico	4957	4042	3983
Tempo em que o pico ocorre (s)	10,35	10,25	10,15

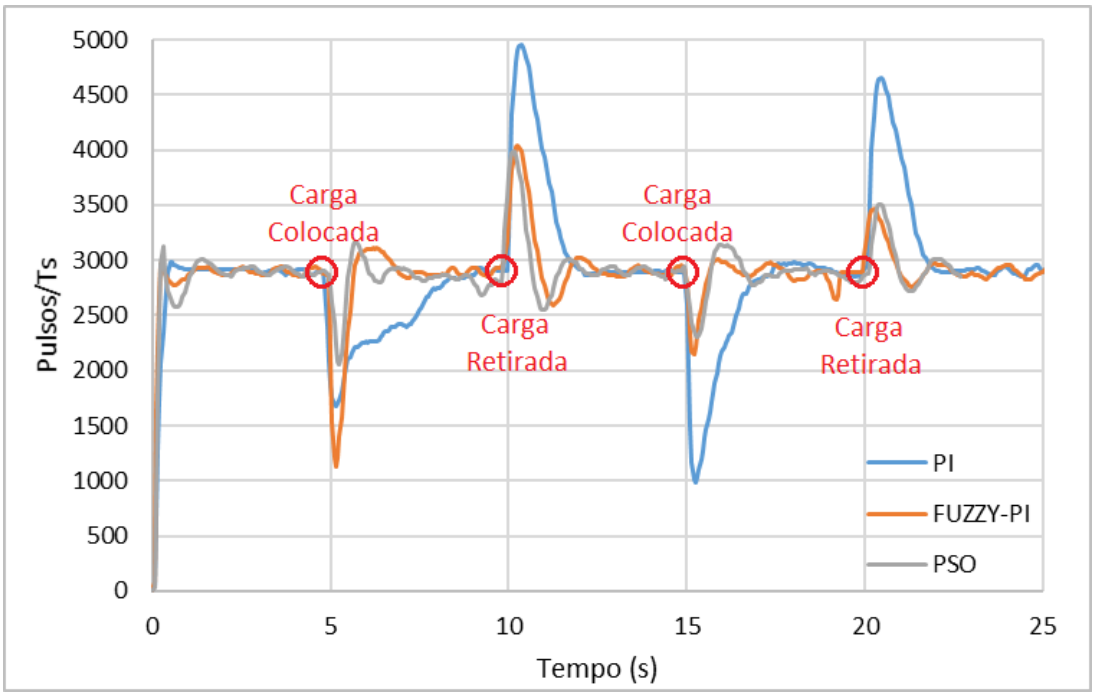
Fonte: Autoria Própria.

Figura 22 – Comparação dos três controladores implementados sem carga para uma referência de 2900 pulsos



Fonte: Autoria própria

Figura 23 – Comportamento dos três controladores implementados com carga para uma referência de 2900 pulsos



Fonte: Autoria própria

5 CONCLUSÕES E PERSPECTIVAS

A necessidade de se otimizar os processos é cada vez mais presente devido ao aumento da complexidade dos sistemas e o avanço da tecnologia, para isso, novas técnicas de controles necessitam ser desenvolvidas e aplicadas. Métodos de controle como o PI tradicional, neste caso específico para o motor BLDC, não conseguem suprir suas necessidades por não se comportar de forma linear em momentos em que há variações de carga. Para isso, um outro método foi empregado, o controlador híbrido *Fuzzy*-PI o qual obteve um melhor desempenho, porém para tal controlador funcionar de forma adequada é necessário que o projetista tenha pleno conhecimento do sistema a ser controlado e por se tratar de um controle completamente empírico os valores utilizados na simulação e implementação podem não ser os ideais.

Desta maneira, o uso do PSO que cria um universo de possíveis resultados e em poucas interações pode conseguir os melhores valores para se projetar os parâmetros necessário para um sistema *Fuzzy*, tais como as funções de pertinência e seus respectivos graus, bem como a organização da base de regras. Desta forma, a aplicação do PSO para o controlador híbrido *Fuzzy*-PI é bem oportuno e conforme resultados apresentados neste trabalho demonstrou um desempenho superior.

Neste trabalho, pôde-se concluir que a união dessas técnicas de controle e de otimização resultaram em um controlador que conseguiu suprir o baixo tempo de resposta do controlador PI para as variações de carga e reduzir a complexidade e eliminar o empirismo de se desenvolver um controlador que utiliza a lógica Fuzzy.

Se tem como proposta para trabalhos futuros a comparação de técnicas de otimização para o controle de velocidade de quatro motores do mesmo modelo utilizado no trabalho em um *drone*, visando analisar outros impactos que os controladores podem ter sobre o sistema, como por exemplo a autonomia da bateria.

REFERÊNCIAS

ARIS, Raja Siti Nur Adiimah Raja et al. Enhancement of variable speed brushless dc motor using neural network. **Indian Journal of Science and Technology**, v. 9, p. 14, 2016. Citado na página 21.

AZEVEDO, Fernando Mendes De; BRASIL, Lourdes Mattos; OLIVEIRA, Roberto Célio Limão de. **Redes neurais com aplicações em controle e em sistemas especialistas**. [S.l.]: Visual Books, 2000. Citado na página 23.

