

**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ  
CÂMPUS PATO BRANCO  
ENGENHARIA DE COMPUTAÇÃO**

**CACIANO DANGUI MATTIELLO**

**COMPARATIVO ENTRE CONTROLADOR PID E FUZZY NO  
CONTROLE DE ATITUDE EM UM QUADRICÓPTERO**

**TRABALHO DE CONCLUSÃO DE CURSO**

**PATO BRANCO**

**2014**

**CACIANO DANGUI MATTIELLO**

**COMPARATIVO ENTRE CONTROLADOR PID E FUZZY NO  
CONTROLE DE ATITUDE EM UM QUADRICÓPTERO**

Trabalho de Conclusão do Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do curso de Engenharia de Computação da Universidade Tecnológica Federal do Paraná - Câmpus Pato Branco como requisito parcial para obtenção do título de Engenheiro de Computação.

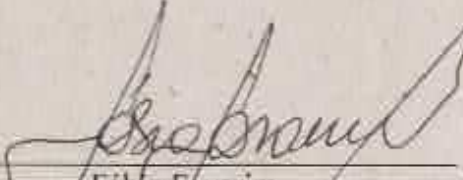
Orientador: Prof. Dr. Fábio Favarim

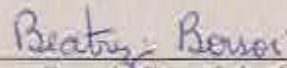
**PATO BRANCO  
2014**

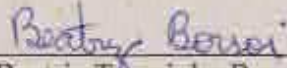


### TERMO DE APROVAÇÃO


Às 13h30 do dia 15 de agosto de 2014, na sala V007, UTFPR, Câmpus Pato Branco, reuniu-se a banca examinadora composta pelos professores Fábio Favarim (orientador), Beatriz Terezinha Borsoi e Kathya Silvia Collazos Linares para avaliar o trabalho de conclusão de curso com o título **Comparativo entre controlador PID e fuzzy no controle de atitude em um quadricóptero**, do aluno **Caciano Danguí Mattiello**, matrícula 674753, do curso de Engenharia de Computação. Após a apresentação o candidato foi arguido pela banca examinadora. Em seguida foi realizada a deliberação pela banca examinadora que considerou o trabalho aprovado.

  
Fábio Favarim  
Orientador (UTFPR)

  
Beatriz Terezinha Borsoi  
(UTFPR)

  
Beatriz Terezinha Borsoi  
Coordenador de TCC

  
Kathya Silvia Collazos Linares  
(UTFPR)

  
Marco Antonio de Castro Barbosa  
Coordenador do Curso de  
Engenharia de Computação

## **AGRADECIMENTOS**

A Joelma Pereira que esteve ao meu lado em todos os momentos difíceis, antes e durante toda a graduação.

Ao Prof. Dr. Fábio Favarim pela orientação, ajuda e prestatividade em todos os momentos da graduação.

As professoras Dra. Beatriz Borsoi e Dra. Kathy Linares pela disposição em participar na banca e sugestões para melhoramento deste trabalho.

Aos demais professores e amigos que estiveram ao longo de toda a graduação, incentivando e apoiando para ser possível a chegada deste momento.

## RESUMO

MATTIELO, Caciano D. Comparativo entre controlador PID e fuzzy no controle de atitude em um quadricóptero. 88 f. Trabalho de Conclusão de Curso – Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Pato Branco, 2014.

Atualmente quadricópteros veem sendo utilizados em praticamente todas as áreas de conhecimento para obtenção de dados por meio aéreo, devido ao seu baixo custo e grande mobilidade. Este trabalho descreve um comparativo entre controladores utilizando a técnica clássica PID e a lógica Fuzzy, aplicados no controle de atitude em um quadricóptero. Foram estudados os controladores e componentes necessários para simulação e elaboração de um protótipo. Todo o protótipo foi modelado matematicamente e efetuada a simulação no ambiente Simulink, no qual os controladores foram implementados e ajustados. Em uma segunda etapa, os controladores foram analisados em ambiente real e confrontados com o simulado. Ao final foram analisadas as diferenças que ocorreram entre o controlador PID e Fuzzy no controle de atitude no ambiente real, no qual o controlador Fuzzy obteve melhores resultados.

**Palavras-chave:** quadricóptero; controle PID; lógica Fuzzy

## ABSTRACT

MATTIELO, Caciano D. Comparative between Fuzzy and PID controllers in attitude control of a quadcopter. 88 f. Final Papers - Computer Engineering, Federal Technological University of Parana. Pato Branco, 2014.

This paper describes a comparative between controllers using the classical technique PID and Fuzzy logic applied at the attitude control on a quadcopter. Were studied the controllers and components necessary for the simulation and the working up of a prototype. The entire prototype was mathematically modeled and simulated at the Simulink environment; in which the controllers were implemented and adjusted. In a second step, the controllers were analyzed in a real environment and confronted with the simulated results. Were analyzed at the end, the differences that occurred between the PID and Fuzzy controllers at the control in the real environment, concluding that the fuzzy controller achieved better results.

**Keywords:** quadricopter; PID control; Fuzzy Logic

## LISTA DE FIGURAS

FIGURA 1	– Bréguet-Richet em 1907 .....	13
FIGURA 2	– Ohemichen No 2 .....	14
FIGURA 3	– Flying Octopus .....	14
FIGURA 4	– AR Drone da Parrot .....	15
FIGURA 5	– Quadricoptero para <i>Aerial Robotic Construction</i> .....	16
FIGURA 6	– Construção realizada com ARC .....	16
FIGURA 7	– Configurações a) <i>inrunner</i> ; b) <i>outrunner</i> .....	17
FIGURA 8	– Relação entre corrente, <i>Back EMF</i> e fase do motor <i>brushless</i> .....	18
FIGURA 9	– Passo de hélice .....	19
FIGURA 10	– Exemplo de acelerômetro do tipo MEMS .....	20
FIGURA 11	– Princípio de catilever em acelerômetros .....	20
FIGURA 12	– Esquema do giroscópio com a força de Coriolis .....	21
FIGURA 13	– Giroscópio em uma pastilha de silício .....	21
FIGURA 14	– Controle com PID no tempo contínuo .....	22
FIGURA 15	– Representação do copo de água na forma de conjuntos com lógica clássica (a) e nebulosa (b). .....	23
FIGURA 16	– Funções de pertinência de conjuntos nebulosos (a) triangular, (b) trapezoidal e (c) gaussiana .....	24
FIGURA 17	– Normalidade (a) subnormal, (b) normal. ....	25
FIGURA 18	– Operações <i>fuzzy</i> : (a) Conjunto A e B; (b) União de A e B; (c) Intersecção de A e B; (d) Complemento de A .....	25
FIGURA 19	– Exemplo de utilização das variáveis linguísticas .....	26
FIGURA 20	– Organização de um sistema fuzzy .....	29
FIGURA 21	– Controle com <i>fuzzy</i> .....	30
FIGURA 22	– Esquema básico para o quadricoptero. ....	31
FIGURA 23	– Frame Talon Turnigy V1. ....	32
FIGURA 24	– Motor Brushless DC. ....	32
FIGURA 25	– Hélice 10x4.5 balanceada. ....	33
FIGURA 26	– ESC de 25A fabricado pela Turnigy. ....	34
FIGURA 27	– Bateria Zippy Compact LiPo 3S 25C 3700mAh. ....	35
FIGURA 28	– Conjunto de sensores 10 DOF. ....	35
FIGURA 29	– Launchpad Tiva com ARM Cortex M4F da Texas. ....	36
FIGURA 30	– Balança de precisão SF-400. ....	37
FIGURA 31	– Estrutura básica do funcionamento de um Drone .....	39
FIGURA 32	– Bloco 3DOF do Simulink. ....	45
FIGURA 33	– Blocos para simulação PID. ....	46
FIGURA 34	– Blocos para simulação Fuzzy. ....	47
FIGURA 35	– Bloco do Misturador. ....	47
FIGURA 36	– Bloco de Forças Atuantes. ....	48
FIGURA 37	– Bloco de Resposta dos Motores. ....	48
FIGURA 38	– Relação entre PWM e Velocidade nos Motores 1, 2, 3 e 4 respectivamente. ....	49
FIGURA 39	– Região linear de cada motor. ....	49
FIGURA 40	– Empuxo dos Motores e Coeficiente de Empuxo. ....	50

FIGURA 41	–	Comparação entre a função de transferência e valores reais. ....	51
FIGURA 42	–	Modelos geométricos para cálculo do momento de inércia . ....	51
FIGURA 43	–	Controle para Velocidade Angular de Rolagem. ....	54
FIGURA 44	–	Controle para Velocidade Angular de Arfagem. ....	54
FIGURA 45	–	Controle para Ângulo de Rolagem. ....	55
FIGURA 46	–	Controle para Ângulo de Arfagem. ....	55
FIGURA 47	–	Distúrbios de 2 Hz no eixo de Rolagem. ....	56
FIGURA 48	–	Distúrbios de 2 Hz no eixo de Arfagem. ....	56
FIGURA 49	–	Teste do controlador fuzzy Rolagem com distúrbios de 2 Hz. ....	57
FIGURA 50	–	Teste do controlador fuzzy de Arfagem com distúrbios de 2 Hz. ....	58
FIGURA 51	–	Resposta do controlador PID em ambiente real. ....	58
FIGURA 52	–	Resposta do controlador Fuzzy em ambiente real. ....	59
FIGURA 53	–	Representação do Sistema Neural ....	64
FIGURA 54	–	Perceptron Rosenblatt (1957) ....	65
FIGURA 55	–	Classificação de pontos ....	65



## LISTA DE TABELAS

TABELA 1	– Especificações do motor Turnigy L2215J .....	33
TABELA 2	– Especificações dos sensores .....	34
TABELA 3	– Principais características do TM4C123G .....	36
TABELA 4	– Movimentos do Drone .....	38
TABELA 5	– Variáveis Linguísticas. ....	57
TABELA 6	– Regras Fuzzy para controle angular. ....	57

## SUMÁRIO

<b>1 INTRODUÇÃO</b>	<b>11</b>
1.1 PROBLEMA DE PESQUISA	11
1.2 JUSTIFICATIVA	12
1.3 OBJETIVOS	12
1.3.1 Objetivo Geral	12
1.3.2 Objetivos Específicos	12
1.4 ORGANIZAÇÃO DO TEXTO	12
<b>2 REVISÃO BIBLIOGRÁFICA</b>	<b>13</b>
2.1 HISTÓRIA DOS QUADRICÓPTEROS	13
2.2 QUADRICÓPTEROS NA ATUALIDADE	14
2.2.1 AR Drone	15
2.2.2 <i>Aerial Robotic Construction- ARC</i>	15
2.3 PRINCIPAIS COMPONENTES DE UM QUADRICÓPTERO	17
2.3.1 Motores <i>Brushless</i> PM	17
2.3.2 Hélices	18
2.3.3 Sensores	19
2.4 MÉTODOS DE CONTROLE E OTIMIZAÇÃO	22
2.4.1 Controlador Proporcional, Integrativo e Derivativo (PID)	22
2.4.2 Lógica <i>Fuzzy</i>	22
2.4.2.1 Funções de Pertinência	23
2.4.2.2 Operadores Fuzzy	24
2.4.2.3 Variáveis Linguísticas	26
2.4.2.4 Regras Fuzzy	27
2.4.2.5 Inferência <i>Fuzzy</i>	27
2.4.2.6 Sistemas Fuzzy	28
2.4.2.7 Controladores Fuzzy	29
<b>3 MATERIAIS E MÉTODOS</b>	<b>31</b>
3.1 MATERIAIS	31
3.1.1 Estrutura	31
3.1.2 Motores	32
3.1.3 Hélices	33
3.1.4 ESCs	33
3.1.5 Bateria	34
3.1.6 Sensores	34
3.1.7 Microcontrolador	36
3.1.8 Balança	37
3.2 MÉTODOS	37
3.2.1 Cinemática	38
3.2.2 Dinâmica	41
<b>4 SIMULAÇÃO</b>	<b>45</b>
4.1 MODELO DINÂMICO PARA A ESTRUTURA	45
4.2 MODELO DINÂMICO PARA OS MOTORES E ESCS	45
4.3 MODELO DE BLOCOS DA SIMULAÇÃO	46
4.3.1 Bloco de Controle PID e Fuzzy	46

4.3.2 Bloco de Mistura .....	46
4.3.3 Bloco de Forças Atuantes .....	47
4.3.4 Bloco de Resposta dos Motores .....	47
4.4 IDENTIFICAÇÃO DA CURVA DE VELOCIDADE E FORÇA .....	48
4.5 IDENTIFICAÇÃO DO MODELO MATEMÁTICO ESC/MOTOR/HÉLICE .....	50
4.6 IDENTIFICAÇÃO DO MOMENTO DE INÉRCIA .....	51
<b>5 EXPERIMENTOS E RESULTADOS .....</b>	<b>53</b>
5.1 SIMULAÇÃO .....	53
5.1.1 Controladores PID .....	53
5.1.2 Controlador Fuzzy .....	56
5.2 AMBIENTE REAL .....	58
5.2.1 Controladores PID .....	58
5.2.2 Controladores Fuzzy .....	58
<b>6 CONCLUSÃO .....</b>	<b>60</b>
<b>REFERÊNCIAS .....</b>	<b>61</b>
<b>APÊNDICE A – REDES NEURAIS ARTIFICIAIS .....</b>	<b>64</b>
A.0.2.1Aprendizado supervisionado .....	65
A.0.2.2Aprendizado não-supervisionado .....	65
A.0.2.3Aprendizado por reforço .....	65
<b>APÊNDICE B – SCRIPT PARA IDENTIFICAÇÃO DA FUNÇÃO DE TRANSFÊ- RENCIA NO MATLAB .....</b>	<b>67</b>
<b>APÊNDICE C – IMPLEMENTAÇÃO NO TIVA C PARA OBTENÇÃO DOS DADOS DO CONJUNTO MOTOR, HÉLICE E ESC .....</b>	<b>80</b>

## 1 INTRODUÇÃO

Os Veículos Aéreos Não Tripulados (VANTs) são veículos aéreos sem piloto embarcado, os quais podem ser autônomos ou controlados remotamente. Os primeiros relatos de uso de tais aparatos foram para fins militares, sendo utilizados em espionagens e ataques, evitando assim os riscos de perder soldados nessas manobras (DALAMAGKIDIS; VALAVANIS; PIEGL, 2009). Este projeto visa trabalhar com VANTs do tipo multirotor, especificamente com quatro rotores (quadricóptero), que ficam dispostos em uma estrutura na forma de "+" em que cada uma das extremidades contém um rotor. Esta estrutura possui algumas vantagens se comparada à de outros VANTs. Um quadricóptero pode manter-se em uma posição estática no ar, voar em qualquer direção, decolar, bem como aterrizar na vertical e voar a baixas velocidades. Tal comportamento assemelha-se a helicópteros convencionais, porém, com maior estabilidade devido à quantidade de rotores (VIEIRA, 2011). Uma desvantagem é a autonomia de voo, se comparado aos VANTs com asas, devido sua baixa aerodinâmica, necessitando assim uma maior quantidade de energia para manter sua sustentação.

Devido sua falta de aerodinâmica, tanto sua estabilidade bem como o movimento é mantido somente através de modificações das forças em que cada rotor exerce, baseado nos dados de sensores inerciais colocado estrategicamente no centro de massa do quadricóptero, controle esse chamado de atitude.

Nos últimos anos, os quadricópteros começaram a ser bastante estudados. Com a evolução dos microcontroladores com relação ao seu poder computacional e miniaturização aliado ao aumento de qualidade dos sensores com um baixo custo, facilitou a criação de sistemas mais precisos para seu controle. Ao mesmo tempo com um baixo peso embarcado favoreceu a sua construção, principalmente para obtenção de dados em pesquisas nas mais diversas áreas, como, mapeamento do solo em 3D na Geologia/Visão computacional (TAHAR; AHMAD; AKIB, 2011), verificações onde há riscos para seres humanos na prevenção de desastres nucleares (BOUDERGUI et al., 2011).

### 1.1 PROBLEMA DE PESQUISA

A utilidade de um quadricóptero e suas aplicações, por exemplo coleta de dados, aumenta quanto melhor a capacidade no controle ao manter-se em uma posição ou seguir uma determinada rota. O controle geralmente é efetuado utilizando-se de transformações de coordenadas, principalmente a de Euler.

A modelagem matemática necessária para simular o sistema é complexa. Nem sempre é possível obter um modelo exato e assim simular todas as possibilidades de perturbações com precisão. Os parâmetros de controle são geralmente baseados em aproximações do modelo real. Outra possibilidade é o uso de sistemas inteligentes com aprendizagem sem supervisão, nos quais a modelagem é criada de forma autônoma e adaptativa, diminuindo a complexidade matemática.

O interesse deste trabalho, reside na implementação de diferentes métodos para controle de atitude e na avaliação científica e empírica (em ambientes externos que não são controlados) de experimentos que visam mensurar critérios como:

- Tempo de resposta;

- Estabilidade em ambientes externos estando sujeito à várias perturbações aleatórias, geralmente imprevisíveis.

## 1.2 JUSTIFICATIVA

Sistemas de controle autônomos estão sendo muito utilizados em praticamente todas as áreas de conhecimento. Uma análise e comparação entre os diferentes tipos de controle pode possibilitar propostas de melhorias futuras nos métodos, não somente de atitude para quadricóptero, mas de veículos autônomos de modo geral, já que o mesmo sistema pode ser facilmente adaptado à diferentes veículos ou robôs, onde atualmente o principal foco é a utilização de métodos de inteligência artificial.

## 1.3 OBJETIVOS

### 1.3.1 Objetivo Geral

Comparação do comportamento de controladores Proporcional Integrador e Derivativo (PID) e fuzzy aplicados no controle de atitude em um quadricóptero.

### 1.3.2 Objetivos Específicos

- Estudo dos sistemas inteligentes com finalidade de controle;
- Escolha dos componentes a ser utilizado no protótipo;
- Modelagem matemática da dinâmica do quadricóptero;
- Validação para modelo matemático do protótipo;
- Ajustes dos parâmetros para o controlador PID através de simulação;
- Desenvolvimento do controle Fuzzy e ajuste através de simulação;
- Implementação dos controles em um quadricóptero;
- Coleta de dados em testes práticos dos controladores desenvolvidos;
- Análise dos dados e resultados.

## 1.4 ORGANIZAÇÃO DO TEXTO

Primeiramente será abordada a história dos VANTs do tipo quadricópteros, logo após é verificado o funcionamento dos componentes para um quadricóptero e em seguida as técnicas de controle, como: a técnica tradicional de controle PID e de sistemas inteligentes conhecidos como Lógica *Fuzzy*.

Após a revisão bibliográfica é desenvolvido o método de resolução do problema e materiais envolvidos.

## 2 REVISÃO BIBLIOGRÁFICA

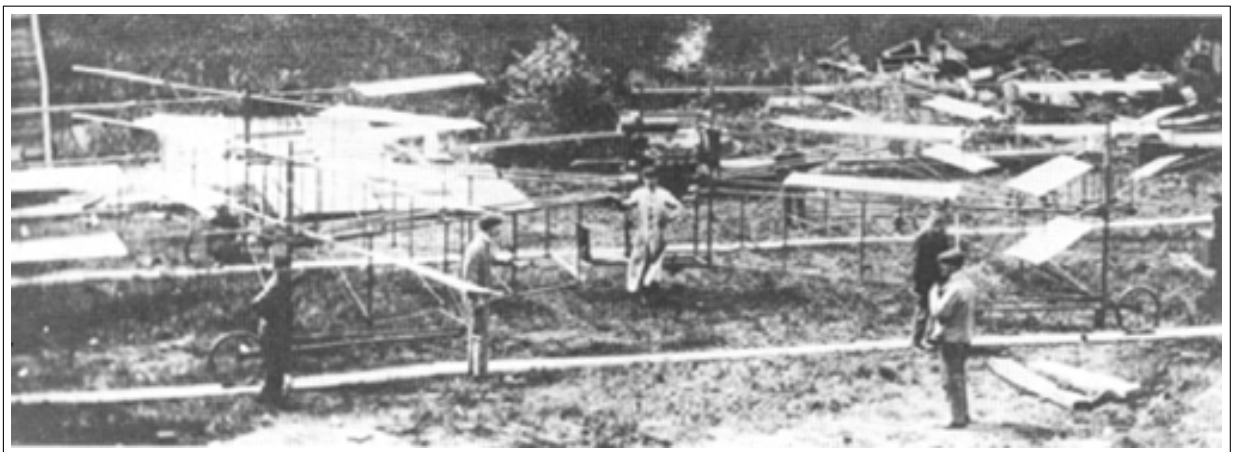
Para o estudo dos problemas envolvidos com multimotores é necessária uma revisão das pesquisas dos mesmos e sua evolução ao longo do tempo, além de problemas envolvidos em seu controle.

### 2.1 HISTÓRIA DOS QUADRICÓPTEROS

Um dos primeiros quadrotoros que é de conhecimento surgiu em 1904, com o Prof. Charles Richet, em uma tentativa de se fazer um pequeno helicóptero autônomo, a qual não obteve êxito. Inspirando-se em Richet e com sua supervisão, os acadêmicos e irmãos Jacques e Louis Bréguet, que mais tarde fundariam a Air France, desenvolveram o Giroplano 1 Bréguet-Richet como mostra a Figura 1.

Segundo Leishman (2002), a ideia consistia de uma estrutura em forma de cruz, na qual, cada extremidade continha um rotor com 8,1 m de diâmetro, cada par de motor era colocado para girar em um sentido, anulando assim a reação do torque na estrutura. A potência para os rotores era gerada por um único motor central a combustão, com força de aproximadamente 40 HP. No centro havia capacidade para 2 pessoas e controles mecânicos dos rotores com alavancas. O peso total era de aproximadamente 578 Kg.

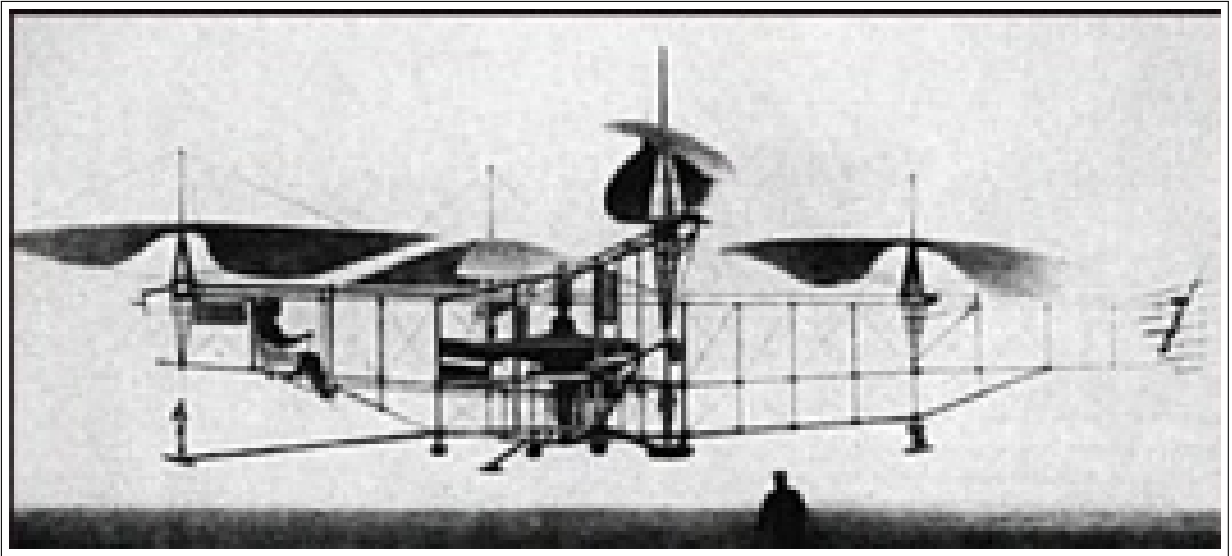
Entre agosto e setembro de 1907, o Giroplano 1, fez seus primeiros voos os quais foram obtidos em alguns momentos de testes a altura de 1,5m do chão, em curtos espaços de tempo. O protótipo era muito instável, pois, todo o controle ficava a cargo do piloto, de forma complexa. Apesar de um êxito parcial, os conceitos básicos se mantêm até hoje na construção de quadrotoros.