CHOI, Kang-Min et al. Active control for seismic response reduction using modal-fuzzy approach. **International Journal of Solids and Structures**, Elsevier, v. 42, n. 16-17, p. 4779–4794, 2005. Citado na página 24.

CHOPRA, Vikram; SINGLA, Sunil K; DEWAN, Lillie. Comparative analysis of tuning a pid controller using intelligent methods. **Acta Polytechnica Hungarica**, v. 11, n. 8, p. 235–249, 2014. Citado na página 22.

DOPPELBAUER, Martin. The invention of the electric motor 1800-1854. **Philosophical Magazine**, v. 59, 1822. Citado na página 15.

EBERHART, Russell C; SHI, Yuhui; KENNEDY, James. **Swarm intelligence**. [S.l.]: Elsevier, 2001. Citado na página 26.

ECKERT, Jony Javorski et al. Energy storage and control optimization for an electric vehicle. **International Journal of Energy Research**, Wiley Online Library, v. 42, n. 11, p. 3506–3523, 2018. Citado na página 17.

EL-SAMAHY, Adel A; SHAMSELDIN, Mohamed A. Brushless dc motor tracking control using self-tuning fuzzy pid control and model reference adaptive control. **Ain Shams Engineering Journal**, Elsevier, v. 9, n. 3, p. 341–352, 2018. Citado 2 vezes nas páginas 19 e 22.

FENG, G et al. A model reference adaptive control algorithm for fuzzy dynamic systems. In: IEEE. **Proceedings of the 4th World Congress on Intelligent Control and Automation (Cat. No. 02EX527)**. [S.l.], 2002. v. 4, p. 3242–3246. Citado na página 24.

FERNANDES, F. G. et al. Implementação de controladores pid utilizando lógica fuzzy e instrumentação industrial. **In: . São Luis: VII Simpósio Brasileiro de Automação Inteligente**, 2005. Citado na página 25.

FRANKLIN, Gene F; POWELL, J David; EMAMI-NAEINI, Abbas. **Sistemas de controle para engenharia**. [S.l.]: Bookman Editora, 2013. Citado na página 15.

GARCIA-GONZALO, Esperanza; FERNANDEZ-MARTINEZ, Juan Luis. A brief historical review of particle swarm optimization (pso). **Journal of Bioinformatics and Intelligent Control**, American Scientific Publishers, v. 1, n. 1, p. 3–16, 2012. Citado na página 26.

GEETHA, V; THANGAVEL, S. Performance analysis of direct torque controlled bldc motor using fuzzy logic. **International Journal of Power Electronics and Drive Systems**, IAES Institute of Advanced Engineering and Science, v. 7, n. 1, p. 144, 2016. Citado na página 26.

GHANY, MA Abdel; SHAMSELDIN, Mohamed A; GHANY, AM Abdel. A novel fuzzy self tuning technique of single neuron pid controller for brushless dc motor. In: IEEE. **2017 nineteenth international middle east power systems conference (MEPCON)**. [S.l.], 2017. p. 1453–1458. Citado na página 16.

GOSWAMI, Rakesh; JOSHI, Dheeraj. Performance review of fuzzy logic based controllers employed in brushless dc motor. **Procedia computer science**, Elsevier, v. 132, p. 623–631, 2018. Citado 2 vezes nas páginas 16 e 25.

IZO, Alexandre. Rota Mundial de Carros Elétricos Cresce 55% No Primeiro Semestre de 2018. 2018. Disponível em: <https://revistaautoesporte.globo.com/Noticias/noticia/2018/08/frota-mundial-de-carros-eletricos-cresce-55-em-um-ano.html>. Citado na página 17.

KENNEDY, James; EBERHART, Russell. Particle swarm optimization. In: IEEE. **Proceedings of ICNN'95-International Conference on Neural Networks**. [S.l.], 1995. v. 4, p. 1942–1948. Citado na página 26.

KOSOW, I.I. **Maquinas eletricas e transformadores**. [S.l.]: Globo, 1993. ISBN 9788525002303. Citado na página 19.

KUMAR, Brajesh; SWAIN, Subrat Kumar; NEOGI, Nirbhar. Controller design for closed loop speed control of bldc motor. **International Journal on Electrical Engineering and Informatics**, School of Electrical Engineering and Informatics, Bandung Institute of . . . , v. 9, n. 1, p. 146, 2017. Citado 2 vezes nas páginas 15 e 23.

KUMPANYA, Danupon; THAIPARNAT, Sattarpoom; PUANGDOWNREONG, Deacha. Parameter identification of bldc motor model via metaheuristic optimization techniques. **Procedia Manufacturing**, Elsevier, v. 4, p. 322–327, 2015. Citado na página 22.

MATULKA, Rebecca. **The History of the Electric Car.-US Department of ENERGY**. 2014. Citado na página 17.

NARMADA, R; AROUNASSALAME, M. Design and performance evaluation of fractional order controller for brushless dc motor. **International Journal on Electrical Engineering and Informatics**, School of Electrical Engineering and Informatics, Bandung Institute of . . . , v. 6, n. 3, p. 606, 2014. Citado na página 20.

NISE, Norman S; SILVA, Fernando Ribeiro da. **Engenharia de sistemas de controle**. [S.l.]: LTC, 2002. v. 3. Citado 3 vezes nas páginas 22, 29 e 32.

OGATA, K. **Engenharia de controle moderno**. PRENTICE HALL BRASIL, 2011. ISBN 9788576058106. Disponível em: <https://books.google.com.br/books?id=iL3FYgEACAAJ>. Citado 2 vezes nas páginas 22 e 23.

SILVA, G. J.; DATTA, A.; BHATTACHARYYA, S. P. New results on the synthesis of pid controllers. **IEEE Transactions on Automatic Control**, v. 47, n. 2, p. 241–252, 2002. Citado na página 25.

SIMÕES, Marcelo Godoy; SHAW, Ian S. **Controle e modelagem fuzzy**. [S.l.]: Editora Blucher, 2007. Citado na página 16.

TORO, Vincent Del. **Fundamentos de máquinas elétricas**. [S.l.]: Prentice-Hall do Brasil, 1994. Citado na página 19.

VARGHESE, Albert John; ROY, Rejo; THIRUNAVUKKARASU, S. Optimized speed control for bldc motor. **International Journal of Innovative Research in Science, Engineering and Technology**, v. 3, n. S1, p. 1019–30, 2014. Citado na página 21.

ANEXOS

ANEXO A – CÓDIGOS UTILIZADOS PARA SIMULAÇÕES E IMPLEMENTAÇÃO

A.1 CÓDIGO PARA DESENVOLVIMENTO DOS CONTROLADORES

Nesta seção foi utilizado o *Software MatLab* com o propósito de se realizar os cálculos necessários para a obtenção da planta discretizada e o desenvolvimento do controlador PI no qual foi aplicado o método lugar das raízes.

Da mesma maneira, desenvolvido o algoritmo para a otimização por enxame de partículas do controlador híbrido *Fuzzy*-PI.

A.1.1 Controlador PI

```
numP=[6050];
denP=[0.05 1];
P=tf(numP,denP);
P=P/11.1;
Ts=0.05;
Pd=c2d(P,Ts);
step(Pd)
hold on
PSS = 1.3;
qsi = log10(100/PSS)/(sqrt((pi^2)+log10(100/PSS)^2));}
Te= 0.5;
wn = 4/(Te*qsi)
sigma = -qsi*wn
wd = wn*sqrt(1-(qsi$^$2))
s= sigma + j*wd
z = exp(s*Ts)
rp = freqresp (Pd,z)
mp = abs(rp)
tetap = angle$^$s(rp)
tetak = -pi - tetap
tetad=angle(z-1)
tetan=tetak+tetad
tan(tetan);
alpha=(imag(z)-(real(z)*ans))/-ans
mk=abs((z-alpha)/(z-1))
k=1/(mk*mp)
numk=k*[1 -alpha]
denk=[1 -1]
K=tf(numk,denk,Ts)
G=K*Pd;
```

```
T=feedback(G,1);
kp1=alpha*k;
ki1=k-kp1;
step(T);
```