**Figura 1 – Bréguet-Richet em 1907**

**Fonte: Leishman (2002).**

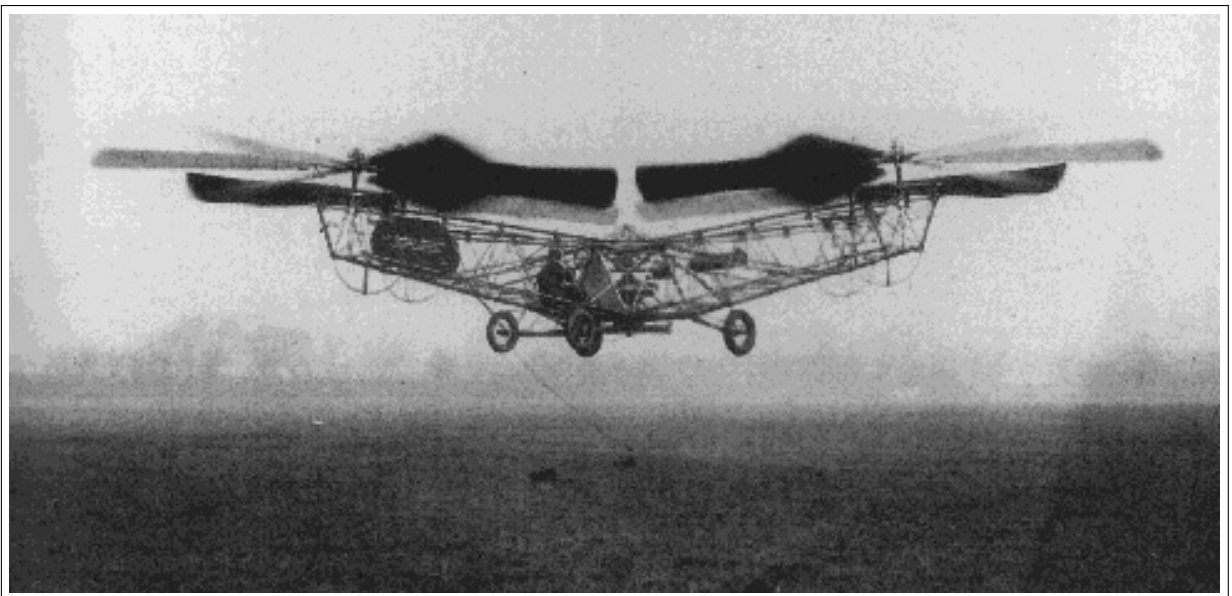
No ano de 1922, o engenheiro mecânico Etienne Ohemichen, desenvolveu um protótipo de voo vertical com quatro rotores e oito hélices para controle nas laterais, conhecido como Ohemichen No 2, como mostra a Figura 2. Seu peso era de 800 Kg com um motor de 180 HP e obteve uma grande estabilidade e controlabilidade, chegando a realizar voos de até 14 min.



**Figura 2 – Ohemichen No 2**

**Fonte: Ross (1953).**

Também no ano de 1922 o russo Georges Bothezate, trabalhando para o exército dos Estados Unidos, criou o Flying Octopus, Figura 3, um quadrotor de 1678 Kg propulsionado por um motor de 220 HP. Segundo (MUNSON, 1969), a exigência do exército era para que o veículo conseguisse voar em altitudes de 100 m, mas na prática atingiu apenas 5 m. O veículo possuía grande controlabilidade, mas seu alto custo de desenvolvimento, baixa potência e com alguns problemas técnicos, acabou sendo abandonado.



**Figura 3 – Flying Octopus**

**Fonte: Leishman (2002).**

## 2.2 QUADRICÓPTEROS NA ATUALIDADE

O desenvolvimento de quadricópteros não teve muita evolução após 1956 devido sua baixa eficiência, somente mais recentemente voltou a ser alvo de pesquisas e desenvolvimento

devido a evolução tecnológica, que supre as necessidades que antes os tornavam inviáveis, como materiais leves, controles e sensores rápidos de grande eficiência.

Ao contrário do início dos quadricópteros, sua finalidade atual não é mais em transporte de pessoas com uma cabine em sua estrutura. Hoje eles são pequenos, utilizados em laboratórios e comercialmente, como mini robôs para as mais diversas finalidades. Alguns serão mostrados a seguir.

### 2.2.1 AR Drone

Comercialmente um dos mais conhecidos quadricópteros é o AR Drone (PARROT, 2013), Figura 4, desenvolvido pela Parrot no começo de 2010. Atualmente é vendido o AR Drone 2.0, que possui uma estrutura em fibra de carbono, microcontrolador ARM Cortex A8 de 1GHz, sistema operacional Linux, câmera de vídeo HD, giroscópio, acelerômetro, magnetômetro, sensores de ultrassom e barômetro.

A principal inovação que a Parrot trouxe foi o controle através de um simples aplicativo de celular com uma conexão *wireless*, em que se pode visualizar a câmera do AR Drone e controlar todos seus movimentos.



**Figura 4 – AR Drone da Parrot**

**Fonte: Parrot (2013).**

### 2.2.2 Aerial Robotic Construction- ARC

Pesquisadores do Instituto Federal de Tecnologia de Zurique, desenvolveram uma nova abordagem na utilização de quadricópteros com o projeto ARC, com a finalidade de construção em grande escala de forma automatizada (WILLMANN et al., 2012). Robôs voadores, Figura



5, levam pequenos elementos de construção e posicionam-os através de um modelo digital de precisão. As construções com esta técnica não necessitam de andaimes como nas tradicionais e é facilmente escalável. No controle de atitude do quadricoptero é utilizado o Linear Quadratic Regulator (LQR).



**Figura 5 – Quadricoptero para *Aerial Robotic Construction***  
**Fonte: Willmann et al. (2012).**

Essas tarefas requerem cooperação entre os VANTs e alta precisão de localização, que atualmente eles o fazem com câmeras posicionadas de forma a localizar cada veículo e todo o gerenciamento é centralizado. Assim eles conseguem de forma autônoma e interativa realizar grandes construções, como na Figura 6.



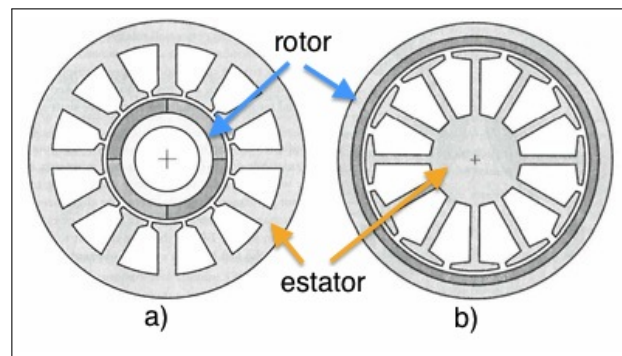
**Figura 6 – Construção realizada com  
ARC**  
**Fonte: Willmann et al. (2012).**

## 2.3 PRINCIPAIS COMPONENTES DE UM QUADRICÓPTERO

### 2.3.1 Motores *Brushless* PM

O motor *Brushless* é a peça que transforma energia elétrica em mecânica para o deslocamento do VANT, basicamente ele gera uma força sobre as hélices do VANT. Hanselman (1994) define o motor do tipo *Brushless* Permanent Magnet (PM), como síncrono e construído com bobinas elétricas no estator e ímãs permanentes no rotor, não possuindo escovas. Assim não há contato mecânico entre rotor e o estator, diminuindo o desgaste e eventuais problemas na transmissão de energia devido a posição fixa das bobinas. O movimento é produzido pelo campo magnético gerado em cada bobina.

Os *brushless* PM tem várias possibilidades de configurações, as principais são os com rotor interno e de rotor externo, mostradas na Figura 7, conhecidos como *inrunner* e *outrunner*. Também podem ter configuração monofásica, bifásica ou trifásica, geralmente o mais encontrado é o com três fases.

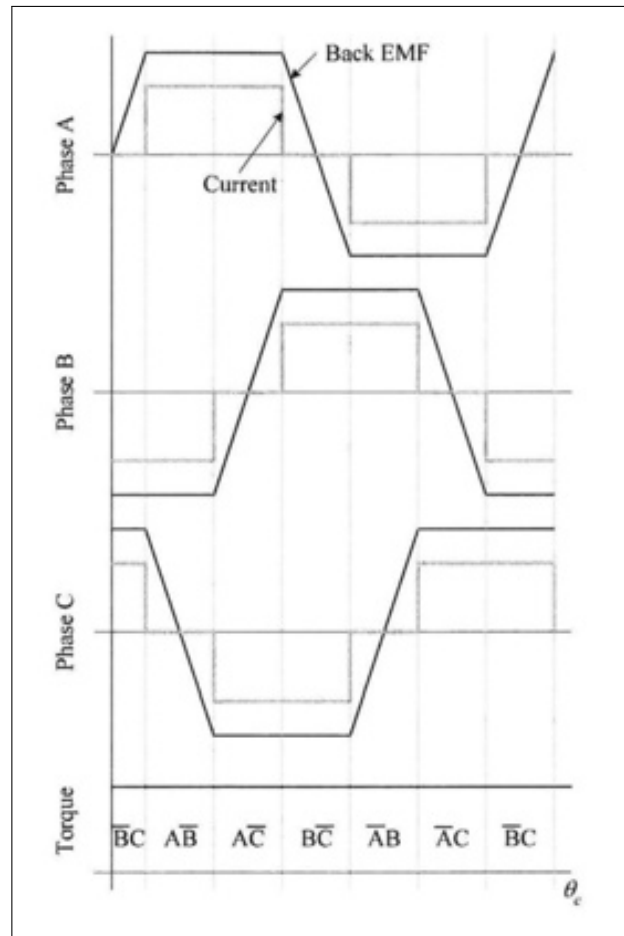


**Figura 7 – Configurações a) *inrunner*; b) *outrunner***

**Fonte: Hanselman (1994).**

Motores *Brushless Direct Current* (BLDC) necessitam de um controlador especial para gerenciar o seu movimento, os *Electrical Speed Controller* (ESC), os quais através de sensores de efeito hall, *encoders* ou com detectores de *Back Electro motive Force* (BEMF), encontram a posição atual do rotor para fornecer corrente no próximo conjunto de bobinas. Assim é mantida a rotação de forma contínua e em um único sentido no rotor.

Para produção de torque é necessário um controle eletrônico em cada fase do motor e gerar defasagem entre elas de maneira sincronizada com a posição atual do motor. Um motor com três fases tem uma defasagem de  $120^\circ$  em cada fase. O funcionamento pode ser verificado na Figura 8, segmentada a cada  $60^\circ$ , tendo a relação entre BEMF com a corrente, fase e torque (HANSELMAN, 1994).



**Figura 8 – Relação entre corrente, *Back EMF* e fase do motor *brushless***

**Fonte: Hanselman (1994).**

### 2.3.2 Hélices

Hélice é o componente que transforma a energia fornecida pelo motor em tração ou propulsão. Os dados mais significantes é o diâmetro e passo da hélice. Para hélice, o círculo de giro fornece o diâmetro e a distância percorrida por volta da hélice é o passo, como mostra a Figura 9. As informações sobre as hélices são fornecidas pelos fabricantes com o formato *DiâmetroXPasso* em polegadas, como por exemplo, uma hélice de 12" no diâmetro e 5" no passo é identificada como 12X5.

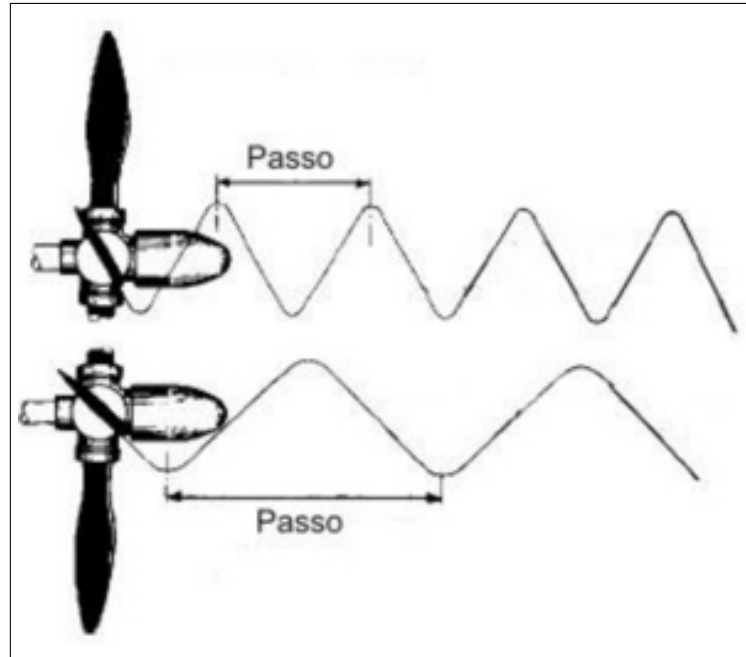


Figura 9 – Passo de hélice

Fonte: Adaptado de McCormick (1979).

McCormick (1979) define três parâmetros básicos que geram o comportamento da hélice:

- Coeficiente de empuxo  $c_T$  ;
- Coeficiente de potência  $c_P$  ;
- e raio da hélice  $r$  .

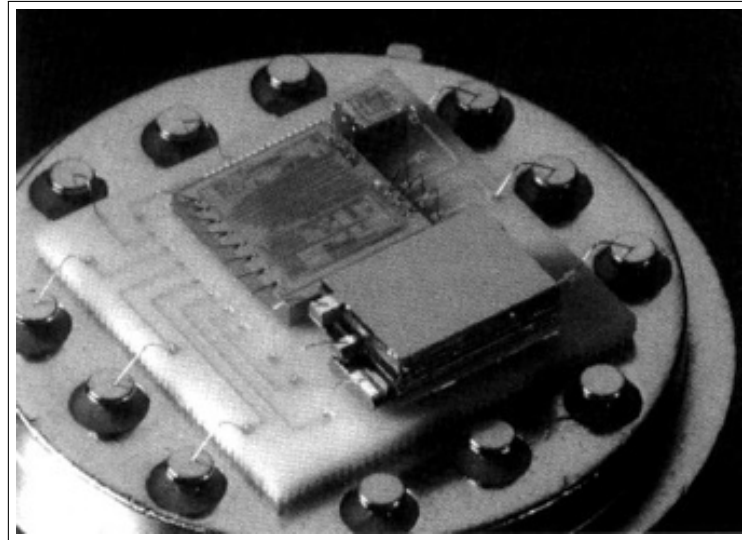
Com esses parâmetros é possível obter a força de empuxo  $T$  , Equação 1, e a potência da hélice  $P_p$ , Equação 2. Sendo  $\rho_a$  a densidade do ar e  $\omega_p$  a velocidade da hélice.

$$T = c_T \frac{4\rho_a r^4}{\pi^2} \omega_p^2 \quad (1)$$

$$P_p = c_P \frac{4\rho_a r^5}{\pi^3} \omega_p^3 \quad (2)$$

### 2.3.3 Sensores

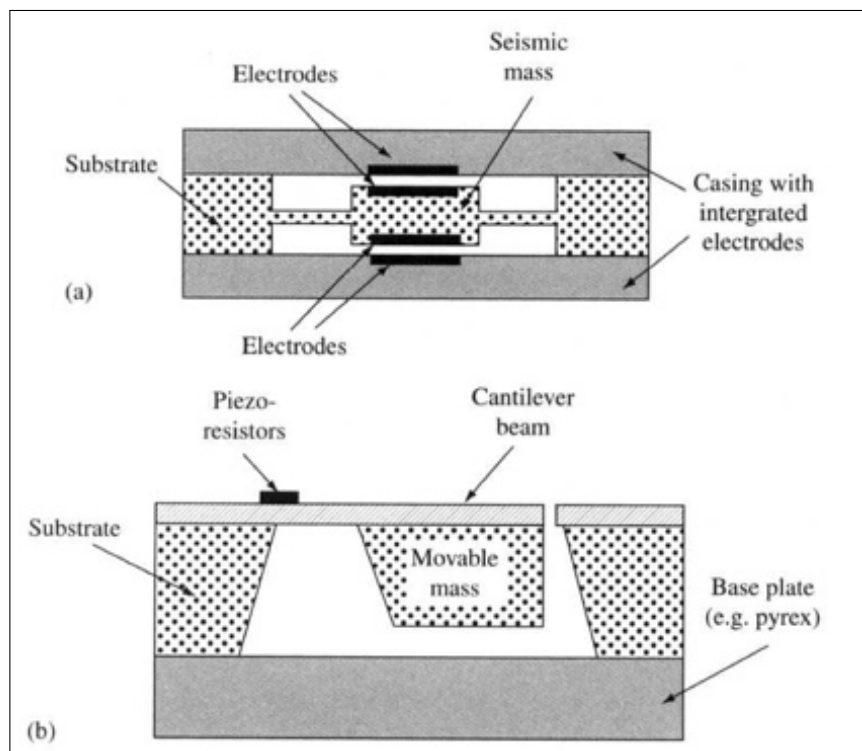
Com a evolução da tecnologia de Microsistemas Eletromecânicos (MEMS), os sensores se tornaram mais compactos, leves e com grande precisão. Gardner e Varadan (2001) definem MEMS como pequenos elementos mecânicos e eletrônicos em uma pastilha de silício.



**Figura 10 – Exemplo de acelerômetro do tipo MEMS**

Fonte: Gardner e Varadan (2001) .

Acelerômetros modernos utilizam a tecnologia MEMS, Figura 10, os quais são baseados no princípio de cantilever (GARDNER; VARADAN, 2001, p. 122), em que uma massa se descoloca sob uma força inercial e podem ser de dois tipos: capacitivo ou piezoresistivo, demonstrado na Figura 11.

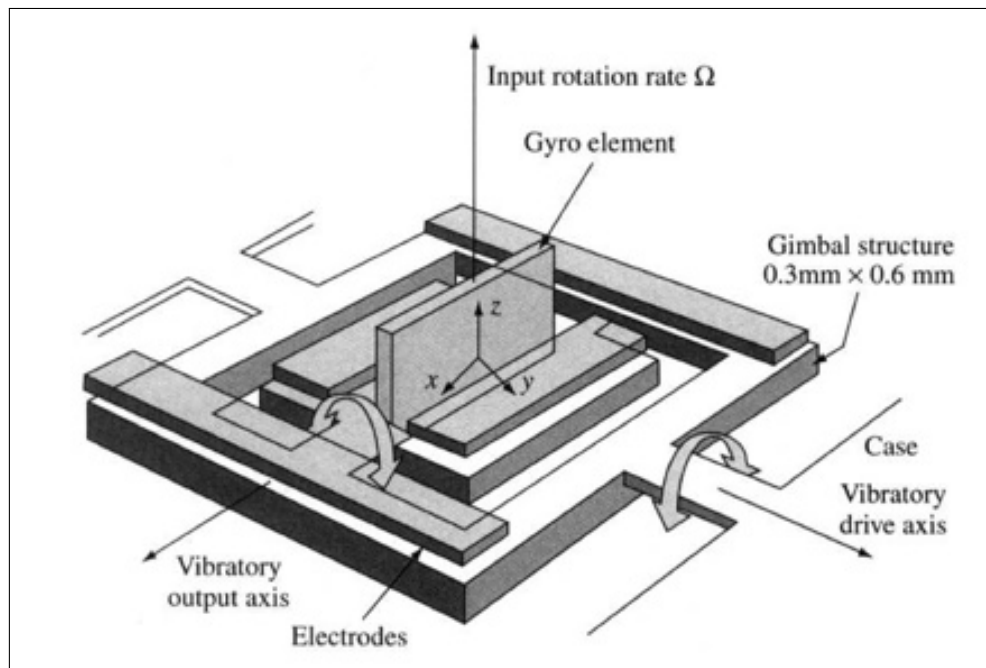


**Figura 11 – Princípio de catilever em acelerômetros**

Fonte: Gardner e Varadan (2001).

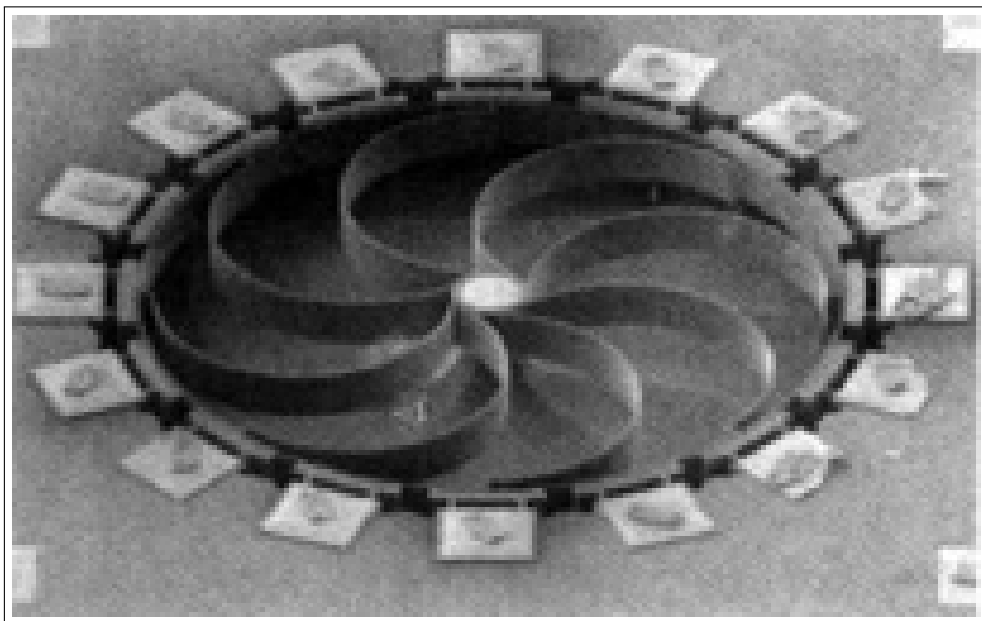
Outro sensor inercial que utiliza MEMS é o giroscópio, que mede as mudanças de orientação em um objeto. O princípio de funcionamento é baseado na força de Coriolis (CO-

RIOLIS, 1896). Basicamente, ressonadores acoplados recebem energia uns dos outros com a força de Coriolis, detectando o giro realizado sob um eixo, como mostra a Figura 12.



**Figura 12 – Esquema do giroscópio com a força de Coriolis**  
**Fonte: Gardner e Varadan (2001).**

Na Figura 13 é possível ver um encapsulamento de giroscópio em forma de anel, que utiliza o princípio de Coriolis.



**Figura 13 – Giroscópio em uma pastilha de silício**  
**Fonte: Gardner e Varadan (2001).**

## 2.4 MÉTODOS DE CONTROLE E OTIMIZAÇÃO

Nesta seção são apresentados os dois métodos de controle utilizados neste trabalho. Inicialmente é apresentado o controlador clássico PID, seguido pela descrição da lógica fuzzy para controle.

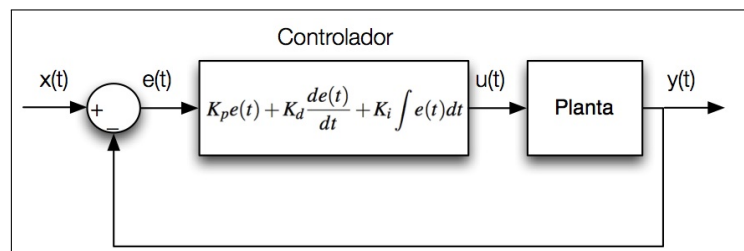
### 2.4.1 Controlador Proporcional, Integrativo e Derivativo (PID)

PID é um controlador versátil, metade dos controladores industriais atualmente empregam controles PID ou seus derivados (OGATA; MAYA; LEONARDI, 2003). Os ajustes do controlador são efetuados em campo, abrindo margem para aplicar diferentes métodos de otimização e derivações a fim de melhorar sua resposta. Algumas dessas técnicas podem ser algoritmos genéticos, redes neurais e até mesmo em conjunto com *fuzzy*.

Não há necessidade a priori de conhecer a planta matemática para utilização deste controle, assim o torna um controlador adaptável, principalmente para processos que não é possível efetuar modelagem matemática. Seu controle se dá pelas equações (3) para tempo contínuo e (4) em tempo discreto, em que  $K_p$  é o ganho proporcional para aumentar ou diminuir a energia na saída do controlador,  $K_d$  o ganho derivativo que atua em variações bruscas e  $K_i$  ganho integral que diminui o erro final. No caso de utilização de tempo discreto é acrescentado  $T_s$  relativo ao tempo de amostragem.

$$u(t) = K_p e(t) + K_d \frac{de(t)}{dt} + K_i \int e(t) dt \quad (3)$$

$$u(k) = K_p e(k) + \frac{K_d}{\delta} e(k) + K_i T_s \sum_{i=0}^k e(i) \quad (4)$$



**Figura 14 – Controle com PID no tempo contínuo**

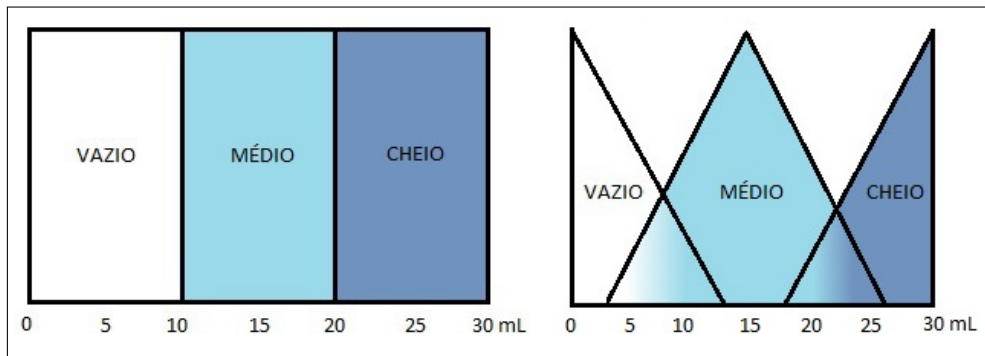
**Fonte: Adaptado de Ogata, Maya e Leonardi (2003).**

Na Figura 14 é demonstrado o esquema básico para controle com PID, com um sinal de entrada  $x(t)$  e saída  $y(t)$ , a diferença entre eles resulta em um valor de erro  $e(t)$ , que passa pela equação do PID aplicando um sinal de atuação na entrada da planta  $u(t)$ , como por exemplo, aumento ou redução de potência em um motor para estabilizar a velocidade. A planta é o modelo matemático do sistema físico a ser controlado.