A.1.2 Algoritmo de otimização por enxame de partículas

```
% chama o controlador PI
controlador();
% estabelecimento dos parâmetros de inicialização
NUM_PARTICULAS = 40;
NUM_GERACOES = 35;
nIteracoes = 1;
best_fit = 0;
fit_max=0; vetor_fit_max=0;

% Inicialização dos limites para as funções de Pertinência
ub1 = [0 1.5 2.5 3.5 600 2650 1300 4000 3 6 15];
lb1 = [0 1 2 3 300 2150 800 3500 2.5 5 7];
numRegras=25;
ub=ub1; lb=lb1;
for i=1:numRegras
    % limites para kp
    ub(i+length(ub1)) = 4;
    lb(i+length(lb1)) = 1;
    % limites para ki
    ub(numRegras+i+length(ub1)) = 4;
    lb(numRegras+i+length(lb1)) = 1;
end
[~, numVar] = size(ub);
numFuncoes = 11; %regras SC
numRegras = numVar - numFuncoes;

% Executa o PSO
for n=1:nIteracoes
disp(' ----------------  BPSO   ---------------- ');

% Inicia o enxame com vetor velocidade zero
    swarm = [];
    for i=1:NUM_PARTICULAS
        swarm = [swarm, Particula()];
        for j=1:length(lb)
            swarm(i).vel(j)=0;
        end
    end
```

```
% Gera posição aleatória
    for i=1:length(swarm)
        swarm(i)=swarm(i).inicPosicao(lb,ub,numRegras);
    end

%Inicia a partícula gBest
   gBest = Particula();

  classdef Particula
    properties
        pos = [];
        vel = [];
        fit = 0;
        pbestpos;
        pbest = 0;

    end
    methods
        function self = inicPosicao(self,lb,ub,numRegras)
            for i = 1:length(lb)
     self.pos = [self.pos, rand*(ub(1,i)-lb(1,i))+lb(1,i)];
            end
            for k=1:numRegras
while(self.pos(k)==0 && self.pos(k+numRegras)==0)
        self.pos(k) = randi([lb(1,k),ub(1,k)]);
        self.pos(k+numRegras)=
        randi([lb(1,k+numRegras),ub(1,k+numRegras)]);
                end
            end
        end
        function self = calculaFit(self)
            self.fit = avaliacao(self.pos);
            if self.fit > self.pbest
                self.pbest = self.fit;
                self.pbestpos = self.pos;
            end
        end
        function self = calcVel(self,gBest,i,NUM_GERACOES)
        w=0.5;
        c1=1;
        c2=2.5;
for i=1:length(self.vel)
self.vel(i) = w*self.vel(i) + c1*rand*(self.pbestpos(i)
- self.pos(i)) +
c2*rand*(gBest.pos(i) - self.pos(i));
            end
        end
        function self = calcPos(self,lb,ub,numRegras)
            self.pos = round(self.pos+self.vel);
```

```
%Checagem dos Limites
            contLimites=0;
            for c=1:length(lb)
                if self.pos(c) > ub(c)
                    self.pos(c) = ub(c);
                    contLimites=contLimites+1;
                elseif self.pos(c) < lb(c)
                    self.pos(c) = lb(c);
                    contLimites=contLimites+1;
                end
            end
    fprintf('Bounds: %d out of %d.\n',contLimites,length(lb));
            for k=1:numRegras
    while(self.pos(k)==0 && self.pos(k+numRegras)==0)
    self.pos(k) = randi([lb(1,k),ub(1,k)]);
    self.pos(k+numRegras) = randi([lb(1,k+numRegras),
    ub(1,k+numRegras)]);
                    disp('Regra nula por calcPos');
                end
            end
        end
        function self = mut(self,lb,ub,numRegras)
            self.pos(i) = randi([lb(1,k),ub(1,k)]);
            for k1=1:numRegras
    while(self.pos(k1)==0 && self.pos(k1+numRegras)==0)
    self.pos(k1) = randi([lb(1,k1),ub(1,k1)]);
    self.pos(k1+numRegras) =  randi([lb(1,k1+numRegras),
    ub(1,k1+numRegras)]);
                    disp('Regra nula por mutacao');
                end
            end
        end
    end
  end

%Realização dos Cálculos
    i = 1;
    while i <= NUM_GERACOES

%Calculo fitness
        for j=1:NUM_PARTICULAS
        fprintf('Avaliacao %d de %d\n',j,NUM_PARTICULAS);
            swarm(j)=swarm(j).calculaFit();

% Atualiza o gbest
            if(swarm(j).fit > gBest.fit)
                gBest = swarm(j);
            end
```

```
        end
        hold off
        fprintf('\n');
        %plot - melhor fitness
        vetor_fit_max(i)=gBest.fit;
        subplot(2,2,[1,2])
        plot(vetor_fit_max)
        axis([0 inf 0 max(vetor_fit_max)])

% Busca local da melhor velocidade e posição
        for j=1:NUM_PARTICULAS
            swarm(j)=swarm(j).calcVel(gBest,i,NUM_GERACOES);
        end

        for j=1:NUM_PARTICULAS
            swarm(j)=swarm(j).calcPos(lb,ub,numRegras);
        end

        fprintf('Ger: %d \nMelhor Fit: %f\n\n',i,gBest.fit);
        if gBest.fit == 1
            break
        end
        i=i+1;
    end
    disp('-- FIM --');

    best(n,:) = gBest.pos;
    dispersaoG(1,n) = i;
    dispersaoR(1,n) = gBest.fit;
    if gBest.fit > best_fit
        fit_idx = n;
        vetor_fit_max1 = vetor_fit_max;
        best_fit = gBest.fit;
    end
save('log.mat','best','fit_idx','
dispersaoG','dispersaoR','vetor_fit_max1');
end

% Respostas
load log.mat

%Atualiza o arquivo .fis do controlador Fuzzy
atualizacaoFuzzy(best(fit_idx,:));
   function atualizacaoFuzzy(x1)
   global a

% Configuração do Fuzzy
x2=x1(1,1:11);
a=newfis('fuzzy');
```

```
a=addvar(a,'input','erro',[-x2(8) x2(8)]);
a=addmf(a,'input',1,'NG','trimf',[-x2(8) -x2(8) -x2(6)]);
a=addmf(a,'input',1,'NP','trimf',[-x2(8) -x2(6) x2(1)]);
a=addmf(a,'input',1,'ZO','trimf',[-x2(6) x2(1) x2(6)]);
a=addmf(a,'input',1,'PP','trimf',[x2(1) x2(6) x2(8)]);
a=addmf(a,'input',1,'PG','trimf',[x2(6) x2(8) x2(8)]);
;
a=addvar(a,'input','derro',[-x2(7) x2(7)]);
a=addmf(a,'input',2,'NG','trimf',[-x2(7) -x2(7) -x2(5)]);
a=addmf(a,'input',2,'NP','trimf',[-x2(7) -x2(5) x2(1)]);
a=addmf(a,'input',2,'ZO','trimf',[-x2(5) x2(1) x2(5)]);
a=addmf(a,'input',2,'PP','trimf',[x2(1) x2(5) x2(7)]);
a=addmf(a,'input',2,'PG','trimf',[x2(5) x2(7) x2(7)]);

a=addvar(a,'output','Kp',[x2(1) x2(4)]);
a=addmf(a,'output',1,'Z','trimf',[x2(1) x2(1) x2(2)]);
a=addmf(a,'output',1,'S','trimf',[x2(1) x2(2) x2(3)]);
a=addmf(a,'output',1,'M','trimf',[x2(2) x2(3) x2(4)]);
a=addmf(a,'output',1,'B','trimf',[x2(3) x2(4) x2(4)]);

a=addvar(a,'output','Ki',[x2(1) x2(11)]);
a=addmf(a,'output',2,'Z','trimf',[x2(1) x2(1) x2(9)]);
a=addmf(a,'output',2,'S','trimf',[x2(1) x2(9) x2(10)]);
a=addmf(a,'output',2,'M','trimf',[x2(9) x2(10) x2(11)]);
a=addmf(a,'output',2,'B','trimf',[x2(10) x2(11) x2(11)]);
ruleList=[ ...
1 1 round(x1(12)) round(x1(37)) 1 1
1 2 round(x1(13)) round(x1(38)) 1 1
1 3 round(x1(14)) round(x1(39)) 1 1
1 4 round(x1(15)) round(x1(40)) 1 1
1 5 round(x1(16)) round(x1(41)) 1 1
2 1 round(x1(17)) round(x1(42)) 1 1
2 2 round(x1(18)) round(x1(43)) 1 1
2 3 round(x1(19)) round(x1(44)) 1 1
2 4 round(x1(20)) round(x1(45)) 1 1
2 5 round(x1(21)) round(x1(46)) 1 1
3 1 round(x1(22)) round(x1(47)) 1 1
3 2 round(x1(23)) round(x1(48)) 1 1
3 3 round(x1(24)) round(x1(49)) 1 1
3 4 round(x1(25)) round(x1(50)) 1 1
3 5 round(x1(26)) round(x1(51)) 1 1
4 1 round(x1(27)) round(x1(52)) 1 1
4 2 round(x1(28)) round(x1(53)) 1 1
4 3 round(x1(29)) round(x1(54)) 1 1
4 4 round(x1(30)) round(x1(55)) 1 1
4 5 round(x1(31)) round(x1(56)) 1 1
5 1 round(x1(32)) round(x1(57)) 1 1
5 2 round(x1(33)) round(x1(58)) 1 1
```

```
5 3 round(x1(34)) round(x1(59)) 1 1
5 4 round(x1(35)) round(x1(60)) 1 1
5 5 round(x1(36)) round(x1(61)) 1 1
];
a = addrule(a,ruleList);
writefis(a,'fuzzy');
end

fuzzy fuzzy.fis

%simula o controlador Fuzzy
sim('Controle_Fuzzy_PID_Motor23');
```