### 2.4.2 Lógica Fuzzy

A visão da lógica clássica tem distinções bem definidas na separação de conjuntos, com pertence ou não ao conjunto, de forma inflexível, sem existir meio termo. Para muitos

problemas essa lógica pode não ser a mais adequada, como por exemplo, definir se um copo tem água. Pela lógica clássica o copo somente pode pertencer ou não ao conjunto do cheio, médio ou vazio, não existindo uma flexibilidade na fronteira do conjunto, por exemplo, "quase cheio" e "quase vazio", como mostra a Figura 15.



**Figura 15 – Representação do copo de água na forma de conjuntos com lógica clássica (a) e nebulosa (b).**

Na década de 1960, o professor Zadeh verificou a dificuldade na lógica clássica para problemas imprecisos e publicou um artigo com os conceitos dos conjuntos Fuzzy (ou nebuloso), considerada a origem dos conjuntos Fuzzy. Com o Fuzzy existem graus de pertinência, os quais podem assumir qualquer valor em um intervalo definido, criando a imprecisão que caracteriza os conjuntos nebulosos (ZADEH, 1965).

#### 2.4.2.1 Funções de Pertinência

Nos conjuntos nebulosos é necessário uma função de pertinência que para cada  $x$  defina o grau de pertinência no conjunto *fuzzy*, com forma  $\rho_A : X \rightarrow [0, 1]$ . A principal característica é que a progressão ou regressão no grau de pertinência seja gradativo, eliminando as limitações impostas na teoria dos conjuntos clássicos (CALVO, 2007).

Existem muitos formatos para as função de pertinência e nem sempre é fácil escolher o que mais se adapta ao problema a ser resolvido. Algumas funções, mostradas na Figura 16, usuais são:

- Triangular:

$$\rho_A(x) = \begin{cases} 0, & \text{se } x \geq a \\ \frac{x-a}{m-a}, & \text{se } x \in [a, m] \\ \frac{b-x}{b-m}, & \text{se } x \in [m, b] \\ 0, & \text{se } x \leq b \end{cases} \quad (5)$$

- Trapezoidal:

$$\rho_A(x) = \begin{cases} 0, & \text{se } x > a \\ \frac{x-a}{m-a}, & \text{se } x \in [a, m] \\ 1, & \text{se } x \in [m, n] \\ \frac{b-x}{b-n}, & \text{se } x \in [n, b] \\ 0, & \text{se } x < b \end{cases} \quad (6)$$



- Gaussiana:

$$\rho_A(x) = e^{-\sigma(x-m)^2}, k > 0 \quad (7)$$

Sendo que  $m$  é um valor modal, e  $a$  e  $b$  menor e o maior limitante, respectivamente, para valores não nulos de  $\rho_A(x)$ .

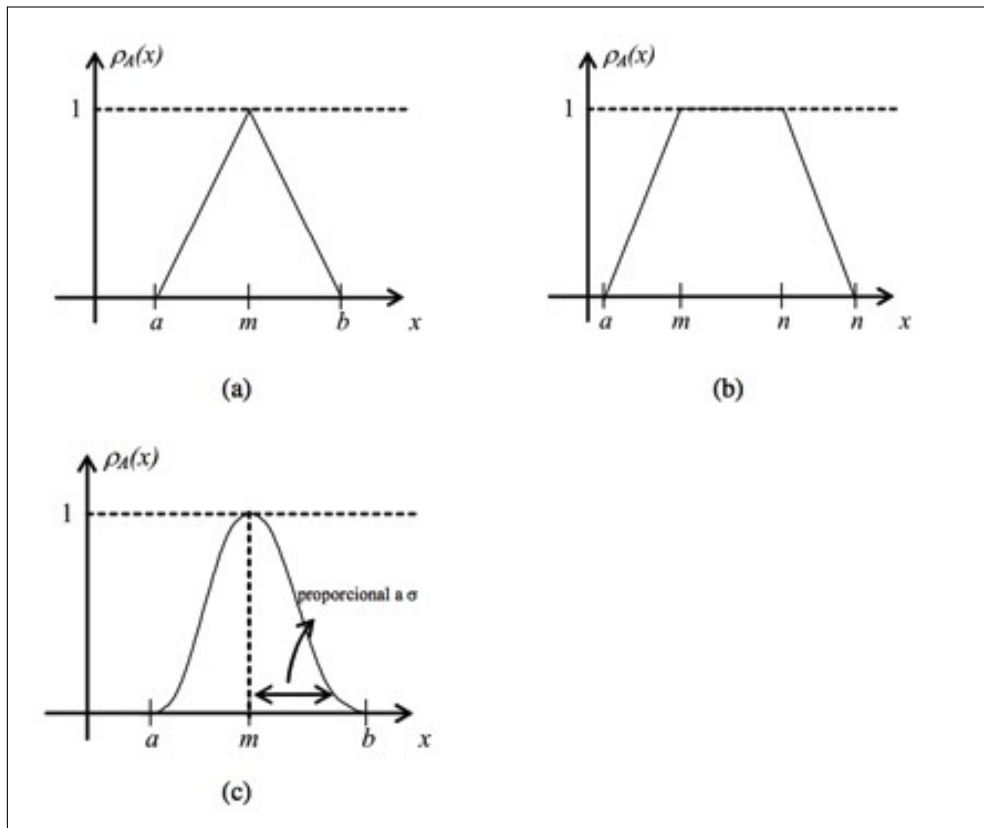


Figura 16 – Funções de pertinência de conjuntos nebulosos (a) triangular, (b) trapezoidal e (c) gaussiana

Fonte: Adaptado de CALVO (2007).

Também é possível otimizar as funções de pertinência fazendo uso de técnicas em redes neurais (CALVO et al., 2004) e algoritmos genéticos (HOFFMANN, 2001).

#### 2.4.2.2 Operadores Fuzzy

Ao trabalhar com conjuntos *fuzzy* é possível realizar várias operações. Segundo (DELGADO, 2002), existem operações com argumento único, que modificam o formato da função de pertinência, como, normalização (que converte um conjunto subnormal em normal), Figura 17, concentração (que diminui os valores da função de pertinência) e dilatação (que aumenta os valores da função de pertinência).

As operações básicas de múltiplos argumentos, são na maioria derivadas da teoria de conjunto clássica, que são feitas através da função de pertinência, Figura 18. Algumas definidas por (ZADEH, 1965) são:

- Vazio:  $A = \emptyset$ , se e somente se  $\rho_A(x) = 0, \forall x \in X$

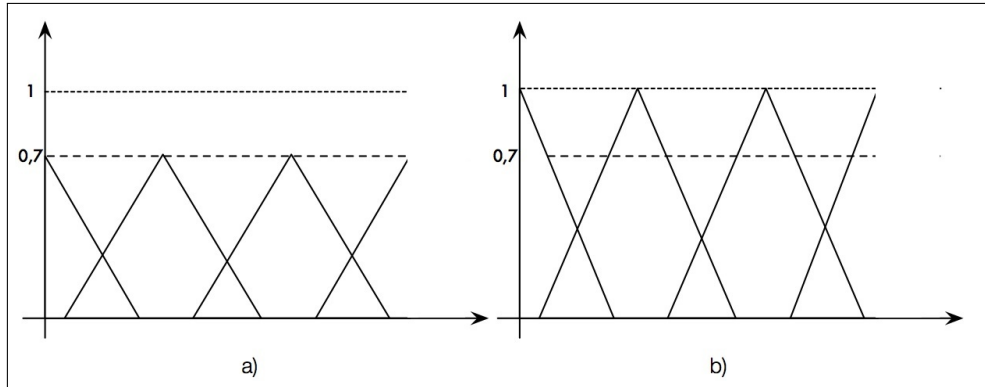


Figura 17 – Normalidade (a) subnormal, (b) normal.

- Complemento:  $\rho_{\bar{A}}(x) = 1 - \rho_A(x), \forall x \in X$
- Igualdade:  $A = B$ , se e somente se  $\rho_A(x) = \rho_B(x), \forall x \in X$
- Pertence:  $A \subset B$ , se  $\rho_A(x) \leq \rho_B(x), \forall x \in X$
- Multiplicação:  $A \cdot B, \rho_{A \cdot B}(x) = \rho_A(x) \cdot \rho_B(x), \forall x \in X$
- União:  $A \cup B, \rho_{A \cup B}(x) = \rho_A(x) \vee \rho_B(x), \forall x \in X$
- Intersecção:  $A \cap B, \rho_{A \cap B}(x) = \rho_A(x) \wedge \rho_B(x), \forall x \in X$

Na união e intersecção são utilizados os operadores de disjunção OU e a conjunção E, respectivamente, devido ao isomorfismo entre a teoria dos conjuntos e a lógica proposicional bi-valores.

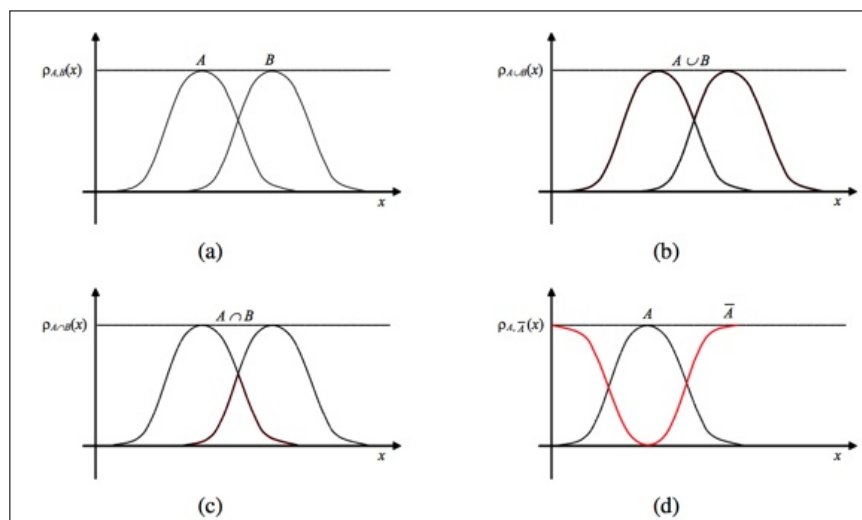


Figura 18 – Operações *fuzzy*: (a) Conjunto A e B; (b) União de A e B; (c) Intersecção de A e B; (d) Complemento de A

Fonte: CALVO (2007).

Com os conjuntos nebulosos, os operadores básicos se enquadraram em duas classes com funções de duas variáveis, conhecidas como t-norma, ou simplesmente **t** e s-norma, alguns autores referem-se como t-co-norma, **s**.

Segundo (JÚNIOR; YONEYAMA, 2002), uma operação  $t : [0, 1] \cdot [0, 1] \rightarrow [0, 1]$  é dita ser uma t-norma se:

- a)  $t(0,0) = 0$ , condição de contorno.
- b)  $t(a,b) = t(b,a)$ , comutatividade.
- c)  $t(a,t(b,c)) = t(t(a,b),c)$ , associatividade.
- d)  $(a \leq c) \wedge (b \leq d) \rightarrow t(a,b) \leq t(c,d)$ , monotonicidade.
- e)  $t(a,1) = a$ , condição de contorno.

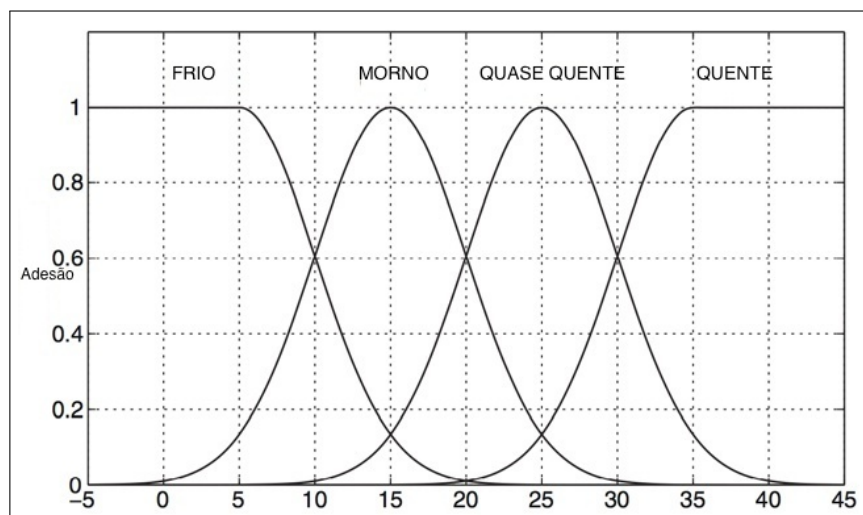
E s-norma quando uma operação  $s$  possuir as propriedades a),b),c),e) e ainda satisfazer:

- $s(1,1) = 1$
- $s(b,0) = b$

#### 2.4.2.3 Variáveis Linguísticas

Conjuntos *fuzzy* são geralmente identificados por variáveis linguísticas, por exemplo, "alto", "médio" ou "pequeno". A utilização faz com que a lógica *fuzzy* se aproxime da forma de pensar humana, usando a linguagem natural em vez da simbologia matemática.

Em Delgado (2002) é afirmado que número de termos linguísticos, configuração dos conjuntos *fuzzy* a cada termo e distribuição no universo de discurso é dependente da natureza do problema. Em um problema de controle do nível de água é possível nomear com "vazio", "cheio", "quase cheio". Já em um problema de controle da temperatura, o mais apropriado é utilizar termos como "frio", "morno", "quase quente" e "quente", demonstrado na Figura 19.



**Figura 19 – Exemplo de utilização das variáveis linguísticas**

Fonte: Traduzido de Lilly (2011).

#### 2.4.2.4 Regras Fuzzy

A maioria das decisões humanas ocorrem na forma da regra "se-então". Na Lógica clássica existem quatro formas de "se-então", que correspondem a *modus ponendo ponens*, *modus tollendo tollens*, *modus ponendo tollens* e *modus tollendo ponens*, que tem origem no latim, "modo que afirmando afirma", "modo que negando nega", "modo que afirmando, nega" e "modo que negando, afirma", respectivamente (LILLY, 2011).

É possível exemplificar as regras com a direção de um veículo para um alvo:

- Se o alvo está à frente, então, sentido da direção é reta (*modus ponendo ponens*).
- Se o alvo não está à frente, então, sentido da direção não é reta (*modus tollendo tollens*).
- Se tem um obstáculo à frente, então, sentido da direção não é reta (*modus ponendo tollens*).
- Se tem um obstáculo mas não é à frente, então, sentido da direção é reta (*modus tollendo ponens*).

Esses modos de raciocínio são possíveis de serem implementados na lógica *fuzzy*. De fato, a grande maioria das regras "se-então" utilizados em controle e identificação com *fuzzy* estão na forma *modus ponens*.

Como exemplo, o controle de parada de um carro (LILLY, 2011), "Se VELOCIDADE é RÁPIDA então PRESSÃO NO FREIO é ALTA", o qual "VELOCIDADE é RÁPIDA" é a premissa e "PRESSÃO NO FREIO é ALTA" a consequência. E ainda, VELOCIDADE é a variável linguística de entrada, RÁPIDO é o valor linguístico para VELOCIDADE e é o conjunto *fuzzy* no universo VELOCIDADE. PRESSÃO NO FREIO é a variável linguística de saída e ALTA é o valor linguístico de PRESSÃO NO FREIO é também um conjunto *fuzzy* do universo PRESSÃO NO FREIO.

O número de regras necessárias para um problema está associado à tarefa que será realizada. Geralmente são necessárias muitas regras, abrangendo as diferentes condições na tarefa. Esse conjunto de regras é conhecido como Base de Regras.

#### 2.4.2.5 Inferência Fuzzy

Para Delgado (2002), qualquer método de raciocínio pode ser definido em um processo de inferência que produz conclusões a partir de um conjunto formado por uma base de regras e fatos conhecidos.

Uma declaração de implicação ou condicional *fuzzy* é uma descrição de variáveis linguísticas. Considerando dois conjuntos *fuzzy*  $A$ , que representa valores linguísticos no universo  $X$ , e  $B$ , que representam valores linguísticos no universo  $Y$ . Ao realizar uma regra *fuzzy* da forma SE  $A$  ENTÃO  $B$ , pode ser definida matematicamente como:  $R: A \rightarrow B = A \times B$ .

Ao realizar duas relações como:

- $R_1: SE A ENTÃO B$ ;
- $R_2: SE B ENTÃO C$ ;

É possível deduzir,  $R_{12}$ : SE A ENTÃO C, ou seja, uma composição  $R_{12} = R_1 \circ R_2$ , que pode ser definida por regras do tipo *max-min*:

$$\rho_{12}(x, z) = \bigvee_y (\rho_{R_1}(x, y) \wedge \rho_{R_2}(y, z)) \quad (8)$$

ou por um tipo *max-produto*:

$$\rho_{12}(x, z) = \bigvee_y (\rho_{R_1}(x, y) \cdot \rho_{R_2}(y, z)) \quad (9)$$

As operações *max-min* e *max-produto*, podem ser definidas como o produto interno de duas matrizes ao trabalhar em conjuntos *fuzzy* discretos, com a multiplicação substituída pela operação *min* e a soma substituída pela *max* (GOMIDE; GUDWIN; TANSCHKEIT, 1995).

Levando em consideração um dos métodos mais importantes de inferência (CALVO, 2007), o conjunto *fuzzy* de saída é obtido com: comparação, agregação e conclusão, onde:

- Comparação de proposições: Para cada variável  $X_i$ , o fato da premissa 1 é comparado com a hipótese da premissa 2.
- Agregação: Os resultados das comparações são agregados em um operador t-norma produto.
- Conclusão: Para cada regra de inferência resulta em uma conclusão, função essa consequente da regra e o valor agregado das comparações.

#### 2.4.2.6 Sistemas Fuzzy

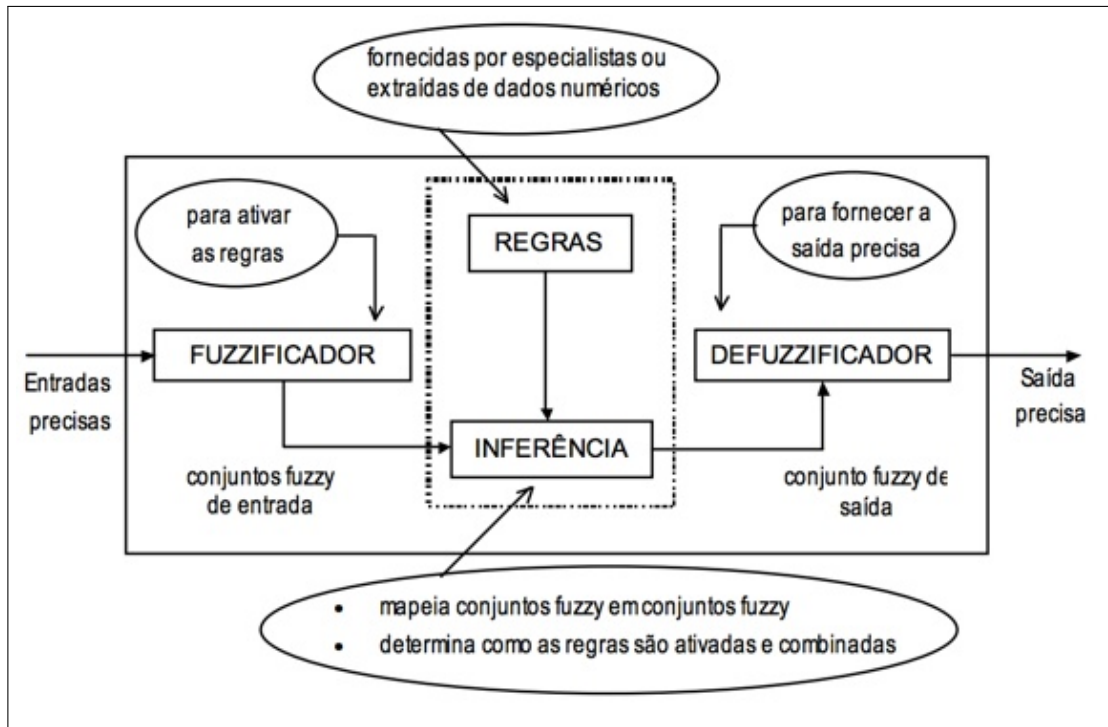
Também conhecido como sistema de inferência *fuzzy* (SIF), é baseado em um modelo simples, onde tem entradas, processamento e saída, como mostra a Figura 20.

A entrada é um número real determinístico que é normalizado, conhecido como "crisp", geralmente são dados de sensores, que passam por um processo de "fuzzificação", que consistem em converter o *crisp* para conjuntos *fuzzy* com as funções de pertinência e então passar pelo estágio de inferência (processamento).

O processo de inferência é feito com a base de regras e a base de dados, com o perfil das funções de pertinência utilizadas nas regras, gerando um novo conjunto *fuzzy* de saída. A saída então é transformada novamente em *crisp*, com uma função. A função do processo de saída pode ser realizada de diversas maneiras, algumas mais usuais são:

- Média dos Máximos: É feita uma média dos valores máximos correspondentes da função de pertinência do conjunto de saída, gerando um valor *crisp*.
- Centro de Massa: O valor *crisp* é o centro de massa,

$$\hat{z} = \frac{\int_z \rho_S(z) z dz}{\int_z \rho_S(z) dz} \quad (10)$$



**Figura 20 – Organização de um sistema fuzzy**

Fonte: (CALVO, 2007).

- Centro de Área: O valor *crisp* resulta do balanço de duas áreas,

$$\int_{-\infty}^{\hat{z}} \rho_S(z) dz = \int_{\hat{z}}^{\infty} \rho_S(z) dz \quad (11)$$

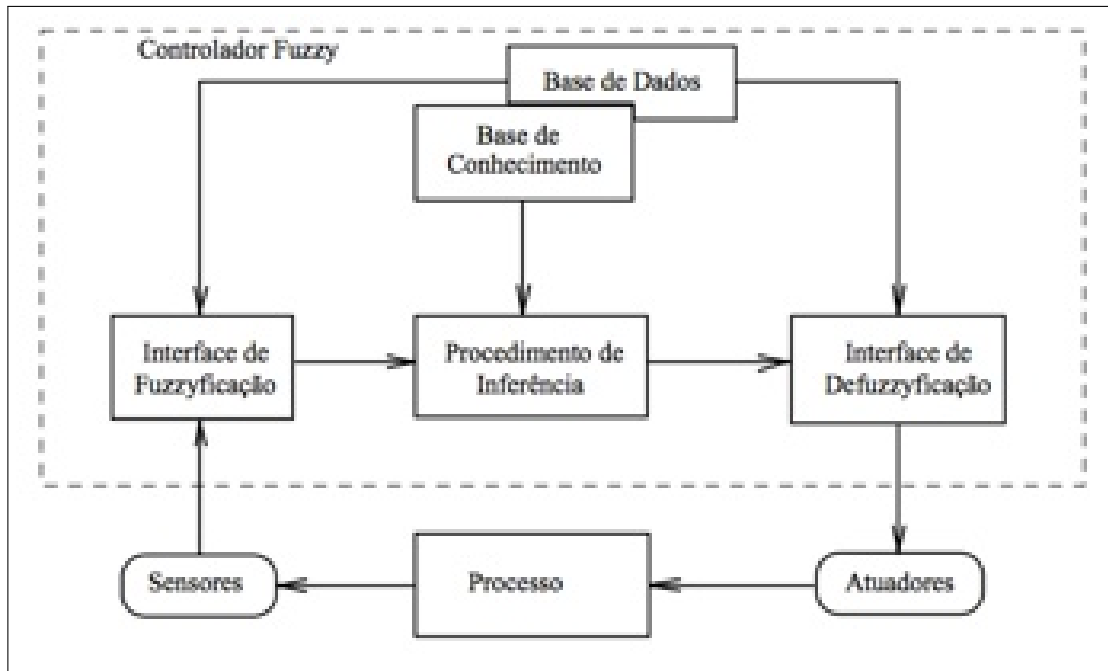
Na computação nebulosa é necessário um raciocínio nebuloso e um mecanismo de inferência para obter a resposta. Existem vários mecanismos, sendo os mais comuns: inferência composicional e escalonada (DELGADO, 2002). Com os mecanismos de inferência são definidos modelos de sistemas nebulosos, sendo os mais utilizados: Mamdani (MAMDANI; ASSILIAN, 1975) e Takagi-Sugeno (TAKAGI; SUGENO, 1985).

#### 2.4.2.7 Controladores Fuzzy

O modelo desenvolvido por Mamdani e Assilian (1975), surgiu com a tentativa de controlar uma caldeira de máquina a vapor, usando uma base de regras para controle baseadas em variáveis linguísticas.

Segundo Gomide, Gudwin e Tanscheit (1995), um controlador *fuzzy* baseado em regras é descrito por meio das regras linguísticas interconectadas com várias ações a serem tomadas. A estrutura básica do controlador pode ser verificada na Figura 21.

Na saída do processo são adquiridas as informações determinísticas com sensores, as quais passam pelo processo de "fuzzyficação" que as transformam em conjunto *fuzzy*. Entre esse processo geralmente estão os conversores A/D e D/A, os fatores de escala e quantização. A alimentação para entrada do processo é feita de forma inversa, transformando o conjunto *fuzzy*, através da "defuzzyficação", em um valor determinístico.



**Figura 21 – Controle com *fuzzy***

**Fonte: Gomide, Gudwin e Tanscheit (1995).**

A essência do controlador é definida através das bases de regras (conhecimento) que são feitas por um especialista da área de atuação, a teoria de conjuntos *fuzzy* apenas é o meio de transformação dessas informações em dados matemáticos, onde são aplicadas as inferências (GOMIDE; GUDWIN; TANSCHAIT, 1995).

### 3 MATERIAIS E MÉTODOS

Os materiais deste trabalho estão apresentados na Seção 3.1. Os métodos estão apresentados na Seção 3.2.

#### 3.1 MATERIAIS

O quadricoptero geralmente é construído com materiais leves, como alumínio ou fibra de carbono e deve suportar as forças geradas pelos rotores que ficam em cada extremidade da estrutura em forma de "+". Para a geração de força nos rotores é optado por motores *brushless* DC, o qual precisa de um *Electronic Speed Controller* (ESC) para gerar os sinais de potência. Ao todo em um quadricoptero são necessários quatro ESCs, um para cada motor, que são interligados a uma placa central com um microcontrolador, com os algoritmos de controle, que fornece um sinal de PWM ao ESC, informando a potência desejada. Junto a placa de controle também são colocados sensores inerciais, como, acelerômetro, giroscópio e magnetômetro, os quais fornecem informações para o controle de atitude do quadricoptero.

Na Figura 22 é apresentado o diagrama de relacionamento entre os componentes.

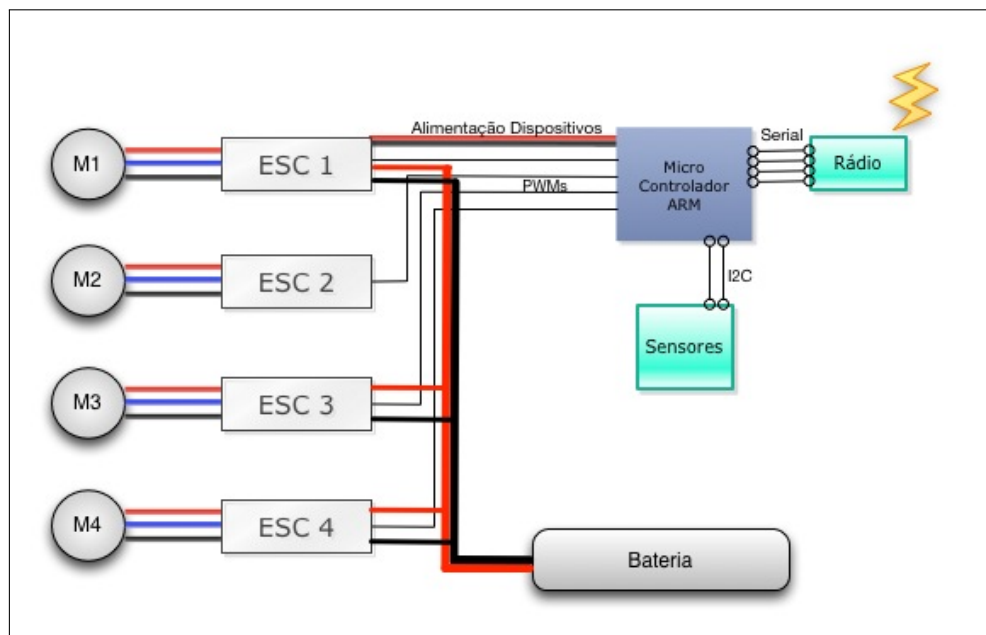


Figura 22 – Esquema básico para o quadricoptero.

##### 3.1.1 Estrutura

Uma estrutura ideal para suporte do quadricoptero tem que ser resistente o suficiente para resistir às tensões geradas pelo momento e possíveis impactos, com um material mais leve possível, aumentando assim sua autonomia de voo. Alguns dos materiais com estas características são o alumínio, fibra de vidro e a fibra de carbono.

Hoje estão disponíveis no mercado diversas estruturas pré-moldadas em fibra de carbono, unindo leveza e a resistência necessária. Para este projeto foi adquirida a estrutura Talon fabricada pela Turnigy, mostrada na Figura 23.





**Figura 23 – Frame Talon Turnigy V1.**

### 3.1.2 Motores

Neste projeto foi utilizado o motor *outrunner* BLDC Turnigy L2215J (TURNIGY, 2013) trifásico, apresentado na Figura 24 e configurações na Tabela 1. O motor turnigy L2215J funciona com a relação de 900 RPM por volt, sendo assim, a tensão máxima de 11,1V, obtemos teoricamente a velocidade máxima de 9990 RPM.



**Figura 24 – Motor Brushless DC.**

**Tabela 1 – Especificações do motor Turnigy L2215J**

<b>Parâmetro</b>	<b>Valor</b>
Tensão	11,1 V
Kv	900 RPM/V
Potência Máxima	200 W
Corrente Máxima	18 A
Corrente sem carga	0,6 A
Polos	12
Peso	61 g

**Fonte: Dados do fabricante.**

### 3.1.3 Hélices

Através das equações 1 e 2 na Seção 2, que aumentando no diâmetro da hélice, aumenta consideravelmente o empuxo, como consequência é necessário uma maior potência do motor, portanto as hélices devem ser escolhidas baseadas na potência do motor. Outro fator é a velocidade angular das hélices, quanto maior a relação RPM/Volt menor deve ser a hélice.

Neste projeto a escolha foi uma hélice 10x4,5 com peso de 11g. Figura 25, recomendada pela fabricante do motor, a qual, tem coeficientes próximos de  $c_T = 0,11$  e  $c_P = 0,05$ , sem considerar as variações com relação a velocidade e densidade do ar. O empuxo máximo gerado por cada motor será de aproximadamente 800g, sendo o total de 3,2Kg de empuxo com quatro motores.

O peso do protótipo é estimado em 1,2Kg, portanto para manter no ar plainando será exigido aproximadamente 37,5 % de potência total em cada motor, sobrando força suficiente para movimentos rápidos, além de permitir carregar peso extra, como uma câmera fotográfica.

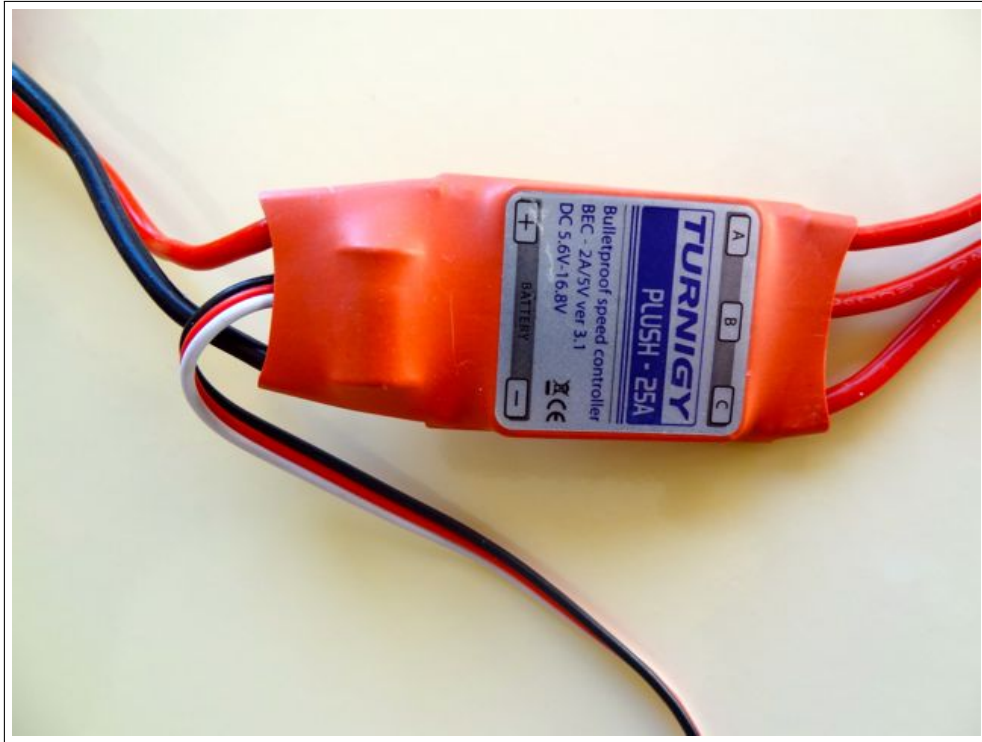
Cabe ressaltar que antes da utilização das hélices se faz necessário realizar seu balanceamento, certificando que o peso e seu momento nas duas pás seja o mesmo, mantendo o equilíbrio e diminuindo as vibrações que podem levar a rupturas na estrutura e menor estabilidade em voo. Observe que no lado esquerdo da hélice mostrado na Figura 25, foi acrescentado fita adesiva de modo a fazer o balanceamento da hélice.



**Figura 25 – Hélice 10x4.5 balanceada.**

### 3.1.4 ESCs

Os ESCs devem suportar a corrente máxima do motor, no caso 18 A. Neste projeto será utilizado os ESC Turnigy Plus 25 A, que conta com suporte a motores com 12 polos, pesa 22g e seu sistema é microprocessado, reduzindo assim seu tamanho e peso. Os comandos de potência na saída são feitos através de PWM na entrada, a uma taxa máxima de 200 Hz.



**Figura 26 – ESC de 25A fabricado pela Turnigy.**

### 3.1.5 Bateria

Nas baterias é preciso levar em consideração a relação carga/peso e a corrente máxima de descarga. Cada motor consome 18A na potência máxima, para quatro é um total de 72A que devem ser suportados pela bateria. A escolha foi da bateria com três células de Lítio-Polímero (Li-Po) da marca Zippy, que pode ser visualizada na Figura 27. A bateria pesa 264g, possui carga de 3700mAh, tensão de 11,1V e descarga máxima de 92A.

### 3.1.6 Sensores

Para obter os dados de inercia e posição foi utilizado o acelerômetro e giroscópio contidos no MPU-6050 fabricado pela Invensense (2013), bússola digital HMC5883 fabricado pela Honeywell (2010) e barômetro MS5611 fabricado pela Specialties (2012), mostrados na Figura 28. As escolhas são devido as suas características, documentação e resolução.

**Tabela 2 – Especificações dos sensores**

	<b>Acelerômetro</b>	<b>Giroscópio</b>	<b>Barômetro</b>
<b>Qtde. Eixos</b>	3	3	1
<b>Escala</b>	$\pm 2g$ a $\pm 16g$	$\pm 250, \pm 500, \pm 1000, \pm 2000^\circ/s$	10 a 1200 mbar
<b>Consumo</b>	500 $\mu A$	3,6mA	1 $\mu A$
<b>Conversor ADC</b>	16-bit	16-bit	24-bit

**Fonte: Dados do fabricante.**

Para obter a velocidade em que os motores estão girando foi utilizado um sensor de efeito Hall SS460P da Honeywell (HONYWELL, 2013). A utilização deste componente se deve



Figura 27 – Bateria Zippy Compact LiPo 3S 25C 3700mAh.

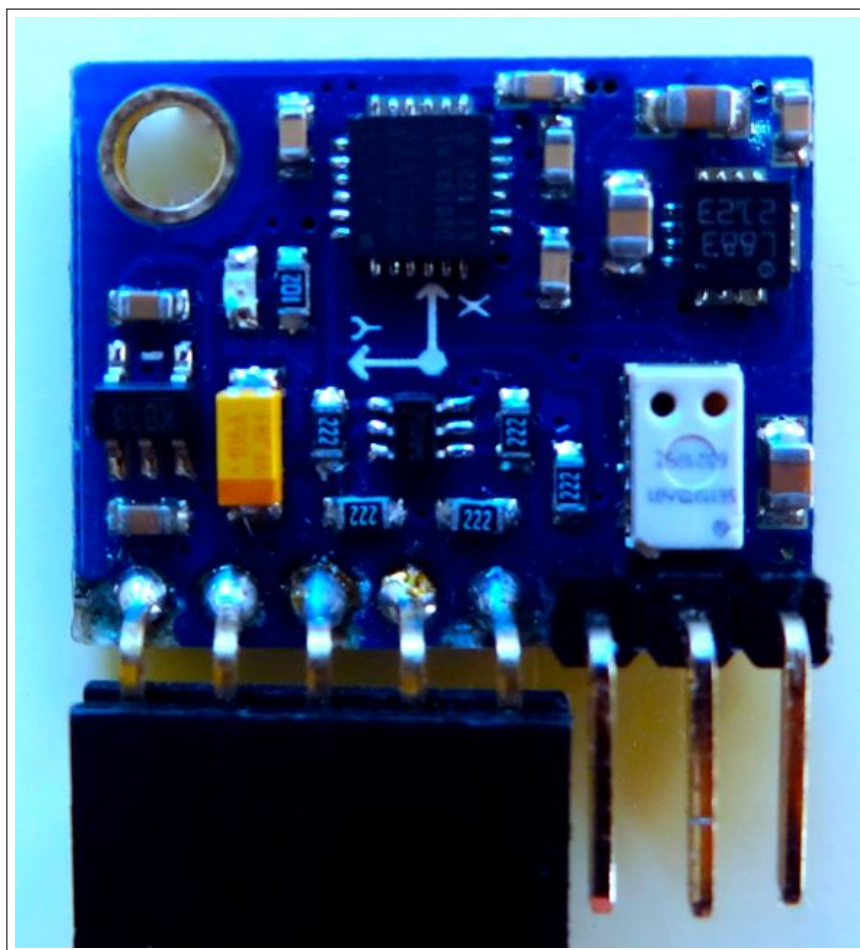


Figura 28 – Conjunto de sensores 10 DOF.

a ele ser digital, não necessitando assim circuitos adicionais para condicionamento de sinais.

Quando exposto a uma polaridade magnética norte, sua saída é estado "baixo". Ao expor a uma polaridade sul, sua saída digital é estado "alto".

### 3.1.7 Microcontrolador

Os processamentos são efetuados por um microcontrolador do tipo ARM Cortex M4F, o qual possui unidade de ponto flutuante, que aumentam a velocidade dos cálculos envolvendo ângulos, parte fundamental para estabilização no processo de controle.

O modelo escolhido é o Tiva TM4C123G fabricado pela Texas Instruments (INSTRUMENTS, 2014), sua utilização é facilitada por um kit de desenvolvimento chamado *Launchpad*, Figura 29, integrando todos os componentes necessários para seu funcionamento, a um custo relativamente barato. As especificações estão na Tabela 3. Outro ponto favorável é a possibilidade de usar 16 saídas PWMs em *hardware*, facilitando a tarefa de controle dos motores.

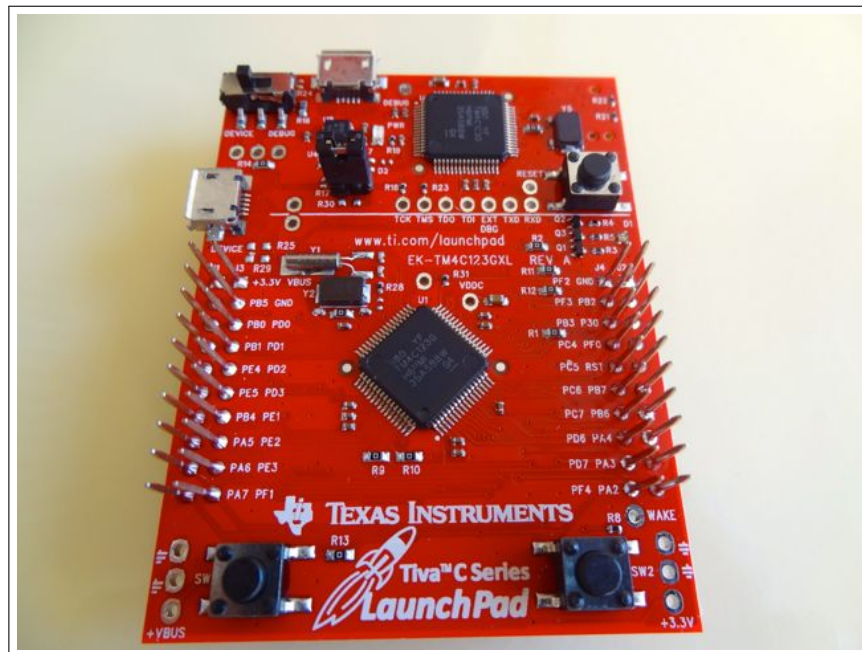


Figura 29 – Launchpad Tiva com ARM Cortex M4F da Texas.

Tabela 3 – Principais características do TM4C123G

Característica	Descrição
Performance	80MHz, 100 DMIPS
Flash	256KB
SRAM	32KB
EEPROM	2KB
Quantidade de UARTs	8
Quantidade de SSI	4
Quantidade de $I^2C$	4
Quantidade de PWMs	2 Módulos com um total de 16 saídas PWM
Quantidade de Pinos	64 LQFP

Fonte: Dados do fabricante.

### 3.1.8 Balança

Para medir a força de empuxo foi utilizado uma balança digital de precisão SF-400, mostrada na Figura 30, com graduação de 1g e peso máximo suportado de 7 Kg.



**Figura 30 – Balança de precisão SF-400.**

## 3.2 MÉTODOS

Após estabelecida a problemática para a realização deste trabalho, foi realizada pesquisa bibliográfica sobre drones do tipo quadricóptero, controle PID e lógica fuzzy em livros e artigos científicos disponíveis em bases de dados conhecidas como Google Scholar, IEEE, ACM, Springer, o resultado destas pesquisas estão descritos no Capítulo 2.

A partir da pesquisa bibliográfica foram escolhidos e adquiridos os componentes para a construção do protótipo a ser utilizado nos experimentos. Com a chegada dos componentes a estrutura física do protótipo foi construída.

O próximo passo seria iniciar a simulação da dinâmica dos controles do quadricóptero, no entanto, antes disso percebeu-se a necessidade de fazer a modelagem da dinâmica do quadricóptero (Seção 3.2.2) e a cinemática (Seção 3.2.1). As equações envolvendo a dinâmica do quadricóptero são complexas, assim foram necessárias algumas simplificações a fim de facilitar os cálculos, mas sem perder a precisão dos dados. Foi considerado que o quadricóptero possui uma estrutura rígida e simétrica com centro de massa e gravidade na posição central da estrutura e o torque desenvolvido pelas hélices será o quadrado da velocidade.

A partir da modelagem foi realizada a simulação dos controladores PID e Fuzzy (Capítulo 4). Com base nessas simulações que os controladores foram analisados, testados e ajustados para então serem colocados em prática no protótipo.

Para avaliação da qualidade de cada controle elaborado, foi feito uso das ferramentas Matlab e Simulink, que forneceram uma plataforma com recursos necessários para os testes.

Com a representação matemática demonstrada na Seção 3.2.1 foi possível efetuar a criação do ambiente de simulação e inserir perturbações com o intuito de avaliar as respostas em diferentes condições. Em todo procedimento de teste foi elaborado uma representação gráfica com os comportamentos de rolagem, guinada e altitude, bem como da resposta do sistema de controle e entradas dos sensores inerciais.

Após as simulações dos controladores PID e Fuzzy, estes controles foram implementados no protótipo do quadricóptero e os experimentos foram realizados em ambiente real (Capítulo 5). Para a realização dos experimentos, o protótipo foi colocado em uma área externa com espaço suficiente para testes longos. Na questão de evitar possíveis avarias, os testes iniciais foram limitados em um grau de liberdade por teste, com a fixação dos eixos e gradualmente liberando cada eixo até chegar em quatro graus.

Os procedimentos de avaliação nos testes externos foram realizados através dos dados obtidos com os sensores inerciais e os dados plotados graficamente e também com uma análise empírica de comportamento e suas respostas de controle.

Por fim, o resultados das simulações e dos experimentos foram analisados e confrontados.

### 3.2.1 Cinemática

Um Drone tem seis eixos de liberdade, porém, com quatro rotores fixos é possível atuar em apenas quatro eixos, conhecidos como controles de atitude (PAULA, 2012), sendo eles:

- Arfagem  $\theta$  : Momento no eixo y;
- Rolagem  $\phi$  : Momento no eixo x;
- Guinada  $\psi$  : Momento no eixo z;
- Altitude  $z$  : Movimento no eixo Z.

Um modelo simplificado pode ser visto na Figura 31.

Os movimentos podem ser realizados coordenando a velocidade de cada rotor, como mostra a Tabela 4 que cada grupo de hélices gera um movimento da estrutura.

**Tabela 4 – Movimentos do Drone**

<b>Movimento</b>	<b>Relação de velocidade angular nas hélices</b>
Arfagem	$\omega_3 - \omega_1$
Rolagem	$\omega_4 - \omega_2$
Guinada	$\omega_1 + \omega_3 - \omega_2 - \omega_4$
Altitude	$\omega_1 + \omega_2 + \omega_3 + \omega_4$

Para o sistema de coordenadas para o Drone foi utilizado Newton-Euler, um sistema para transformação de coordenadas para obter a posição translacional e rotacional em um corpo genérico com seis graus de liberdade. É necessário realizar uma adaptação para o uso em Drones, definida por Bresciani (2008), como é vista a seguir.

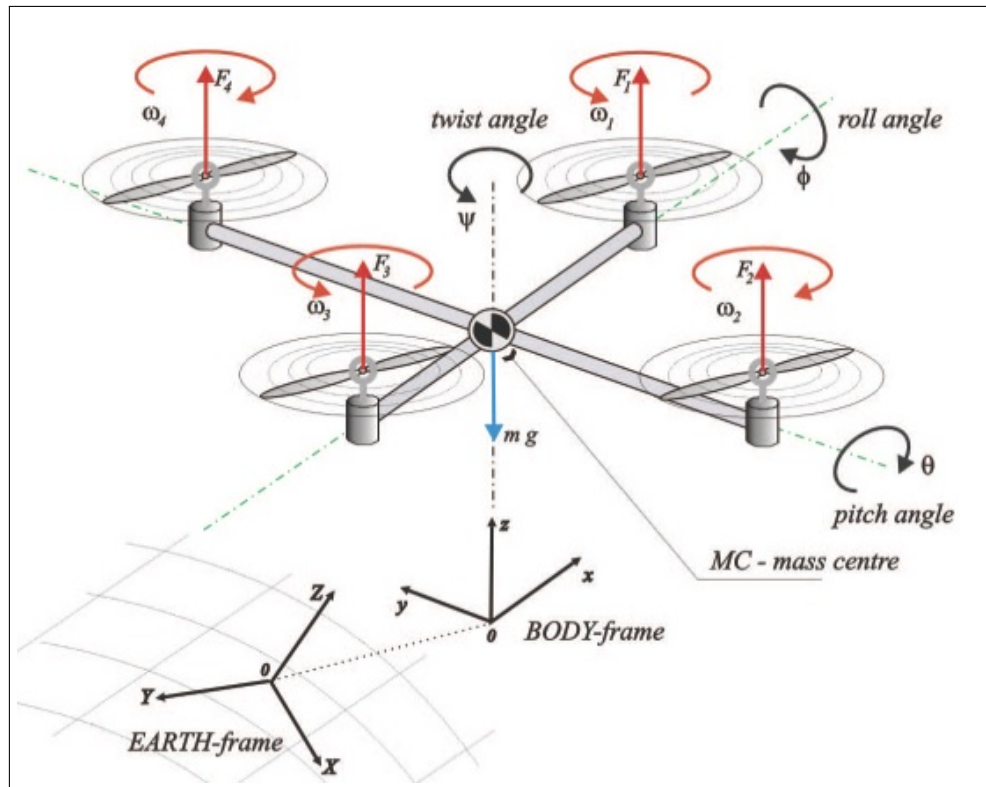


Figura 31 – Estrutura básica do funcionamento de um Drone

Fonte: Meyer et al. (2012).