A.2 CÓDIGO PARA IMPLEMENTAÇÃO DOS CONTROLADORES NO MICROCONTROLADOR ESP32

Para a implementação do código ao microcontrolador e por familiaridade com a linguagem, foi utilizada a IDE do Arduino. Nesta seção estão os códigos utilizados para todos os controladores presentes neste trabalho.

A.2.1 Controle em malhar aberta

Este código foi utilizado com o propósito de obter a função de transferência do sistema.

```
#include <Wire.h>
hw_timer_t * timer = NULL;
int pulseCount;

void setup() {
  Serial.begin(115200);
  attachInterrupt(digitalPinToInterrupt(15), counter, RISING);
  timer = timerBegin(0, 1, true);
  timerAttachInterrupt(timer, calc, true);
  timerAlarmWrite(timer, 2000000, true);
  timerAlarmEnable(timer);
}

void calc() {
 Serial.println(pulseCount);
 pulseCount = 0;
}
```

```
void counter() {
  pulseCount++;
}
void loop() {}
```

A.2.2 Controlador PID

```
#include <HardwareSerial.h>
#include <ESP32Servo.h>
hw_timer_t * timer = NULL;
int ref = 2900;
int vel,pwmreal;
double uk=0, uk1=0, ek=0, ek1=0, pwm = 0, erro;
Servo ESC;

void setup() {
Serial.begin(115200);
Serial2.begin(115200, SERIAL_8N1, 16, 17);
ESC.attach(18);
delay(5000);
start();
delay(3000);
}

void start(){
  ESC.write(180);
  delay(3000);
  ESC.write(40);
}

void loop() {
   if (Serial2.available()>0){
    vel = Serial2.parseInt();
    }
  erro = ref - vel;
  ek = erro/255;

  Ts=50ms
      uk= uk1+ (0.0009113*ek) + (0.0002364*ek1);
      pwmreal=uk;
    uk1=uk;
    ek1=ek;

  if(uk>1) uk=1;
  if(uk<0) uk=0;

  pwm=180*uk;
```

```
  ESC.write(pwm);
  Serial.println(vel);
  vel = 0;
}
```