Além dos movimentos é necessário um sistema de coordenadas, que serão duas: Referencial fixo a estrutura ( $Q$ ) e móvel em relação a Terra,  $x$ ,  $y$ ,  $z$ ; Referencial fixo em relação a Terra ( $E$ ),  $X$ ,  $Y$ ,  $Z$ .

As coordenadas em relação a estrutura serão definidas a partir do centro de massa como:

- Eixo  $x$ : Positivo na direção ao Motor 1 ( $M_1$ ) e negativo na direção ao Motor 3 ( $M_3$ );
- Eixo  $y$ : Positivo na direção ao Motor 4 ( $M_4$ ) e negativo na direção ao Motor 2 ( $M_2$ );
- Eixo  $z$ : Positivo sentido superior as hélices e negativo no inferior.

As coordenadas em relação a Terra serão definidas a partir do:

- Eixo  $X$ : Aponta sentido Norte da Terra;
- Eixo  $Y$ : Aponta sentido Oeste da Terra;
- Eixo  $Z$ : Aponta verticalmente a Terra.

Assim é possível definir posição linear em relação a Terra ( $\Gamma^E$ ) e angular ( $\Theta^E$ ) do Drone. A posição linear é definida por um vetor que aponta da Terra em direção a estrutura do Drone, visto na equação 12.

$$\Gamma^E = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (12)$$



A posição angular  $\Theta^E$  é obtida pela orientação da Terra em relação a estrutura do Drone, como mostra a equação 13.

$$\Theta^E = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \quad (13)$$

Em relação a velocidade linear do Drone ( $V^Q$ ) e angular ( $\omega^Q$ ), tem as equações (14) e (15).

$$V^Q = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad (14)$$

$$\omega^Q = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (15)$$

Combinando os vetores é possível obter posição generalizada da terra ( $\xi$ ) e a velocidade generalizada do Drone ( $v$ ), equações (16) e (17).

$$\xi = [\Gamma^E \quad \Theta^E]^T = [X \quad Y \quad Z \quad \phi \quad \theta \quad \psi]^T \quad (16)$$

$$v = [V^Q \quad \omega^Q]^T = [u \quad v \quad w \quad p \quad q \quad r]^T \quad (17)$$

É possível descrever como sistema  $Q$  se desloca de forma a sua orientação angular coincida com o sistema  $E$  com matriz de rotação ( $R_\Theta$ ) definido pela multiplicação de três matrizes, equações (18), (19), (20), e finalmente  $R_\Theta$  na equação (21)

$$\mathbf{R}(\phi, x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix} \quad (18)$$

$$\mathbf{R}(\theta, y) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \quad (19)$$

$$\mathbf{R}(\psi, z) = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (20)$$

$$\mathbf{R}_\Theta = \mathbf{R}(\phi, x)\mathbf{R}(\theta, y)\mathbf{R}(\psi, z) = \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\phi s_\theta s_\psi & s_\psi s_\phi + c_\phi s_\theta c_\psi \\ s_\psi c_\theta & c_\psi c_\phi + s_\psi s_\theta s_\psi & -c_\psi s_\phi + s_\psi s_\theta c_\psi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \quad (21)$$

A relação entre a velocidade linear de  $Q$  com a de  $E$ , podem ser relacionadas com a matriz de rotação como mostra equação (22).

$$\dot{\Gamma}^E = \mathbf{R}_\Theta \mathbf{V}^Q \quad (22)$$

Da mesma forma é possível relacionar a velocidade angular de  $Q$  com a de  $E$ , que pode ser chamado  $\dot{\Theta}^E$  Taxa de Euler, através da matriz de transferência ( $T_{\Theta}$ ). As equações (23) e (24) demonstram.

$$\omega^Q = \mathbf{T}_{\Theta}^{-1} \dot{\Theta}^E \quad (23)$$

$$\dot{\Theta}_E = \mathbf{T}_{\Theta} \omega^Q \quad (24)$$

A matriz de transferência  $\mathbf{T}_{\Theta}$  é determinada com a resolução das taxas de Euler  $\dot{\Theta}_E$ , como mostra as equações (25), (26) e (27).

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \mathbf{R}(\phi, x)^{-1} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \mathbf{R}(\phi, x)^{-1} \mathbf{R}(\theta, y)^{-1} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} = \mathbf{T}_{\Theta}^{-1} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (25)$$

$$\mathbf{T}_{\Theta}^{-1} = \begin{bmatrix} 1 & 0 & -\sin \theta \\ 0 & \cos \phi & \cos \theta \cos \phi \\ 0 & \sin \phi & \cos \theta \cos \phi \end{bmatrix} \quad (26)$$

$$\mathbf{T}_{\Theta} = \begin{bmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \frac{\sin \phi}{\cos \theta} & \frac{\cos \phi}{\cos \theta} \end{bmatrix} \quad (27)$$

Com uma matriz de generalização ( $J_{\Theta}$ ) e as equações (22) e (24), é possível fazer a transformação para uma equação equivalente entre a velocidade generalizada da terra  $\dot{\xi}$  e a velocidade generalizada do Drone  $v$ , como demonstra as equações (28) e (29).

$$\dot{\xi} = J_{\Theta} v \quad (28)$$

$$J_{\Theta} = \begin{bmatrix} R_{\Theta} & 0_{3 \times 3} \\ 0_{3 \times 3} & T_{\Theta} \end{bmatrix} \quad (29)$$

### 3.2.2 Dinâmica

Na dinâmica do Drone é necessário considerar a massa ( $m$ ) e matriz de inércia  $I$ , descritas na Equação(30), sendo  $I_{3 \times 3}$  matriz identidade 3 por 3,  $\tau^Q$  vetor de torque do Drone,  $\dot{V}^Q$  aceleração linear,  $\dot{\omega}^Q$  aceleração angular e  $F^Q$  o vetor de forças do Drone. Afim de facilitar a modelagem, a matriz de inércia sera considerada invariante no tempo.

$$\begin{bmatrix} mI_{3 \times 3} & 0_{3 \times 3} \\ 0_{3 \times 3} & I \end{bmatrix} \begin{bmatrix} \dot{V}^Q \\ \dot{\omega}^Q \end{bmatrix} + \begin{bmatrix} \omega^Q \times (mV^Q) \\ \omega^Q \times (I\omega^Q) \end{bmatrix} = \begin{bmatrix} F^Q \\ \tau^Q \end{bmatrix} \quad (30)$$

Bresciani (2008) demonstra que passando a Equação (30) para a forma de matriz, é obtido a equação (31). Em que  $\dot{v}$  é a aceleração generalizada,  $C_B(v)$  é o a aceleração centrípeta de *Coriolis* e  $\Lambda$  o vetor generalizado de forças sobre o corpo. A matriz inercial é descrita na Equação (32).

$$M_Q \dot{v} + C_Q(v) = \Lambda \quad (31)$$

$$M_Q = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{XX} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{YY} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{ZZ} \end{bmatrix} \quad (32)$$

Por ser considerada a estrutura rígida e simétrica, a matriz de inércia é diagonal e constante. A aceleração centrípeta de *Coriolis* é descrita na Equação (33), em que  $S(k)$  o operador de matriz anti-simétrica.

$$C_Q(v) = \begin{bmatrix} 0_{3 \times 3} & -mS(V^Q) \\ 0_{3 \times 3} & -S(I\omega^Q) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & mw & -mv \\ 0 & 0 & 0 & -mw & 0 & mu \\ 0 & 0 & 0 & mv & -mu & 0 \\ 0 & 0 & 0 & 0 & I_{ZZ}r & I_{YY}q \\ 0 & 0 & 0 & -I_{ZZ}r & 0 & I_{XX}p \\ 0 & 0 & 0 & I_{YY}q & -I_{YY}p & 0 \end{bmatrix} \quad (33)$$

Sendo  $k$  um vetor com valor genérico de três dimensões,  $k = [k_1 \ k_2 \ k_3]^T$ , a matriz antissimétrica  $S(k)$  é definida como mostra a Equação (34).

$$S(k) = -S^T(k) = \begin{bmatrix} 0 & -k_3 & k_1 \\ k_3 & 0 & -k_1 \\ -k_2 & k_1 & 0 \end{bmatrix} \quad (34)$$

A Equação (31) é genérica para corpos rígidos que obedecem as restrições impostas na facilitação de cálculos. Especificamente no Drone, ela pode ser dividida em três componentes de forças atuantes, cada uma caracterizada por sua natureza: gravitacional, efeitos giroscópicos e torques produzidos diretamente pelas hélices.

O vetor de força gravitacional ( $G_Q(\xi)$ ) é devido a aceleração da gravidade  $g$ , mostrado na Equação (35), onde  $F_G^Q$  é o vetor de força gravitacional em  $Q$ ,  $F_G^E$  vetor gravitacional de  $E$  e  $R_\Theta$  a matriz ortogonal normalizada, sendo sua inversa igual a transposta.

$$G_Q(\xi) = \begin{bmatrix} F_G^Q \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} R_\Theta^{-1} F_G^E \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} R_\Theta^T \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} mg \sin(\theta) \\ -mg \cos(\theta) \sin(\phi) \\ -mg \cos(\theta) \sin(\phi) \\ 0 \end{bmatrix} \quad (35)$$

Outra componente de força está relacionada aos efeitos giroscópicos produzidos pela rotação das hélices. As hélices estão rotacionando duas para o sentido horário e duas para o anti-horário. Se os ângulos de rolagem e arfagem forem zero, a diferença entre as somas de velocidades entre os sentidos das hélices gera um torque giroscópico, como demonstra a Equação (37).

$$O_Q(v)\Omega = \left[ \begin{array}{c} 0_{3 \times 1} \\ -\sum_{k=1}^4 J_{TP} \left( \omega^Q \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) (-1)^k \Omega_k \end{array} \right] = \left[ \begin{array}{c} 0_{3 \times 1} \\ J_{TP} \begin{bmatrix} -q \\ p \\ 0 \end{bmatrix} \end{array} \right] \Omega = J_{TP} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ q & -q & q & -q \\ -p & p & -p & p \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (37)$$

Sendo,  $O_Q(v)$  a matriz giroscópica da hélice,  $J_{TP}$  o momento inercial de rotação em torno do eixo da hélice,  $\Omega$  é a soma algébrica das velocidades de cada hélice  $\Omega_k$  na Equação (38).

$$\Omega = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4, \quad \Omega = \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \\ \Omega_4 \end{bmatrix} \quad (38)$$

A terceira componente de força, torques produzidos pelas hélices, é obtida através do quadrado da velocidade de rotação das hélices. A matriz de movimento  $E_Q$  multiplicada pelo vetor de velocidade ao quadrado, pode ser obtido o vetor de movimento ( $U_Q$ ). Conforme Equação (39), sendo os vetores de movimento:  $U_1$  altitude,  $U_2$ , rolagem,  $U_3$  arfagem,  $U_4$  guinada e  $b$  coeficiente de empuxo,  $d$  coeficiente de arrasto e  $l$  a distância entre hélices.

$$U_Q(\Omega) = E_Q \Omega^2 = \begin{bmatrix} 0 \\ 0 \\ U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ bl(\Omega_4^2 - \Omega_2^2) \\ bl(\Omega_3^2 - \Omega_1^2) \\ d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2) \end{bmatrix} \quad (39)$$

E finalmente a matriz de constantes para o movimento  $E_Q$ , na Equação (40).

$$E_Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ b & b & b & b \\ 0 & -bl & 0 & bl \\ -bl & 0 & bl & 0 \\ -d & d & -d & d \end{bmatrix} \quad (40)$$

Assim é possível descrever a dinâmica do Drone pelas Equação (41) e reorganizar em função da velocidade, Equação (42). Para facilitar, podem também ser colocados na forma de

sistemas de equações, como mostram as equações (43) e (44).

$$M_Q \dot{\mathbf{v}} + C_Q(\mathbf{v}) \mathbf{v} = G_Q(\xi) + O_Q(\mathbf{v}) \Omega + E_Q \Omega^2 \quad (41)$$

$$\dot{\mathbf{v}} = M_Q \dot{\mathbf{v}}^{-1} (-C_Q(\mathbf{v}) \mathbf{v} + G_Q(\xi) + O_Q(\mathbf{v}) \Omega + E_Q \Omega^2) \quad (42)$$

$$\left\{ \begin{array}{l} \ddot{X} = (\sin \psi \sin \phi + \cos \psi \sin \theta \cos \phi) \frac{U_1}{m} \\ \ddot{Y} = (-\cos \psi \sin \phi + \sin \psi \sin \theta \cos \phi) \frac{U_1}{m} \\ \ddot{Z} = (-g + \cos \theta \cos \phi) \frac{U_1}{m} \\ \dot{p} = \frac{I_{YY} - I_{ZZ}}{I_{XX}} qr - \frac{J_{TP}}{I_{XX}} q \Omega + \frac{U_2}{I_{XX}} \\ \dot{q} = \frac{I_{ZZ} - I_{XX}}{I_{YY}} qr - \frac{J_{TP}}{I_{YY}} q \Omega + \frac{U_3}{I_{YY}} \\ \dot{r} = \frac{I_{XX} - I_{YY}}{I_{ZZ}} qr - \frac{U_4}{I_{ZZ}} \end{array} \right. \quad (43)$$

$$\left\{ \begin{array}{l} U_1 = b(-\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ U_2 = lb(-\Omega_2^2 + \Omega_4^2) \\ U_3 = lb(-\Omega_1^2 + \Omega_3^2) \\ U_4 = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \\ \Omega = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 \end{array} \right. \quad (44)$$

## 4 SIMULAÇÃO

A simulação foi essencial para o desenvolvimento do controlador neste trabalho, através dela é possível determinar o comportamento e desempenho aproximado de todo o sistema.

Toda a simulação foi realizada com o software Matlab e a sua ferramenta Simulink (MATHWORKS, 2013), com elas foram desenvolvidos os métodos para obtenção do modelo matemático aproximado e todos os testes para cada controlador.

### 4.1 MODELO DINÂMICO PARA A ESTRUTURA

Com os parâmetros do momento de inércia e peso da estrutura do protótipo, é possível utilizar a biblioteca do Simulink *Aerospace Blockset* que possui a ferramenta *3DOF Equation of Motion* como mostra a Figura 32. Seus parâmetros de entrada são as forças e momentos nos eixos  $x$ ,  $y$  e  $z$  da estrutura. Como saída, essencialmente é obtido a velocidade angular, ângulos de ataque e a matriz de direção.

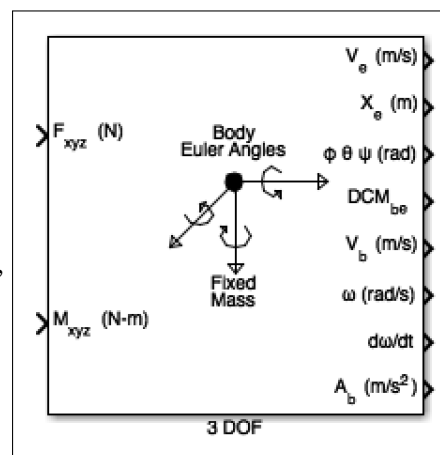


Figura 32 – Bloco 3DOF do Simulink.

### 4.2 MODELO DINÂMICO PARA OS MOTORES E ESCS

O modelo matemático de cada motor foi obtido por experimentação. Para tal foi desenvolvido um código no microcontrolador Tiva C para amostragem de velocidade angular nos motores. A rotina implementada está descrito no Apêndice C. Um dos problemas ao obter a velocidade está com o tempo necessário de amostragem a fim de obter sua resposta de transição e assim seu modelo matemático. Para tal foi utilizado um sensor de efeito Hall ligado ao microcontrolador.

Por se tratar de um motor *outrunner* utilizado neste trabalho, Figura 24, seus ímãs ficam ao redor do estator, o que facilita a obtenção da velocidade de giro através de um sensor de efeito Hall (KIM et al., 2011). Cada passagem de um dos ímãs pelo sensor é obtido um pulso de tempo proporcional a distância percorrida pelo mesmo. Levando como exemplo motor com 14 ímãs proporcionalmente distribuídos, é possível obter a velocidade com apenas 1/7 do tempo total para completar um giro, que é o tempo entre a passagem de um ímã para outro.

Programando o microcontrolador para ativar o motor através de PWM com um ESC, é possível obter uma relação entre entrada (PWM) e saída (velocidade angular). Excitando o motor com vários valores PWM e obtendo sua saída relacionada, pode ser obtido um modelo matemático do conjunto ESC, motor e hélice. O Matlab possui ferramentas para identificação de funções de transferência. Aqui foi utilizado a estrutura ARX para obtenção do modelo. O ARX é um modelo polinomial e seus parâmetros são identificados pelo método dos mínimos quadrados (KON et al., 2013).

Para facilitar a obtenção do modelo foi elaborado uma rotina no Matlab, descrito no Apêndice B, para ser alimentado com os dados obtidos pelo microcontrolador e sensor de efeito hall, onde ao final gera uma função de transferência estimada para cada motor.

### 4.3 MODELO DE BLOCOS DA SIMULAÇÃO

A simulação pode ser dividida em quatro partes: controle, misturador, forças atuantes e planta, como demonstrado na Figura 33 para o PID e Figura 34 para o fuzzy.

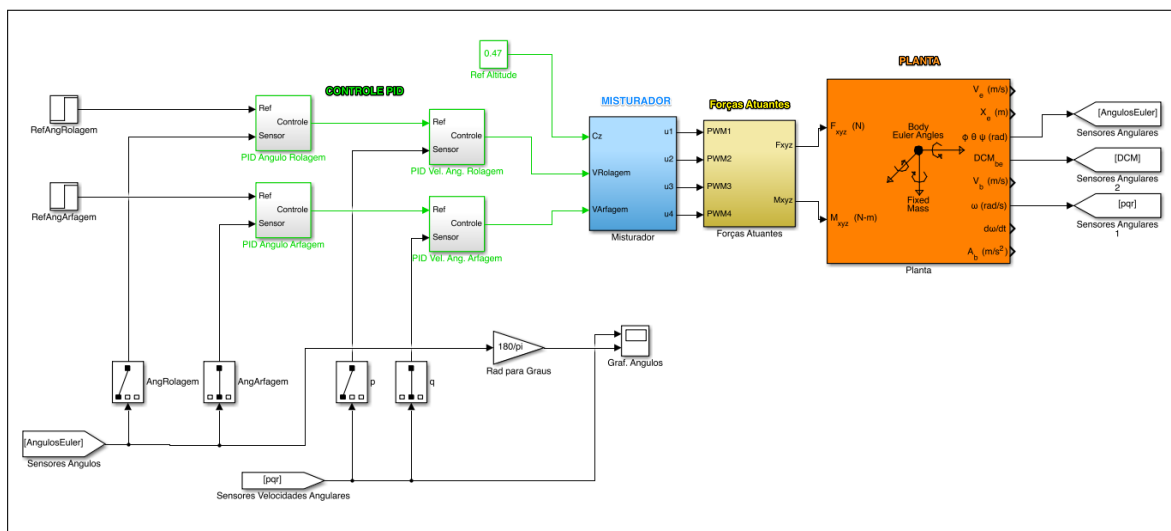


Figura 33 – Blocos para simulação PID.

#### 4.3.1 Bloco de Controle PID e Fuzzy

Neste bloco ficam os controladores para o sistema. O controle é feito em dois estágios, primeiramente controlando a velocidade angular e somente depois é controlado o ângulo de ataque, formando assim um controle em cadeia. Separando o sistema desta maneira torna-se mais simples a elaboração dos controles que diretamente elaborar o controle dos ângulos.

Na Figura 33 é possível ver o controle utilizando três PID para controles das velocidades angulares  $p, q, r$  e mais três controles para os ângulos em relação a estrutura  $\theta, \phi$  e  $\psi$ . Na Figura 34 mostra o sistema de controle fuzzy.

#### 4.3.2 Bloco de Mistura

Neste bloco são associado os sinais de controles para cada ângulo, demonstrado na Figura 35, a fim de gerar um PWM para os quatro motores que atuam para movimento no

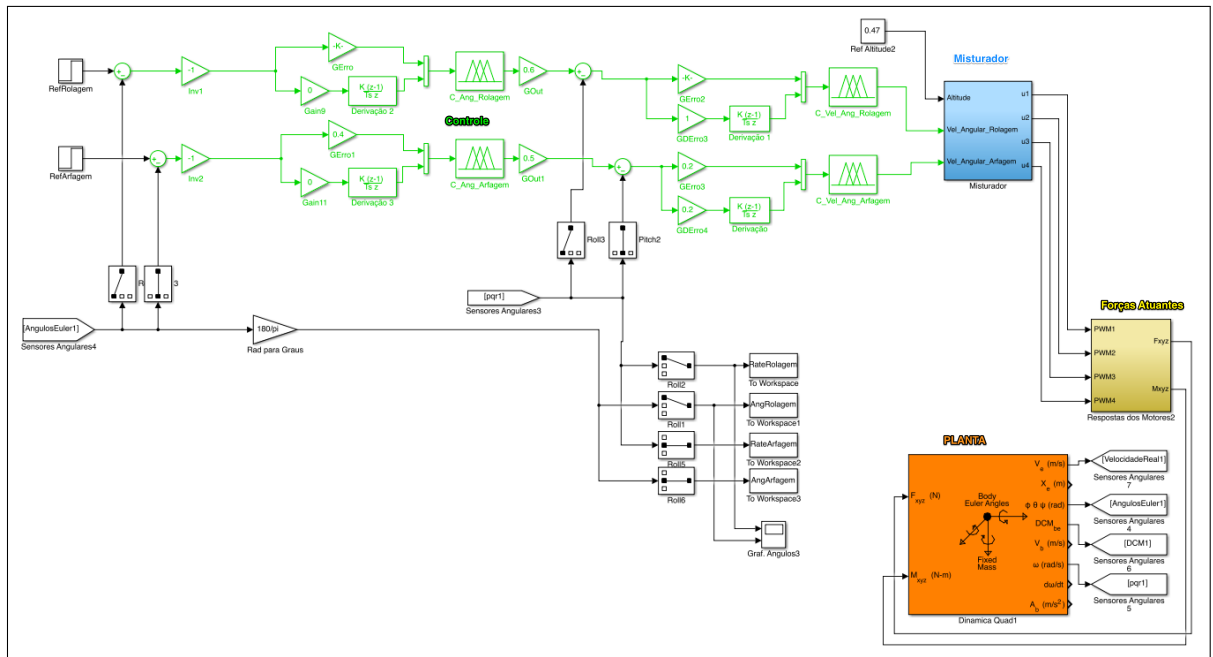


Figura 34 – Blocos para simulação Fuzzy.

sentido correto. Este bloco é baseado com a resposta definidas no sistema de Equações 44.

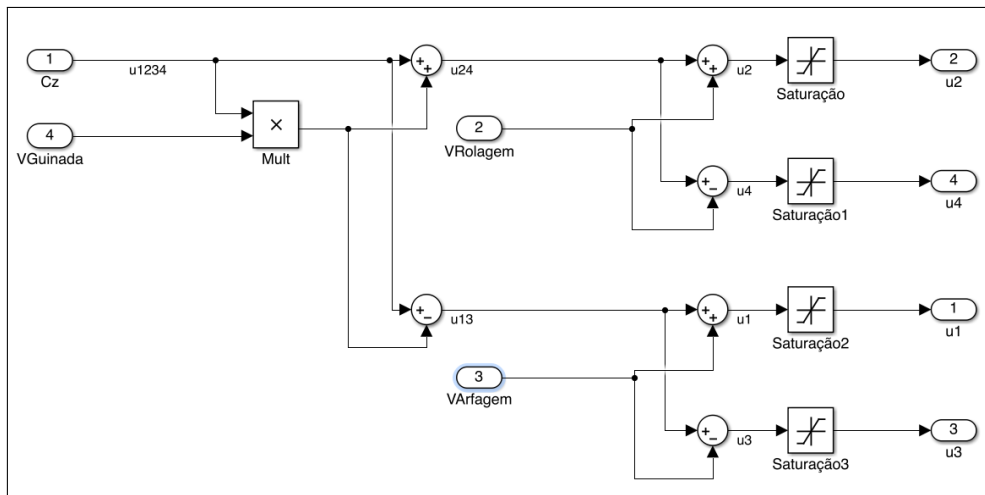


Figura 35 – Bloco do Misturador.

### 4.3.3 Bloco de Forças Atuantes

No bloco de forças atuantes, demonstrado na Figura 36, é feita a resposta produzida pelos PWM em cada motor, gerando uma força de empuxo  $U_1$  e os momentos  $U_2$ ,  $U_3$  e  $U_4$ , obedecendo ao sistema de Equações 44. E finalmente com o DCM é definido a atuação da força peso no sistema, baseado na direção em que se encontra.

### 4.3.4 Bloco de Resposta dos Motores

A resposta dos motores é obtida pela função de transferência na modelagem experimental, ao utilizar o método ARX, todas as respostas lineares são necessariamente removidas



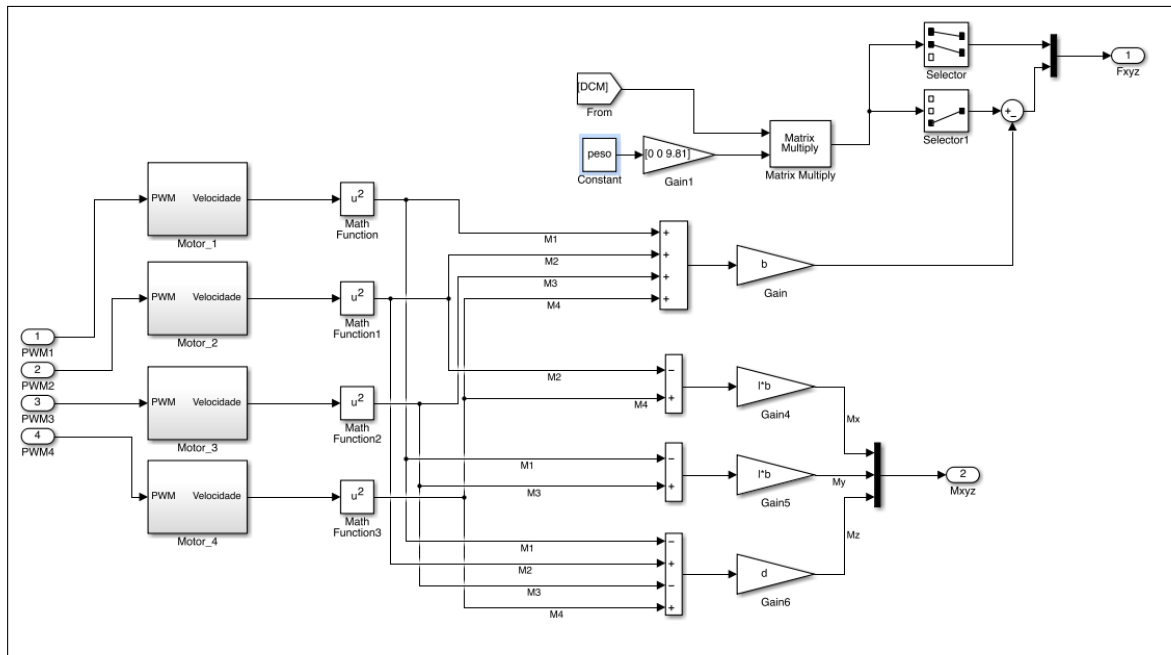


Figura 36 – Bloco de Forças Atuantes.

antes da identificação, gerando assim um valor constante de entrada e saída que necessita ser acrescentada ao sistema, junto com a função de transferência, como mostra a Figura 37.

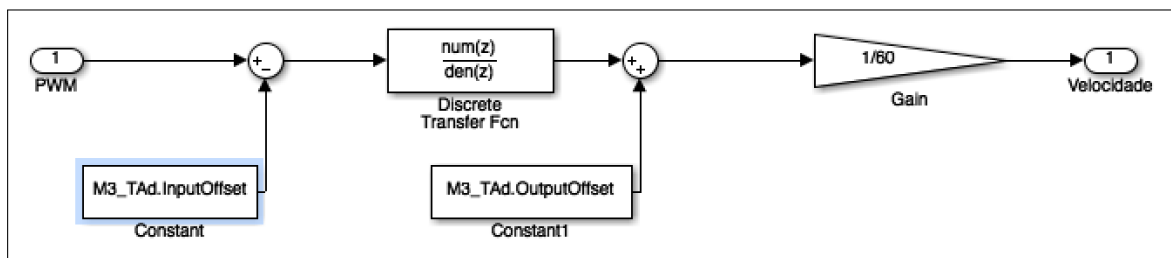


Figura 37 – Bloco de Resposta dos Motores.

#### 4.4 IDENTIFICAÇÃO DA CURVA DE VELOCIDADE E FORÇA

O motor *brushless DC* foi acionado com o ESC da Turnigy, seu funcionamento é microprocessado, tendo como entrada um sinal PWM com *Duty Cycle* variando de  $1000\mu s$  até  $2000\mu s$ . Porém, essa variação não foi necessariamente proporcional com a velocidade de zero até o máximo de rotação do motor de forma linear, como pode ser verificado na Figura 38.

Para facilitar a modelagem, é utilizada apenas a região linear da curva de velocidade, que foi escolhido de  $1040\mu s$  até  $1770\mu s$ , como demonstrado na Figura 39. As funções que representam estas regiões podem ser vistas na Equação 45, para conseguir uma maior acurácia, o polinômio utilizado foi de quinto grau.

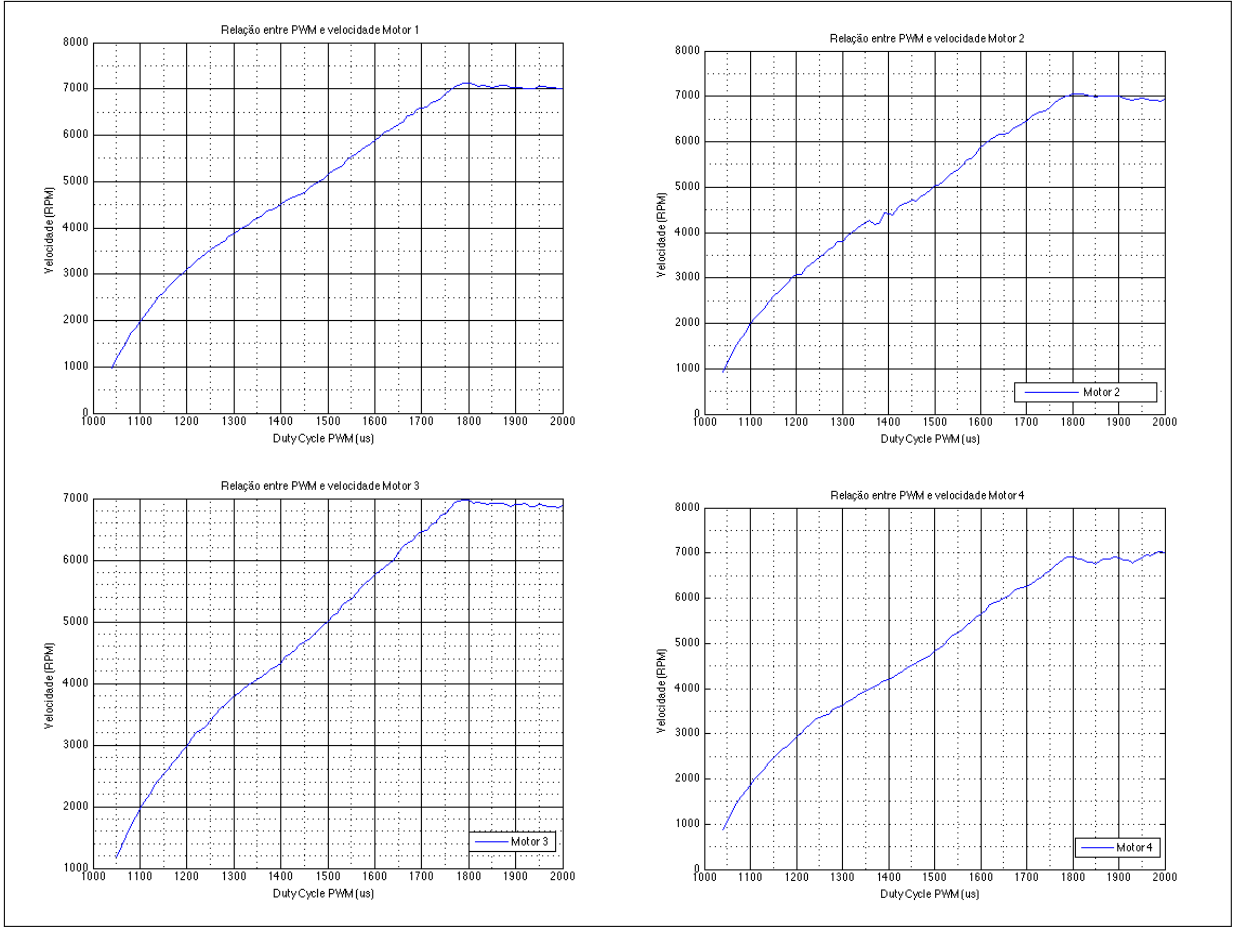


Figura 38 – Relação entre PWM e Velocidade nos Motores 1, 2, 3 e 4 respectivamente.

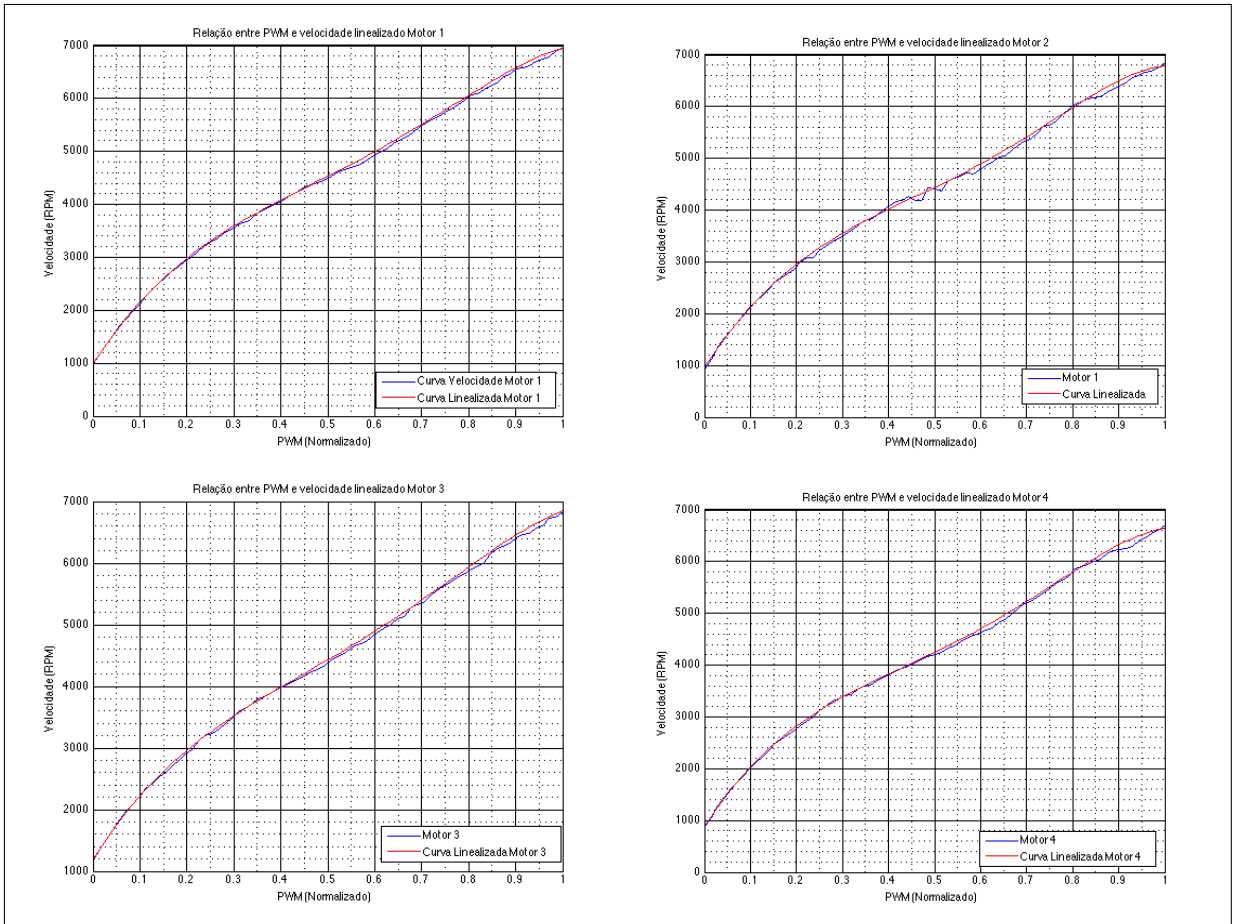


Figura 39 – Região linear de cada motor.

$$\begin{cases} V_{M1} = -4404,2x^5 - 2286,3x^4 + 22346,5x^3 - 23474,7x^2 + 13781,6x + 991,8337 \\ V_{M2} = -12926,4x^5 - 16336,8x^4 + 8909,4x^3 - 19924,5x^2 + 13421x + 975,0618 \\ V_{M3} = -4137,6x^5 - 1296,9x^4 + 18694,6x^3 - 19639,7x^2 + 12030,9x + 1195,9 \\ V_{M4} = -7367,5x^5 + 1666,2x^4 + 22520,5x^3 - 24781,9x^2 + 13706,6x + 893,4 \end{cases} \quad (45)$$

O empuxo dos motores foi aferido utilizando a balança SF-400. Todos os motores tiveram uma relação velocidade com empuxo muito próximos, como mostra a Figura 40. O coeficiente de empuxo  $b$  foi obtido no ponto onde a força de empuxo, somado dos quatro motores no quadricóptero, sustenta seu próprio peso, aproximadamente 255g em cada motor, tendo a constante  $b$  igual a  $613,44 \cdot 10^{-6}$ , o qual ficou com a curvatura próxima das forças aferidas nos motores.

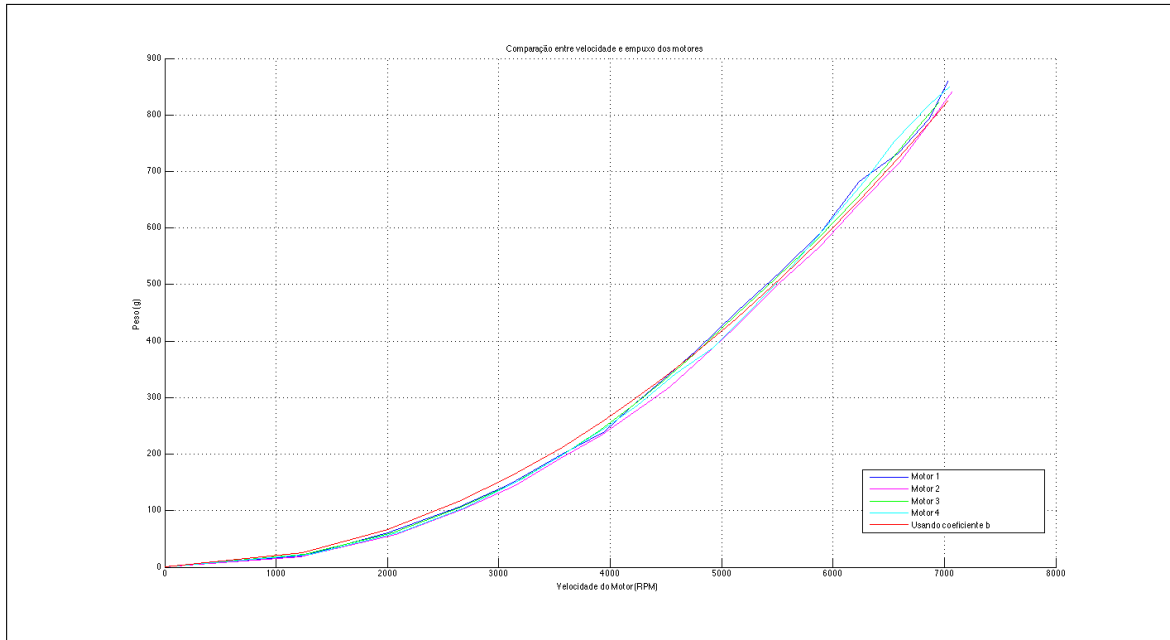
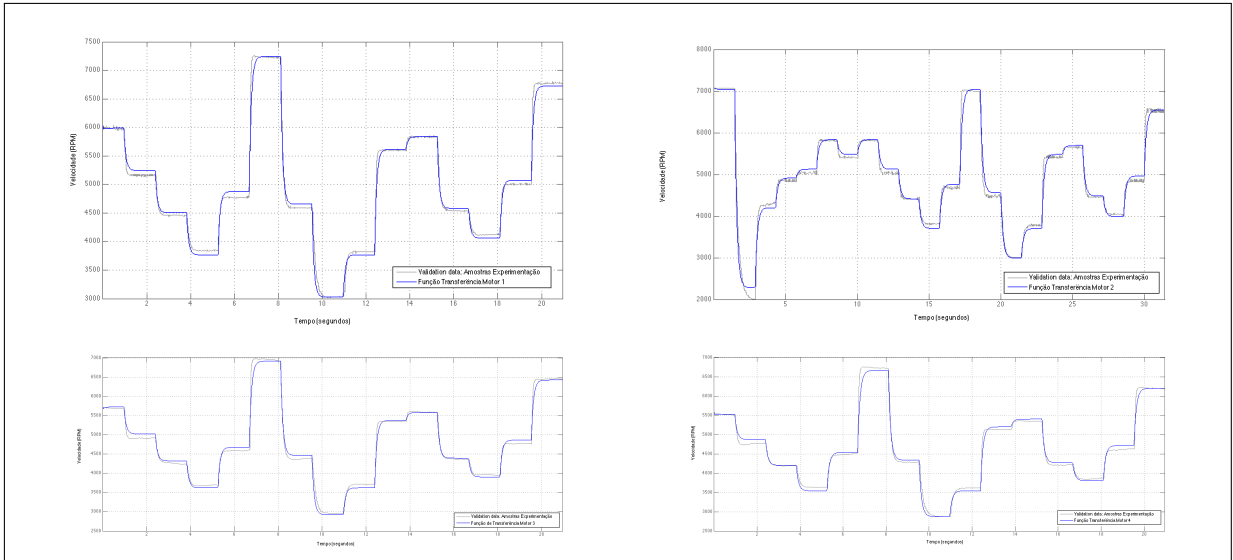


Figura 40 – Empuxo dos Motores e Coeficiente de Empuxo.

#### 4.5 IDENTIFICAÇÃO DO MODELO MATEMÁTICO ESC/MOTOR/HÉLICE

Para identificação do modelo foram elaborado testes com várias mudanças de velocidades, o código utilizado está no Apêndice C. A função de transferência discreta identificada pelo método ARX de cada motor poder ser observada nas equações 46, com acerto de 91,28%, 88,03%, 90,37% e 89,44% para os conjuntos motor/ESC/hélice um, dois, três e quatro respectivamente. As comparações podem ser vistas na Figura 41. Esse modelo tem a entrada um PWM normalizado e saída a velocidade do motor em RPM.

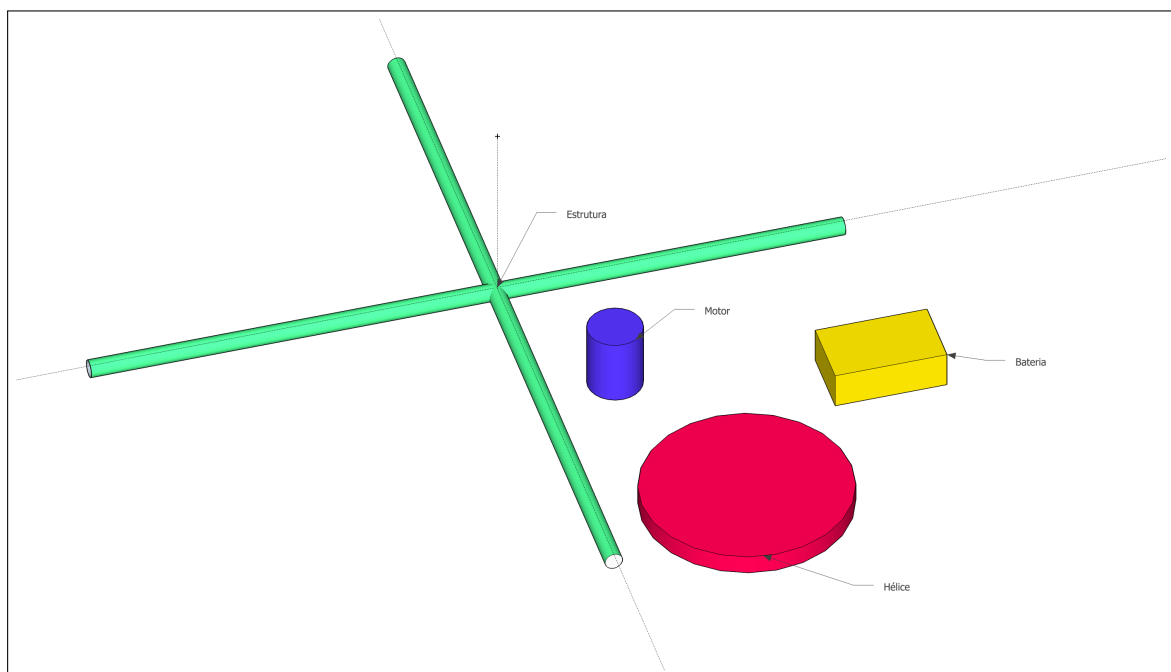
$$\begin{cases} \frac{\Omega_1}{PWM} = \frac{-0,03156z^{-4} + 87,37z^{-5}}{1 - 0,947z^{-1} - 0,0368z^{-2}} \\ \frac{\Omega_2}{PWM} = \frac{-1,786z^{-1} + 54,1z^{-2}}{1 - 0,9593z^{-1} - 1,273z^{-2} - 0,03065z^{-3}} \\ \frac{\Omega_3}{PWM} = \frac{0,8203z^{-4} + 75,88z^{-5}}{1 - 0,9508z^{-1} - 0,03421z^{-2}} \\ \frac{\Omega_4}{PWM} = \frac{-1,021z^{-4} + 67,57z^{-5}}{1 - 0,9524z^{-1} - 0,03384z^{-2}} \end{cases} \quad (46)$$



**Figura 41 – Comparação entre a função de transferência e valores reais.**

#### 4.6 IDENTIFICAÇÃO DO MOMENTO DE INÉRCIA

O momento de inércia do protótipo,  $I_{xx}$ ,  $I_{yy}$  e  $I_{zz}$  foi separado em quatro partes para facilitar os cálculos: estrutura, motores, hélices e bateria. Cada componente representado por uma figura geométrica, como demonstra a Figura 42. Considerando o protótipo totalmente simétrico, o centro de massa está localizado na posição (0,0,0) de  $xyz$  na Figura 31.



**Figura 42 – Modelos geométricos para cálculo do momento de inércia .**

A soma dos momentos de inércia da estrutura, bateria e as quatro hélices com os mo-

tores resultaram na matriz da Equação 47.

$$\mathbf{I} = \begin{bmatrix} 9.89 & 0 & 0 \\ 0 & 10.32 & 0 \\ 0 & 0 & 18.11 \end{bmatrix} * 10^{-3} [Nms^2] \quad (47)$$

Com esses dados é possível realizar as simulações, os resultados das mesmas e comparações com os dados em ambiente real é apresentado no Seção 5.

## 5 EXPERIMENTOS E RESULTADOS

Neste capítulo identificado e apresentado os parâmetros dos controladores PID e Fuzzy. Com esses dados foi feito na Seção 5.1 a simulação em um ambiente ideal e com distúrbios, verificando assim o comportamento do controlador. Após verificada a estabilidade, foi implementado os controladores no protótipo e então realizado testes em um ambiente real. Estes dados estão apresentados na Seção 5.2.

### 5.1 SIMULAÇÃO

Com os parâmetros e o ambiente de simulação definidos no Capítulo 4, foi elaborado os controladores PID, apresentados na Subseção 5.1.1 e Fuzzy, na Subseção 5.1.2. Os controladores, tanto para o PID como para o Fuzzy, estão definidos em dois níveis: controle de velocidade angular e ligado em cascata o controlador para posicionamento de ângulo de ataque.

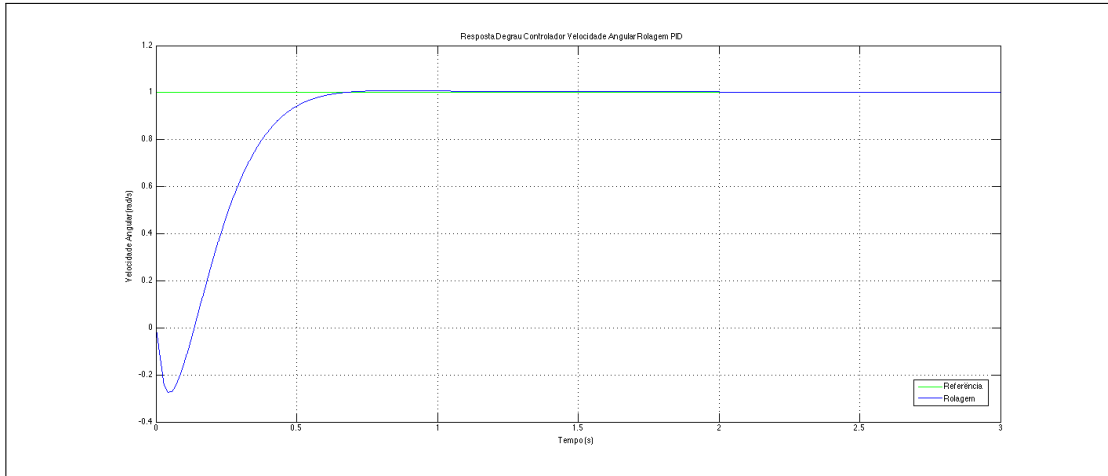
#### 5.1.1 Controladores PID

Através das ferramentas do simulink foi obtido uma aproximação das variáveis de controle  $K_p$ ,  $K_i$  e  $K_d$ , para o modelo desenvolvido. Com um pré controle estabelecido, foi feito o ajuste fino até obter os controles de velocidade angulares aceitáveis para cada eixo, neste trabalho é considerado com sobressinal abaixo de 10 % e tempo de estabilização menor que um segundo.