A.2.3 Controlador híbrido *Fuzzy*-PID e sua versão otimizada pelo PSO

O código abaixo foi utilizado para a implementação do controlador híbrido *Fuzzy*-PID, para a sua implementação só foram mudadas as regras e os valores das funções de pertinência.

```
#define FIS_TYPE double
#define FIS_RESOLUSION 101
#define FIS_MIN -3.4028235E+6
#define FIS_MAX 3.4028235E+6
typedef FIS_TYPE(*_FIS_MF)(FIS_TYPE, FIS_TYPE*);
typedef FIS_TYPE(*_FIS_ARR_OP)(FIS_TYPE, FIS_TYPE);
typedef FIS_TYPE(*_FIS_ARR)(FIS_TYPE*, double, _FIS_ARR_OP);

#include <ESP32Servo.h>
#include <HardwareSerial.h>

int ref = 2900;
double pwm, vel, Velo;
int vel1=0;
double uk=0, uk1=0, ek=0, ek1=0, ek2=0, kp, ki,KP,KI;
Servo ESC;
double erro = 0;
double derro = 0;
float Ts = 0.05;

FIS_TYPE g_fisInput[2];/// erro aqui
FIS_TYPE g_fisOutput[2];


void setup() {
Serial.begin(115200);
Serial2.begin(115200, SERIAL_8N1, 16, 17);
ESC.attach(18);
delay(3000);
start();
delay(5000);
}

void start(){
ESC.write(180);
```

```
delay(3000);
ESC.write(40);
delay(3000);
}

void fuzzy()
{
    g_fisInput[0] = erro;
    g_fisInput[1] = derro;

    g_fisOutput[0] = 0;
    g_fisOutput[1] = 0;

    fis_evaluate();

    kp = g_fisOutput[0];
    ki = g_fisOutput[1];
}

FIS_TYPE fis_trimf(FIS_TYPE x, FIS_TYPE* p)
{
      FIS_TYPE a = p[0], b = p[1], c = p[2];
      FIS_TYPE t1 = (x - a) / (b - a);
      FIS_TYPE t2 = (c - x) / (c - b);
      if ((a == b) && (b == c)) return (FIS_TYPE) (x == a);
      if (a == b) return (FIS_TYPE) (t2*(b <= x)*(x <= c));
      if (b == c) return (FIS_TYPE) (t1*(a <= x)*(x <= b));
      t1 = min(t1, t2);
      double yy=0;
      return (FIS_TYPE) max(t1, yy);
}

FIS_TYPE fis_min(FIS_TYPE a, FIS_TYPE b)
{
    return min(a, b);
}

FIS_TYPE fis_max(FIS_TYPE a, FIS_TYPE b)
{
    return max(a, b);
}

FIS_TYPE fis_array_operation(FIS_TYPE *array,
int size, _FIS_ARR_OP pfnOp)
{
    int i;
    FIS_TYPE ret = 0;
    if (size == 0) return ret;
    if (size == 1) return array[0];
```

```
    ret = array[0];
    for (i = 1; i < size; i++)
    {
        ret = (*pfnOp)(ret, array[i]);
    }
    return ret;
}
_FIS_MF fis_gMF[] =
{
    fis_trimf
};

int fis_gIMFCount[] = { 5, 5 };
int fis_gOMFCount[] = { 4, 4 };

FIS_TYPE fis_gMFI0Coeff1[] = { -5000, -5000, -2500 };
FIS_TYPE fis_gMFI0Coeff2[] = { -5000, -2500, 0 };
FIS_TYPE fis_gMFI0Coeff3[] = { -2500, 0, 2500 };
FIS_TYPE fis_gMFI0Coeff4[] = { 0, 2500, 5000 };
FIS_TYPE fis_gMFI0Coeff5[] = { 2500, 5000, 5000 };
FIS_TYPE* fis_gMFI0Coeff[] = { fis_gMFI0Coeff1, fis_gMFI0Coeff2,
fis_gMFI0Coeff3, fis_gMFI0Coeff4, fis_gMFI0Coeff5 };

FIS_TYPE fis_gMFI1Coeff1[] = { -1200, -1200, -600 };
FIS_TYPE fis_gMFI1Coeff2[] = { -1200, -600, 0 };
FIS_TYPE fis_gMFI1Coeff3[] = { -600, 0, 600 };
FIS_TYPE fis_gMFI1Coeff4[] = { 0, 600, 1200 };
FIS_TYPE fis_gMFI1Coeff5[] = { 600, 1200, 1200 };
FIS_TYPE* fis_gMFI1Coeff[] = { fis_gMFI1Coeff1, fis_gMFI1Coeff2,
fis_gMFI1Coeff3, fis_gMFI1Coeff4, fis_gMFI1Coeff5 };

FIS_TYPE** fis_gMFICoeff[] = { fis_gMFI0Coeff, fis_gMFI1Coeff };

FIS_TYPE fis_gMFO0Coeff1[] = { 0, 0, 1 };
FIS_TYPE fis_gMFO0Coeff2[] = { 0, 1, 2 };
FIS_TYPE fis_gMFO0Coeff3[] = { 1, 2, 3 };
FIS_TYPE fis_gMFO0Coeff4[] = { 2, 3, 3 };
FIS_TYPE* fis_gMFO0Coeff[] = { fis_gMFO0Coeff1, fis_gMFO0Coeff2,