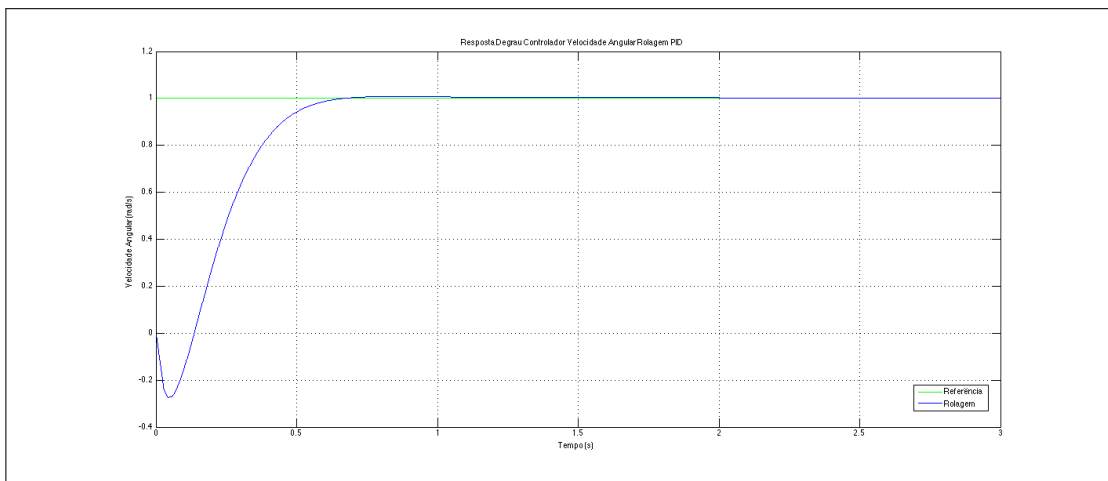
O controlador de velocidade angular obtido é descrito nas equações 48, e 51. O tempo de resposta obtido nos controladores de Rolagem e Arfagem foram de 0,3s, uma resposta rápida o suficiente para um controle eficiente. A resposta transitória do controlador são mostradas nas figuras 45 e 46, ambas mostram que o controlador não possui um sobressinal. Os parâmetros finais foram  $K_p = 4.08e - 05$ ,  $K_i = 3.18e - 07$ ,  $K_d = 0.000506$ ,  $T_f = 0.478$  e  $T_s = 0.0143$ .

$$C_p = \frac{0.001099z^2 - 0.002196z + 0.001098}{z^2 - 1.97z + 0.9701} \quad (48)$$

$$C_q = \frac{0.001099z^2 - 0.002196z + 0.001098}{z^2 - 1.97z + 0.9701} \quad (49)$$



**Figura 43 – Controle para Velocidade Angular de Rolagem.**

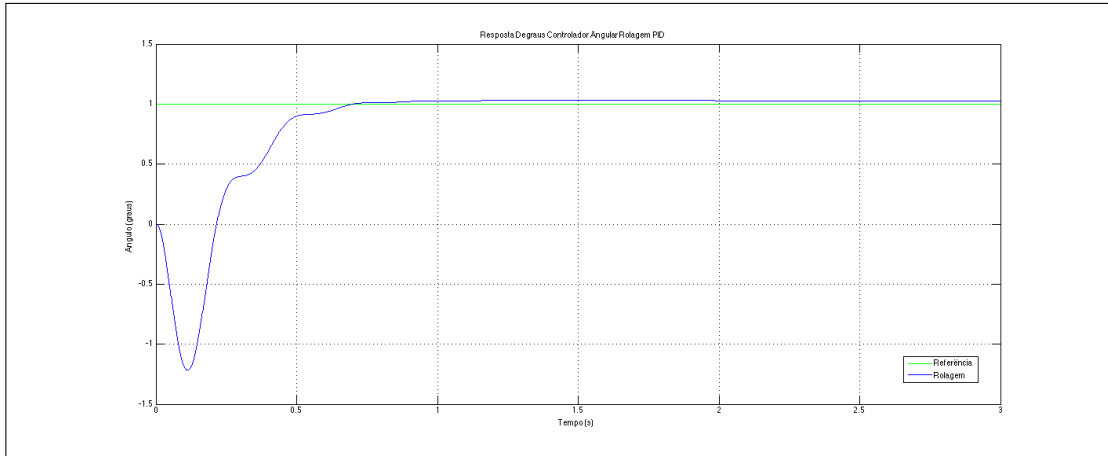


**Figura 44 – Controle para Velocidade Angular de Arfagem.**

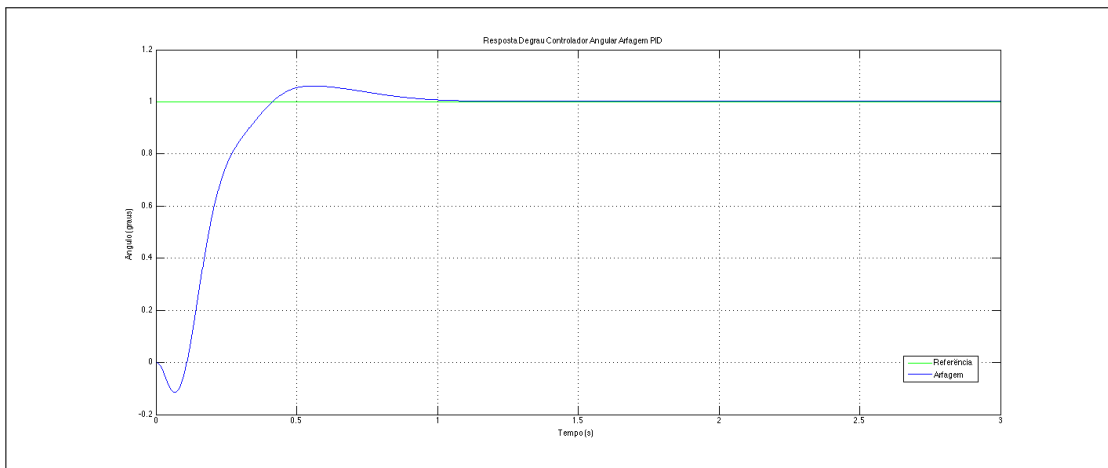
O controlador para a posição angular fica em cascata com o de velocidade angular, como visto na Seção 4.3.1. Através do Simulink, foi obtido os controladores das equações 50 e 51. Os parâmetros finais foram  $K_p = 1.17$ ,  $K_i = 0.0103$ ,  $K_d = -0.661$ ,  $T_f = 0.71$  e  $T_s = 0.0143$ .

$$C_r = \frac{0.2418z^2 - 0.4598z + 0.2181}{z^2 - 1.98z + 0.9799} \quad (50)$$

$$C_p = \frac{0.2418z^2 - 0.4598z + 0.2181}{z^2 - 1.98z + 0.9799} \quad (51)$$



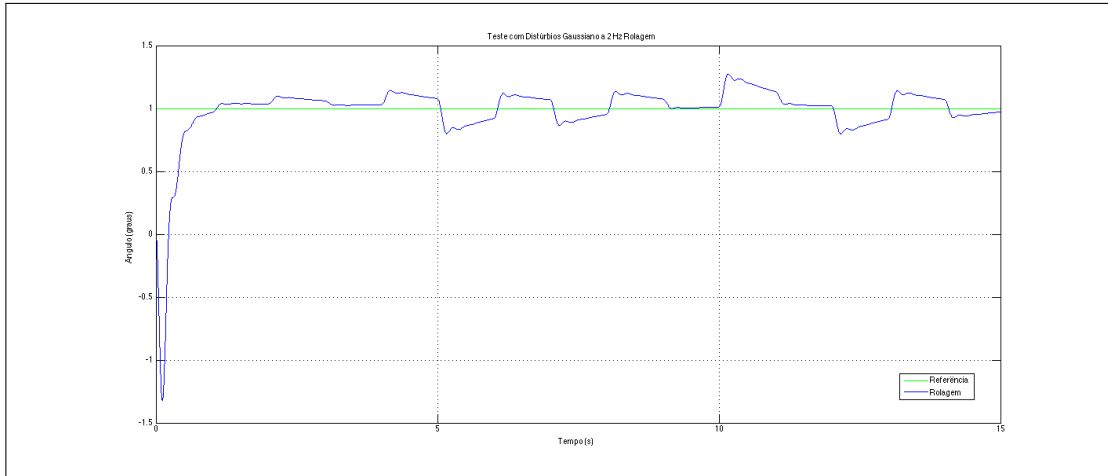
**Figura 45 – Controle para Ângulo de Rolagem.**



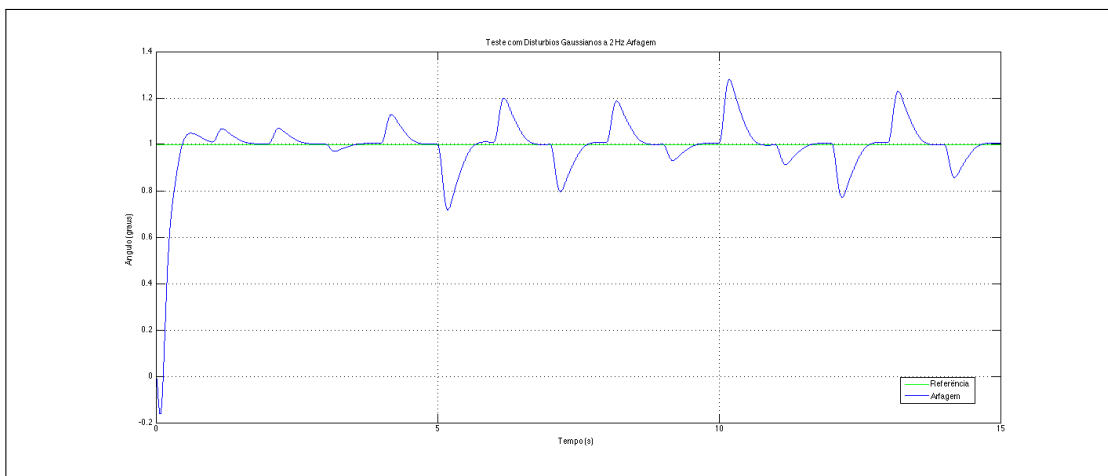
**Figura 46 – Controle para Ângulo de Arfagem.**

Para confirmar a resposta dos controladores, foi inserido um distúrbio na força de momento em cada eixo. Utilizando a referência de  $1^0$  no ângulo de rolagem e arfagem para estabilização, foi usado um gerador de ruídos Gaussiano a uma taxa  $2\text{ Hz}$  e com uma variância de  $10^{-6}$  para gerar as forças de perturbação. As respostas obtidas podem ser analisadas nas figuras 47 e 48.





**Figura 47 – Distúrbios de 2 Hz no eixo de Rolagem.**



**Figura 48 – Distúrbios de 2 Hz no eixo de Arfagem.**

Todos os distúrbios foram corrigidos pelos controladores dentro dos limites estipulados no início desta Seção. Para os controladores PID o teste em ambiente real está na Seção 5.2.

### 5.1.2 Controlador Fuzzy

Na elaboração do controlador fuzzy, foram utilizadas duas entradas, erro e variação do erro, e uma saída para controle. A diferenciação dos conjuntos fuzzy foram feitas com sete variáveis linguísticas em cada entrada e saída, definidas na Tabela 5. As regras foram elaboradas conforme mostra a Tabela 6.

Tabela 5 – Variáveis Linguísticas.

VARIÁVEL	TIPO	NÍVEIS
<b>EN<sub>x</sub></b>	ERRO NEGATIVO	1, 2 e 3
<b>EP<sub>x</sub></b>	ERRO POSITIVO	1, 2 e 3
<b>DN<sub>x</sub></b>	VARIAÇÃO ERRO NEGATIVO	1, 2 e 3
<b>DP<sub>x</sub></b>	VARIAÇÃO ERRO POSITIVO	1, 2 e 3
<b>TN<sub>x</sub></b>	TRAÇÃO NEGATIVA	1, 2 e 3
<b>TP<sub>x</sub></b>	TRAÇÃO POSITIVA	1, 2 e 3
<b>DZero</b>	SEM VARIAÇÃO DE ERRO	-
<b>EZero</b>	SEM ERRO	-
<b>ZERO</b>	SEM TRAÇÃO	-

Tabela 6 – Regras Fuzzy para controle angular.

AND	DN3	DN2	DN1	DZero	DP1	DP2	DP3
<b>EN3</b>	TP3	TP3	TP2	TP2	TP2	TP2	TP2
<b>EN2</b>	TP3	TP2	TP2	TP2	TP2	TP2	TP1
<b>EN1</b>	TP2	TP2	TP1	TP1	TP1	TP1	ZERO
<b>EZero</b>	TP2	TP1	TP1	ZERO	TP1	TP1	TN2
<b>EP1</b>	ZERO	TN1	TN1	ZERO	TN1	TN2	TN2
<b>EP2</b>	TN1	TN1	TN2	TN2	TN2	TN2	TN3
<b>EP3</b>	TN1	TN2	TN2	TN2	TN2	TN3	TN3

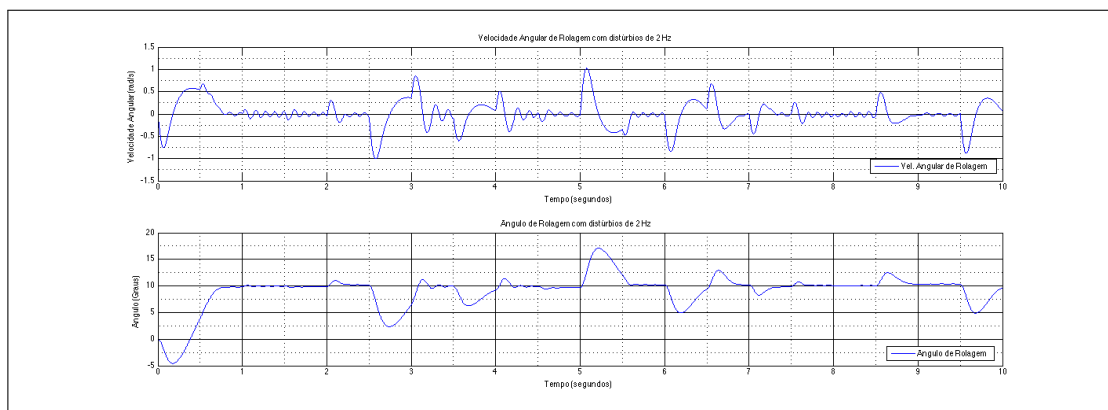
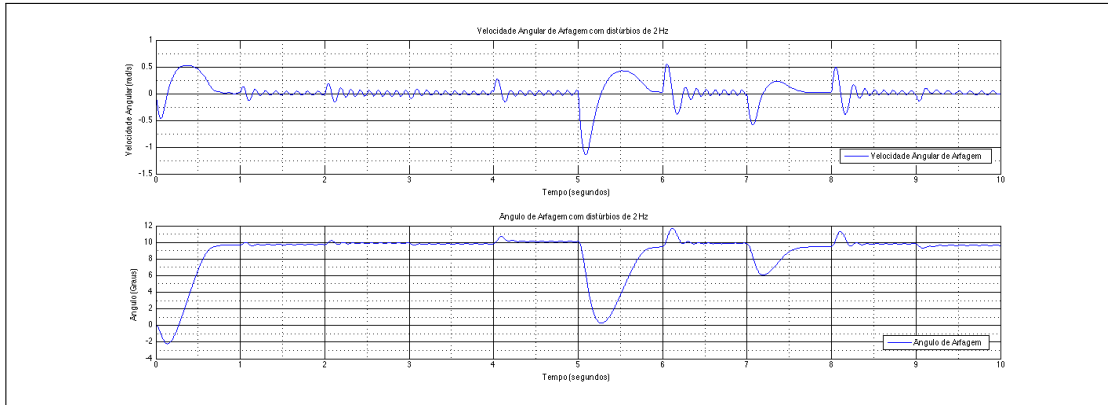


Figura 49 – Teste do controlador fuzzy Rolagem com distúrbios de 2 Hz.



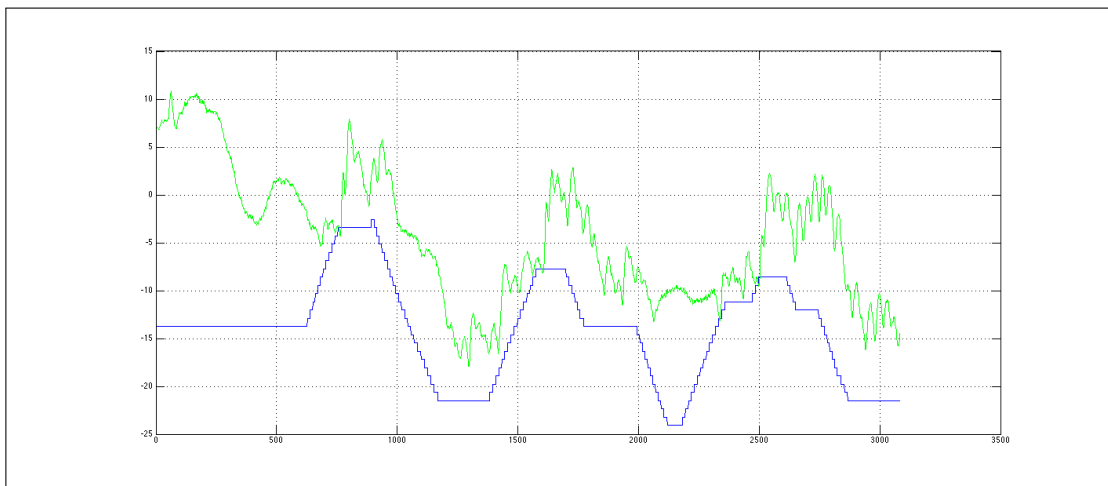
**Figura 50 – Teste do controlador fuzzy de Arfagem com distúrbios de 2 Hz.**

## 5.2 AMBIENTE REAL

No ambiente real o PID e o Fuzzy apresentaram diferenças, comparado com a resposta em simulação, como demonstra as subseções 5.2.1 e 5.2.2. Devido a vibração gerada pela rotação do motor, os sensores obtiveram ruídos significativos nas amostragens, mostrando ser necessário alguma implementação de filtro para amenizar. Os ruídos também afetaram os controladores, gerando algumas oscilações não esperadas.

### 5.2.1 Controladores PID

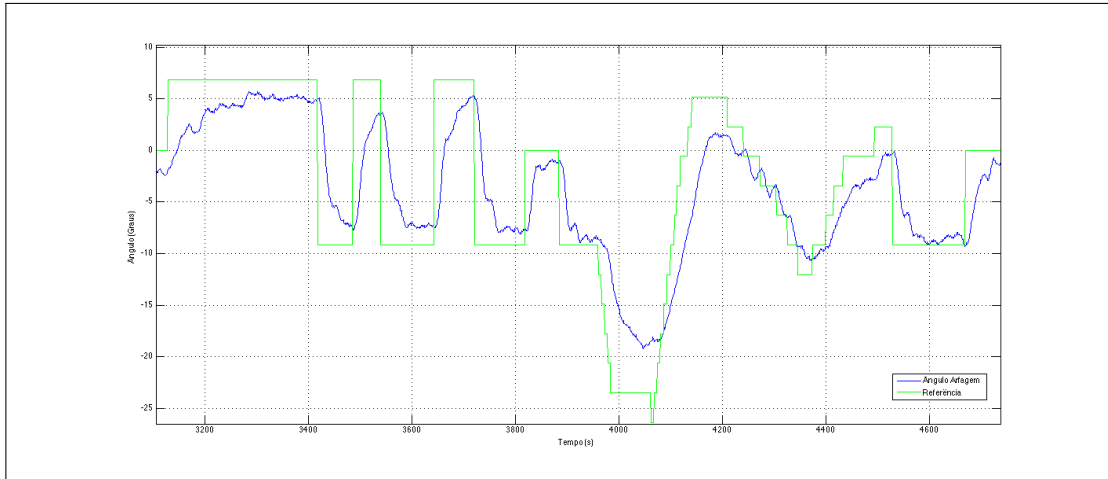
Controlador PID teve uma simulação de 15 segundos com referências aleatórias. Os dados obtidos podem ser analisados na Figura 51. O controlador PID em ambiente real não demonstrou muita estabilidade e um erro em relação a referência, o mesmo foi implementado exatamente como no ambiente de simulação, sem ajustes extras.



**Figura 51 – Resposta do controlador PID em ambiente real.**

### 5.2.2 Controladores Fuzzy

O controlador Fuzzy obteve uma resposta próxima da simulação, como mostra a Figura 52. Ao contrário do PID, os ruídos não interferiram significativamente no processo de controle.



**Figura 52 – Resposta do controlador Fuzzy em ambiente real.**

## 6 CONCLUSÃO

Este Trabalho de Conclusão de Curso propôs o estudo comparativo de técnicas para o controle de atitude em um quadricóptero. Os controles propostos foram o controle clássico com PID e a técnica de controle através da lógica fuzzy. A modelagem do sistema se mostrou complexa e de uma matemática extensa, necessitando estudos multidisciplinares para obtenção de um modelo válido.

No controlador PID foram necessários muitos ajustes até a obtenção de uma boa estabilidade do sistema na simulação. Em ambiente real mostrou os seus problemas quanto as não-linearidades do sistema, causando alguns distúrbios não previstos em simulação.

Já com o controlador Fuzzy os ajustes nas simulações foram menores, de forma mais amigável e intuitiva. Demonstrou em ambiente real uma maior similaridade com a simulação, mesmo com os problemas de não linearidade do sistema.

As principais diferenças entre as duas técnicas, foi a facilidade de implementação do Fuzzy, por este não necessitar um modelo tão exato como no PID e características mais próximas do modo de pensar humano. Porém o Fuzzy demonstrou alguns pequenos problemas pontuais de erros quanto a distúrbios com forças externas ao sistema. Estes mesmo problemas não afetaram tanto o PID, devido a sua parcela integrativa de correção do erro.

Com as simulações e teste em ambiente real, ambos conseguiram controlar a atitude do quadricóptero, sendo o Fuzzy mais fiel as simulações ante ao PID, devido as diferenças entre modelo matemático ideal usado na simulação e real na prática.

Como trabalhos futuros, é possível sugerir a implementação de um controlador LQR e um PID neural. Também testes mais elaborados em ambiente real, com controle de todos os eixos ao mesmo tempo. Para maior precisão, a implementação de um filtro de Kalman para os sensores inerciais ajudaria a evitar oscilações por ruídos de referência na posição em relação aos eixos do protótipo.

## REFERÊNCIAS

- BEALE, R.; JACKSON, T. **Neural Computing - An Introduction**. [S.l.]: Taylor & Francis, 2010. ISBN 9781420050431.
- BOUDERGUI, K.; CARREL, F.; DOMENECH, T.; GUENARD, N.; POLI, J. P.; RAVET, A.; SCHOEPFF, V.; WOO, R. Development of a drone equipped with optimized sensors for nuclear and radiological risk characterization. In: **Advancements in Nuclear Instrumentation Measurement Methods and their Applications (ANIMMA), 2011 2nd International Conference on**. [S.l.: s.n.], 2011. p. 1–9.
- BRESCIANI, T. **Modelling, Identification and Control of a Quadrotor Helicopter**. Lund Sweden: [s.n.], 2008. Disponível em: <<http://www.control.lth.se/documents/2008/5823.pdf>>.
- CALVO, R. **Arquitetura híbrida inteligente para navegação autônoma de robôs**. São Carlos: [s.n.], 2007. Disponível em: <<http://www.teses.usp.br/teses/disponiveis/55/55134/tde-15062007-161911/>>.
- CALVO, R. D.; MYRIAM, R.; HEITOR, L. S.; FIGUEIREDO, M. Indução de regras nebulosas via sistemas classificadores para o controle da arquitetura em redes neurais construtivas para navegação autônoma. In: **VIII Simpósio Brasileiro de Redes Neurais**. [S.l.: s.n.], 2004. p. ?–?
- CORIOLIS, P. M. G. Coriolis mémoire sur le principe des forces vives dans les mouvements relatifs. In: **Journal Ecole polytechnique (France)**. [S.l.: s.n.], 1896. p. 42.
- DALAMAGKIDIS, K.; VALAVANIS, K. P.; PIEGL, L. A. **On Integrating Unmanned Aircraft Systems into the National Airspace System: Issues, Challenges, Operational Restrictions, Certification, and Recommendations**. [S.l.]: Springer-Verlag, 2009. (Intelligent Systems, Control and Automation: Science and Engineering, v. 36).
- DELGADO, M. R. D. B. da S. **Projeto Automatico de Sistemas Nebulosos: Uma Abordagem Co-Evolutiva**. Tese (Doutorado) — FACULDADE DE ENGENHARIA ELETRICA E DE COMPUTACAO, UNIVERSIDADE ESTADUAL DE CAMPINAS, 26 February 2002. Disponível em: <[http://www.dca.fee.unicamp.br/~vonzuben/research/myriam\\_dout.html](http://www.dca.fee.unicamp.br/~vonzuben/research/myriam_dout.html)>.
- EDUARDO, G. d. P. **Neuro controlador otimo por algoritmos geneticos para multiplos sistemas ativos de dinamica veicular em guinada**. Tese (Doutorado) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2009. Disponível em: <<http://www.teses.usp.br/teses/disponiveis/18/18149/tde-17012011-145140/>>.
- GARDNER, J. W.; VARADAN, V. K. **Microsensors, Mems and Smart Devices**. New York, NY, USA: John Wiley & Sons, Inc., 2001. ISBN 047186109X.
- GOMIDE, F.; GUDWIN, R.; TANSCHKEIT, R. Conceitos fundamentais da teoria de conjuntos fuzzy, lógica fuzzy e aplicações. In: **Proc. 6 th IFSA Congress-Tutorials**. [S.l.: s.n.], 1995. p. 1–38.
- HANSELMAN, D. **Brushless permanent-magnet motor design**. [S.l.]: McGraw-Hill, 1994. ISBN 9780070260252.