fis_gMFO0Coeff3, fis_gMFO0Coeff4 };

FIS_TYPE fis_gMFO1Coeff1[] = { 0, 0, 2.4 };
FIS_TYPE fis_gMFO1Coeff2[] = { 0, 2.4, 4.7 };
FIS_TYPE fis_gMFO1Coeff3[] = { 2.4, 4.7, 7 };
FIS_TYPE fis_gMFO1Coeff4[] = { 4.7, 7, 7 };
FIS_TYPE* fis_gMFO1Coeff[] = { fis_gMFO1Coeff1, fis_gMFO1Coeff2,
fis_gMFO1Coeff3, fis_gMFO1Coeff4 };

FIS_TYPE** fis_gMFOCoeff[] = { fis_gMFO0Coeff, fis_gMFO1Coeff };
```

```
int fis_gMFI0[] = { 0, 0, 0, 0, 0 };
int fis_gMFI1[] = { 0, 0, 0, 0, 0 };
int* fis_gMFI[] = { fis_gMFI0, fis_gMFI1};
int fis_gMFO0[] = { 0, 0, 0, 0 };
int fis_gMFO1[] = { 0, 0, 0, 0 };
int* fis_gMFO[] = { fis_gMFO0, fis_gMFO1};
FIS_TYPE fis_gRWeight[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };
int fis_gRType[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };

int fis_gRI0[] = { 1, 1 };
int fis_gRI1[] = { 1, 2 };
int fis_gRI2[] = { 1, 3 };
int fis_gRI3[] = { 1, 4 };
int fis_gRI4[] = { 1, 5 };
int fis_gRI5[] = { 2, 1 };
int fis_gRI6[] = { 2, 2 };
int fis_gRI7[] = { 2, 3 };
int fis_gRI8[] = { 2, 4 };
int fis_gRI9[] = { 2, 5 };
int fis_gRI10[] = { 3, 1 };
int fis_gRI11[] = { 3, 2 };
int fis_gRI12[] = { 3, 3 };
int fis_gRI13[] = { 3, 4 };
int fis_gRI14[] = { 3, 5 };
int fis_gRI15[] = { 4, 1 };
int fis_gRI16[] = { 4, 2 };
int fis_gRI17[] = { 4, 3 };
int fis_gRI18[] = { 4, 4 };
int fis_gRI19[] = { 4, 5 };
int fis_gRI20[] = { 5, 1 };
int fis_gRI21[] = { 5, 2 };
int fis_gRI22[] = { 5, 3 };
int fis_gRI23[] = { 5, 4 };
int fis_gRI24[] = { 5, 5 };
int* fis_gRI[] = { fis_gRI0, fis_gRI1,
fis_gRI2, fis_gRI3, fis_gRI4, fis_gRI5,
fis_gRI6, fis_gRI7, fis_gRI8, fis_gRI9,
fis_gRI10, fis_gRI11, fis_gRI12,
fis_gRI13, fis_gRI14, fis_gRI15,
fis_gRI16, fis_gRI17, fis_gRI18,
fis_gRI19, fis_gRI20, fis_gRI21,
fis_gRI22, fis_gRI23, fis_gRI24 };

int fis_gRO0[] = { 4, 1 };
int fis_gRO1[] = { 4, 1 };
int fis_gRO2[] = { 4, 1 };
int fis_gRO3[] = { 4, 1 };
```

```
int fis_gRO4[] = { 3, 1 };
int fis_gRO5[] = { 3, 3 };
int fis_gRO6[] = { 4, 3 };
int fis_gRO7[] = { 2, 3 };
int fis_gRO8[] = { 2, 3 };
int fis_gRO9[] = { 2, 3 };
int fis_gRO10[] = { 3, 4 };
int fis_gRO11[] = { 4, 4 };
int fis_gRO12[] = { 1, 1 };
int fis_gRO13[] = { 2, 4 };
int fis_gRO14[] = { 4, 4 };
int fis_gRO15[] = { 2, 2 };
int fis_gRO16[] = { 2, 3 };
int fis_gRO17[] = { 2, 3 };
int fis_gRO18[] = { 2, 3 };
int fis_gRO19[] = { 2, 3 };
int fis_gRO20[] = { 3, 1 };
int fis_gRO21[] = { 4, 2 };
int fis_gRO22[] = { 4, 4 };
int fis_gRO23[] = { 3, 4 };
int fis_gRO24[] = { 4, 4 };
int* fis_gRO[] = { fis_gRO0, fis_gRO1,
fis_gRO2, fis_gRO3, fis_gRO4, fis_gRO5,
fis_gRO6, fis_gRO7, fis_gRO8, fis_gRO9,
fis_gRO10, fis_gRO11, fis_gRO12,
fis_gRO13, fis_gRO14, fis_gRO15,
fis_gRO16, fis_gRO17, fis_gRO18,
fis_gRO19, fis_gRO20, fis_gRO21,
fis_gRO22, fis_gRO23, fis_gRO24 };

FIS_TYPE fis_gIMin[] = { -5000, -1200 };
FIS_TYPE fis_gIMax[] = { 5000, 1200 };
FIS_TYPE fis_gOMin[] = { 0, 0 };
FIS_TYPE fis_gOMax[] = { 3, 7 };

FIS_TYPE fis_MF_out(FIS_TYPE** fuzzyRuleSet, FIS_TYPE x, int o)
{
    FIS_TYPE mfOut;
    int r;
    for (r = 0; r < 25; ++r)
    {
        int index = fis_gRO[r][o];
        if (index > 0)
        {
            index = index - 1;
mfOut = (fis_gMF[fis_gMFO[o][index]])(x,fis_gMFOCoeff[o][index]);
        }
        else if (index < 0)
        {
```

```
            index = -index - 1;
mfOut = 1 - (fis_gMF[fis_gMFO[o][index]])
(x,fis_gMFOCoeff[o][index]);
        }
        else
        {
            mfOut = 0;
        }
        fuzzyRuleSet[0][r] = fis_min(mfOut, fuzzyRuleSet[1][r]);
    }
    return fis_array_operation(fuzzyRuleSet[0], 25, fis_max);
}
FIS_TYPE fis_defuzz_centroid(FIS_TYPE** fuzzyRuleSet, int o)
{
FIS_TYPE step = (fis_gOMax[o] - fis_gOMin[o])/ (FIS_RESOLUSION - 1);
    FIS_TYPE area = 0;
    FIS_TYPE momentum = 0;
    FIS_TYPE dist, slice;
    int i;

    for (i = 0; i < FIS_RESOLUSION; ++i){
        dist = fis_gOMin[o] + (step * i);
        slice = step * fis_MF_out(fuzzyRuleSet, dist, o);
        area += slice;
        momentum += slice*dist;
    }

return ((area == 0) ? ((fis_gOMax[o] + fis_gOMin[o])/ 2) :
(momentum / area));}
void fis_evaluate()
{
    FIS_TYPE fuzzyInput0[] = { 0, 0, 0, 0, 0 };
    FIS_TYPE fuzzyInput1[] = { 0, 0, 0, 0, 0 };
    FIS_TYPE* fuzzyInput[2] = { fuzzyInput0, fuzzyInput1, };
    FIS_TYPE fuzzyOutput0[] = { 0, 0, 0, 0 };
    FIS_TYPE fuzzyOutput1[] = { 0, 0, 0, 0 };
    FIS_TYPE* fuzzyOutput[2] = { fuzzyOutput0, fuzzyOutput1, };
    FIS_TYPE fuzzyRules[25] = { 0 };
    FIS_TYPE fuzzyFires[25] = { 0 };
    FIS_TYPE* fuzzyRuleSet[] = { fuzzyRules, fuzzyFires };
    FIS_TYPE sW = 0;
    int i, j, r, o;
    for (i = 0; i < 2; ++i)
    {
        for (j = 0; j < fis_gIMFCount[i]; ++j)
        {
    fuzzyInput[i][j] = (fis_gMF[fis_gMFI[i][j]])
    (g_fisInput[i], fis_gMFICoeff[i][j]);
        }
```

```
    }
    int index = 0;
    for (r = 0; r < 25; ++r)
    {
        if (fis_gRType[r] == 1)
        {
            fuzzyFires[r] = FIS_MAX;
            for (i = 0; i < 2; ++i)
            {
                index = fis_gRI[r][i];
                if (index > 0)
fuzzyFires[r] = fis_min(fuzzyFires[r], fuzzyInput[i]
[index - 1]);
                else if (index < 0)
fuzzyFires[r] = fis_min(fuzzyFires[r], 1 - fuzzyInput[i]
[-index - 1]);
                else
fuzzyFires[r] = fis_min(fuzzyFires[r], 1);
            }
        }
        else
        {
            fuzzyFires[r] = FIS_MIN;
            for (i = 0; i < 2; ++i)
            {
                index = fis_gRI[r][i];
                if (index > 0)
fuzzyFires[r] = fis_max(fuzzyFires[r], fuzzyInput[i]
[index - 1]);
                else if (index < 0)
fuzzyFires[r] = fis_max(fuzzyFires[r], 1 - fuzzyInput[i]
 [-index - 1]);
                else
fuzzyFires[r] = fis_max(fuzzyFires[r], 0);
            }
        }
        fuzzyFires[r] = fis_gRWeight[r] * fuzzyFires[r];
        sW += fuzzyFires[r];
    }
    if (sW == 0)
    {
        for (o = 0; o < 2; ++o)
        {
    g_fisOutput[o] = ((fis_gOMax[o] + fis_gOMin[o]) / 2);
        }
    }
    else
    {
        for (o = 0; o < 2; ++o)
```

```
        {
    g_fisOutput[o] = fis_defuzz_centroid(fuzzyRuleSet, o);
        }
    }
}
void loop() {
    if (Serial2.available()>0){
    vel = Serial2.parseInt();
 }
    erro = ref - vel;
    derro = (erro-ek1)/Ts;

    fuzzy();
    KP=kp*1;
    KI=ki*1;
    ek = erro/255;

    uk= uk1+ ((0.0009113*KP)*ek) + ((0.0002364*KI)*ek1);
    uk1=uk;
    ek1=ek;
    vel1=vel;

  if(uk>1) uk=1;
  if(uk<0) uk=0;

% PWM do ESC 30 valor MIN e 180 valor MAX

  pwm=180*uk;
  ESC.write(pwm);
  Serial.println(vel);
 vel = 0;
}
```