- HAYKIN, S. **Redes Neurais - 2ed.** [S.l.]: BOOKMAN COMPANHIA ED, 2001. ISBN 9788573077186.
- HOFFMANN, F. Evolutionary algorithms for fuzzy control system design. **Proceedings of the IEEE**, v. 89, n. 9, p. 1318–1333, 2001. ISSN 0018-9219.
- HONYWELL. 2010. Disponível em: <[http://www.adafruit.com/datasheets/HMC5883L\\_3-Axis\\_Digital\\_Compass\\_IC.pdf](http://www.adafruit.com/datasheets/HMC5883L_3-Axis_Digital_Compass_IC.pdf)>.
- HONYWELL. 2013. Disponível em: <<http://www.mouser.com/ds/2/187/DatasheetSS360PT-267420.pdf>>.
- INSTRUMENTS, T. 2014. Disponível em: <<http://www.ti.com/tool/ek-tm4c123gxl>>.
- INVENSENSE. 2013. Disponível em: <<http://invensense.com/mems/gyro/documents/PS-MPU-6000A-00v3.4.pdf>>.
- JÚNIOR, C.; YONEYAMA, T. **Inteligência artificial: em controle e automação.** [S.l.]: Edgard Blucher, 2002. ISBN 9788521203100.
- KIM, S.-Y.; CHOI, C.; LEE, K.; LEE, W. An improved rotor position estimation with vector-tracking observer in pmsm drives with low-resolution hall-effect sensors. **Industrial Electronics, IEEE Transactions on**, v. 58, n. 9, p. 4078–4086, Sept 2011. ISSN 0278-0046.
- KON, J.; YAMASHITA, Y.; TANAKA, T.; TASHIRO, A.; DAIGUJI, M. Practical application of model identification based on {ARX} models with transfer functions. **Control Engineering Practice**, v. 21, n. 2, p. 195 – 203, 2013. ISSN 0967-0661. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0967066112002134>>.
- LEISHMAN, J. G. The bréguet-richet quadrotor helicopter of. **Vertiflite**, n. 47, p. 4, 2002.
- LILLY, J. **Fuzzy Control and Identification.** [S.l.]: Wiley, 2011. ISBN 9781118097816.
- MAMDANI, E. H.; ASSILIAN, S. An experiment in linguistic synthesis with a fuzzy logic controller. **International journal of man-machine studies**, Elsevier, v. 7, n. 1, p. 1–13, 1975.
- MATHWORKS. 2013. Disponível em: <<http://www.mathworks.com/products/simulink/>>.
- MCCORMICK, B. **Aerodynamics, aeronautics, and flight mechanics.** [S.l.]: Wiley, 1979. ISBN 9780471030324.
- MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **Bulletin of Mathematical Biophysics**, v. 5, p. 115–133, 1943.
- MEYER, J.; SENDOBRY, A.; KOHLBRECHER, S.; KLINGAUF, U.; STRYK, O. von. Comprehensive simulation of quadrotor uavs using ros and gazebo. In: **Simulation, Modeling, and Programming for Autonomous Robots.** [S.l.]: Springer, 2012. p. 400–411.
- MUNSON, K. **Helicopters and Other Rotorcraft Since 1907.** [S.l.]: Macmillan, 1969. (The Pocket encyclopedia of world aircraft in color).
- OGATA, K.; MAYA, P. Á.; LEONARDI, F. **Engenharia de controle moderno.** [S.l.]: Prentice Hall, 2003.

OLIVEIRA, A. P. B. S. D. **Controlador de VANT do tipo helicoptero baseado em redes neurais artificiais**. Natal, RN: [s.n.], 2012. Disponível em: <[http://www.sigaa.ufrn.br/sigaa/public/programa/defesas.jsf?lc=pt\\_BR&id=103](http://www.sigaa.ufrn.br/sigaa/public/programa/defesas.jsf?lc=pt_BR&id=103)>.

PARROT. **Quadricoptero Ar Parrot**. 2013. Disponível em: <<http://www.parrot.com>>.

PAULA, J. C. de. **Desenvolvimento de um vant do tipo quadrirotor para obtenção de imagens aéreas em alta definição**. Curitiba, PR: [s.n.], 2012. Disponível em: <[http://dspace.c3sl.ufpr.br/dspace/bitstream/handle/1884/29886/R - D - JULIO CESAR DE PAULA.pdf?sequence=1](http://dspace.c3sl.ufpr.br/dspace/bitstream/handle/1884/29886/R-D-JULIO_CESAR_DE_PAULA.pdf?sequence=1)>.

ROSENBLATT, F. **The perceptron: A perceiving and recognizing automaton**. [S.l.], 1957.

ROSS, F. **Flying Windmills: The Story of the Helicopter**. [S.l.]: Lothrop, Lee & Shepard, 1953.

RUMELHART, D.; HINTON, G.; WILLIAMS, R. Learning representations by back-propagating errors. **Nature**, v. 323, n. 6088, p. 533–536, 1986.

SILVA, L. N. de C. **Análise e síntese de estratégias de aprendizado para redes neurais artificiais**. Campinas, SP: [s.n.], 1998. Disponível em: <<http://www.bibliotecadigital.unicamp.br/document/?code=vtls000136455>>.

SPECIALTIES, M. 2012. Disponível em: <<http://www.meas-spec.com/downloads/MS5611-01BA03.pdf>>.

TAHAR, K.; AHMAD, A.; AKIB, W. Uav-based stereo vision for photogrammetric survey in aerial terrain mapping. In: **Computer Applications and Industrial Electronics (ICCAIE), 2011 IEEE International Conference on**. [S.l.: s.n.], 2011. p. 443–447.

TAKAGI, T.; SUGENO, M. Fuzzy identification of systems and its applications to modeling and control. **Systems, Man and Cybernetics, IEEE Transactions on**, SMC-15, n. 1, p. 116–132, 1985. ISSN 0018-9472.

TURNIGY. 2013. Disponível em: <[http://www.hobbyking.com/hobbyking/store/\\_14737\\_Turnigy\\_L2215J\\_900\\_Brushless\\_Motor\\_200w\\_.html](http://www.hobbyking.com/hobbyking/store/_14737_Turnigy_L2215J_900_Brushless_Motor_200w_.html)>.

VIEIRA, J. C. S. **Plataforma móvel aérea quadrotor**. Dissertação (Mestrado) — Universidade do Minho - Ciclo de Estudos Integrados Conducentes ao Grau de Mestre em Engenharia Eletrónica Industrial e Computadores, 2011. Disponível em: <[http://intranet.dei.uminho.pt/gdmi/site/arquivo/detalhe\\_arquivo.php?id=269](http://intranet.dei.uminho.pt/gdmi/site/arquivo/detalhe_arquivo.php?id=269)>.

WILLMANN, J.; AUGUGLIARO, F.; CADALBERT, T.; D'ANDREA, R.; GRAMAZIO, F.; KOHLER, M. Aerial Robotic Construction: Towards a New Field of Architectural Research. **International journal of architectural computing**, Multi-Science Publishing, v. 10, p. 439–460, 2012.

ZADEH, L. A. Fuzzy sets. **Information and Control**, v. 8, p. 338–353, 1965. Disponível em: <<http://www-bisc.cs.berkeley.edu/Zadeh-1965.pdf>>.



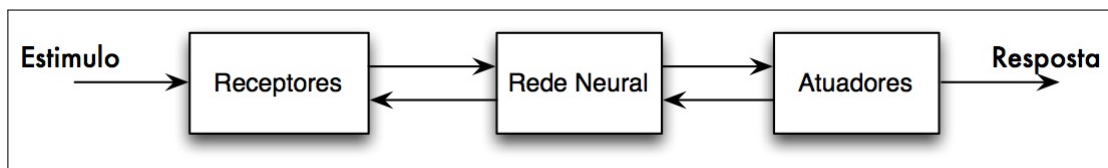
## APÊNDICE A – REDES NEURAIS ARTIFICIAIS

O conceito de redes neurais descrito por HAYKIN (2001) é de um processador maciçamente paralelamente distribuído, com várias unidades de processamento simples. Outra característica das redes neurais é a capacidade de armazenar conhecimento que fica disponível para uso quando necessário.

Para Beale e Jackson (2010), Redes Neurais Artificiais (RNA) são uma representação simplificada de como o cérebro humano lida com informações, esta representação então é aplicada na resolução de problemas computacionais.

O cérebro possui muitos tipos de células, entre elas os neurônios, em uma quantidade de bilhões, criando a rede neural, que são responsáveis por processamento de sinais estimulados, sinapses, externamente e/ou de outros neurônios.

O funcionamento do sistema nervoso no ser humano, simplificado como mostra a Figura 53, tem três estágios. Os estímulos são convertidos em impulsos elétricos pelos receptores e repassados a rede neural. Os atuadores fazem o processo inverso, convertendo os impulsos elétricos em estímulos na saída do sistema (HAYKIN, 2001).



**Figura 53 – Representação do Sistema Neural**

Fonte: (HAYKIN, 2001).

Um dos principais componentes em uma RNA é a representação de funcionamento do neurônio. Um dos primeiros modelos foi apresentado por McCulloch e Pitts (1943), era uma central de processamento binária e provou seu funcionamento executando várias operações lógicas.

Mais tarde, Rosenblatt (1957) idealizou o perceptron, Figura 54, e também um algoritmo para ajustes de pesos no perceptron, provou que em padrões linearmente separáveis ele converge. Um dos elementos básicos é as sinapses  $x_1, x_2, \dots, x_n$ , cada uma com um peso representado por  $W_{k1}, W_{k2}, \dots, W_{km}$  que pondera individualmente as entradas. Todas as entradas então são somadas, que é conhecido como combinador linear. Finalmente a função de ativação funciona como um limitante, permitindo ou não o sinal de saída (OLIVEIRA, 2012).

O ajuste de cada peso sináptico nos perceptrons é realizado através da comparação da saída com o valor desejado, gerando um processo de aprendizagem. A Figura 55 mostra a classificação de pontos com classes diferentes, os ajustes de pesos são efetuados até encontrar uma reta que os separe, sendo possível identificar as diferentes classes (OLIVEIRA, 2012).

Há diversos modelos de redes neurais, um dos critérios de classificação é quanto ao método de aprendizagem. Eles podem ser classificados em três grupos: aprendizado supervisionado, não-supervisionado e por reforço (CALVO, 2007). Essas serão apresentadas em seguida de forma sucinta.

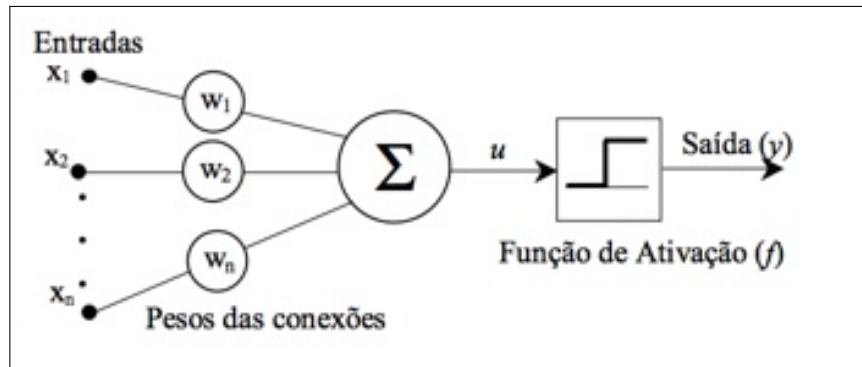


Figura 54 – Perceptron Rosenblatt (1957)

Fonte: Silva (1998).

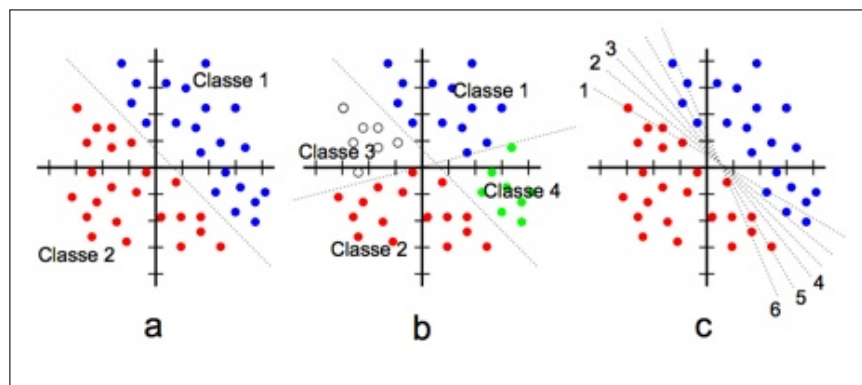


Figura 55 – Classificação de pontos

Fonte: OLIVEIRA (2012).

#### A.0.2.1 Aprendizado supervisionado

O aprendizado ocorre através de um conjunto de exemplos com entrada e saída conhecida. Assim ela deve aprender através da entrada gerar a saída esperada, se diferir, a diferença encontrada é usada para ajustes dos pesos das sinapses. O processo se repete de forma que a cada interação diminua o erro.

Um dos métodos para a correção dos pesos é a *backpropagation* proposto por (RUMELHART; HINTON; WILLIAMS, 1986), que propaga o erro no sentido para trás (*backward*) da camada de saída para a de entrada, assim ajustando cada peso.

#### A.0.2.2 Aprendizado não-supervisionado

Quando não é possível utilizar um conjunto de dados para a aprendizagem, somente é fornecido à rede os padrões de entrada e através destes é gerado uma identificação interna de classes automaticamente. Ao termino do processo de aprendizagem a rede consegue identificar entradas através de correlações automaticamente criadas anteriormente.

#### A.0.2.3 Aprendizado por reforço

O método de aprendizado por reforço, geralmente, tem apenas a informação da qualidade de desempenho do RNA através de um agente que a partir das interações com o ambiente

define a direção do aprendizado. Assim os pesos internos são ajustados de forma a melhorar as saídas futuras para um mesmo estado observado, otimizando-se durante o processo.

Pelo método por reforço o aprendizado é baseado na tentativa e erro, onde é ajustado o agente de modo a ter um fator de exploração provocando ações novas ou utilizando o conhecimento acumulado. Em controle, as várias tentativas acumulam informações as quais produzem um controle mais aperfeiçoado. A grande adaptabilidade gerada pelo método, faz com que seu desempenho seja superior em controle de plantas complexas ao mesmo tempo isso faz com que as funções intermediárias de controle sejam mais instáveis (EDUARDO, 2009).

## APÊNDICE B – SCRIPT PARA IDENTIFICAÇÃO DA FUNÇÃO DE TRANSFÊRENCIA NO MATLAB

```

%% Obtenção das Funções de Transferência para Simulação
%
%%
clc; %aaa
clear;
close all;
close ALL HIDDEN;
format SHORTENG;
%%
% Dados Globais
Resolucao_Sensor= 7;
Amostragem_Var = 700;
Periodo_Tick = 80e-9;
d=10.2360e-6;
b=613.4431e-6;
l=25/100;
peso=1.02;

%% Motor 1
%
%%
% * Variaveis entrada dos dados da curva de velocidade
ArquivoCurva_M1='M1_Curva.txt';

% Região mais linear
M1_InicioL=5;
M1_FimL=77;
M1_LimiteInf=1040; % Duty Cycle PWM em us
M1_LimiteSup=1770; % Duty Cycle PWM em us
M1_ordemPoly=5; % Ordem do polinomio de linealizacao

%%
% * Variaveis entrada de dados para encontrar Função de Transferência
ArquivoVariavel_M1= 'M1_Variavel.txt';

M1_ndelay=12;

M1_na=2;
M1_nb=2;
M1_nk=M1_ndelay;

```

```

M1_np=3;
M1_nz=2;

%%
% * Codigo para Processar os Dados Curva de velocidade Motor 1
%

temp=dlmread(ArquivoCurva_M1, ',',0,0);
M1_EntradaCurva=temp(:,1);
SaidaCurvaTicks = temp(:,2);

% Conversão de velocidade em ticks de tempo para RPM
M1_SaidaCurva=60./(SaidaCurvaTicks.*Periodo_Tick*Resolucao_Sensor);

M1_EntradaCurvaL=M1_EntradaCurva(M1_InicioL:M1_FimL);
M1_SaidaCurvaL=M1_SaidaCurva(M1_InicioL:M1_FimL);
M1_PWM_Lim=M1_EntradaCurvaL;
x=linspace(0,1,size(M1_PWM_Lim,1));
x=x';

%%
% Polinômio Obtido para linealização da velocidade com o PWM para Motor
1
M1_PolyVelocidade = polyfit(M1_PWM_Lim,M1_SaidaCurvaL,M1_ordemPoly)

%%
% Relação entre velocidade e PWM para Motor 1
M1_SaidaPoly=polyval(M1_PolyVelocidade,x);
figure
plot(M1_EntradaCurva,M1_SaidaCurva);
grid minor;
xlabel('Duty Cycle PWM (us)');
ylabel('Velocidade (RPM)');
title('Relação entre PWM e velocidade Motor 1');
%%
% Relação entre velocidade e PWM linealizado e normalizado para o Motor
1
figure;
plot(x,M1_SaidaCurvaL);
hold;
plot(x,M1_SaidaPoly,'red');
grid minor;
xlabel('PWM (Normalizado)');
ylabel('Velocidade (RPM)');
title('Relação entre PWM e velocidade linealizado Motor 1');

%%
% * Codigo para Processar os Dados Função de Transferência para Motor 1
temp=dlmread(ArquivoVariavel_M1, ',',0,0);
EntradaVariavel2= temp(:,1);
SaidaVariavelTicks= temp(:,2);

```

```

QtdeDados = size(EntradaVariavel2,1);
QtdeValidacao = round(QtdeDados/3);
QtdeAvaliacao = QtdeDados;%round(QtdeDados*2/3);

% Conversão de velocidade em ticks de tempo para RPM
M1_SaidaVariavel=60./(SaidaVariavelTicks.*Periodo_Tick*Resolucao_Sensor)
;
% Normalizando o PWM
M1_EntradaVariavel=(EntradaVariavel2-M1_LimiteInf)./(M1_LimiteSup-
M1_LimiteInf);

M1_AmostrasV=iddata(M1_SaidaVariavel(1:QtdeValidacao,1),
M1_EntradaVariavel(1:QtdeValidacao,1),1/Amostragem_Var,'InputName','
PWM','OutputName','Velocidade','OutputUnit','RPM','Name','Amostras
Validação');
M1_AmostrasA=iddata(M1_SaidaVariavel(QtdeValidacao+1:QtdeDados,1),
M1_EntradaVariavel(QtdeValidacao+1:QtdeDados,1),1/Amostragem_Var,'
InputName','PWM','OutputName','Velocidade','OutputUnit','RPM','Name',
'Amostras Experimentação');
[M1_AmostrasVd,M1_TAv]=detrend(M1_AmostrasV,0);
[M1_AmostrasAd,M1_TAd]=detrend(M1_AmostrasA,0);
%%
% Offset removido
M1_TAd

%%
% * Metodo 1 ARX

M1_ftObtida=arx(M1_AmostrasAd,[M1_na M1_nb M1_nk])

compare(M1_AmostrasA,M1_ftObtida,compareOptions('InputOffset',M1_TAd.
InputOffset,'OutputOffset',M1_TAd.OutputOffset))
compare(M1_AmostrasV,M1_ftObtida,compareOptions('InputOffset',M1_TAd.
InputOffset,'OutputOffset',M1_TAd.OutputOffset))

error=resid(M1_AmostrasAd,M1_ftObtida);
figure;
plot(error)
grid minor;
%%
figure;
step(M1_ftObtida,stepDataOptions('StepAmplitude',1-M1_TAd.InputOffset))
grid minor;
%%
stepinfo(M1_ftObtida)

%%
% * Metodo 2 TF Estimate

M1_ftObtida2=tfest(M1_AmostrasAd,M1_np,M1_nz,M1_ndelay,'Ts',1/

```

```

    Amostragem_Var)
compare(M1_AmostrasA,M1_ftObtida2,compareOptions('InputOffset',M1_TAv.
    InputOffset,'OutputOffset',M1_TAv.OutputOffset));
compare(M1_AmostrasV,M1_ftObtida2,compareOptions('InputOffset',M1_TAv.
    InputOffset,'OutputOffset',M1_TAv.OutputOffset))
error=resid(M1_AmostrasAd,M1_ftObtida2);
figure;
plot(error)
grid minor;
%%
figure;
step(M1_ftObtida2,stepDataOptions('StepAmplitude',1-M1_TAd.InputOffset))
;
grid minor;
%%
stepinfo(M1_ftObtida2)

[M1_num,M1_den,Ts] = tfdata(M1_ftObtida,'v');
[M1_num2,M1_den2] = tfdata(M1_ftObtida2,'v');
M1_simulacao.time=0:1/700:size(M1_EntradaVariavel,1)*(1/700)-0.001;
M1_simulacao.signals.values=M1_EntradaVariavel;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%close all hidden;

%% Motor 2
%
%%
% * Variaveis entrada dos dados da curva de velocidade
ArquivoCurva_M2='M2_Curva.txt';

% Região mais linear
M2_InicioL=5;
M2_FimL=77;
M2_LimiteInf=1040; % Duty Cycle PWM em us
M2_LimiteSup=1770; % Duty Cycle PWM em us
M2_ordemPoly=5; % Ordem do polinomio de linealizacao

%%
% * Variaveis entrada de dados para encontrar Função de Transferência
ArquivoVariavel_M2= 'M2_Variavel.txt';

M2_ndelay=12;

M2_na=2;
M2_nb=2;
M2_nk=M2_ndelay;

```

```

M2_np=3;
M2_nz=1;

%%
% * Codigo para Processar os Dados Curva de velocidade Motor 2
%

temp=dlmread(ArquivoCurva_M2, ',',0,0);
M2_EntradaCurva=temp(:,1);
SaidaCurvaTicks = temp(:,2);

M2_SaidaCurva=60./(SaidaCurvaTicks.*Periodo_Tick*Resolucao_Sensor); %
    Conversão de velocidade em ticks de tempo para RPM

M2_EntradaCurvaL=M2_EntradaCurva(M2_InicioL:M2_FimL);
M2_SaidaCurvaL=M2_SaidaCurva(M2_InicioL:M2_FimL);
M2_PWM_Lim=(M2_EntradaCurvaL-M2_LimiteInf)/(M2_LimiteSup-M2_LimiteInf)
    ;%% normalizado
x=linspace(0,1,size(M2_PWM_Lim,1));
x=x';

%%
% Polinômio Obtido para linealização da velocidade com o PWM para Motor
    1
M2_PolyVelocidade = polyfit(M2_PWM_Lim,M2_SaidaCurvaL,M2_ordemPoly)

%%
% Relação entre velocidade e PWM para Motor 2
M2_SaidaPoly=polyval(M2_PolyVelocidade,x);
figure
plot(M2_EntradaCurva,M2_SaidaCurva);
grid minor;
xlabel('Duty Cycle PWM (us)');
ylabel('Velocidade (RPM)');
title('Relação entre PWM e velocidade Motor 2');
%%
% Relação entre velocidade e PWM linealizado e normalizado para o Motor
    1
figure;
plot(x,M2_SaidaCurvaL);
hold;
plot(x,M2_SaidaPoly,'red');
grid minor;
xlabel('PWM (Normalizado)');
ylabel('Velocidade (RPM)');
title('Relação entre PWM e velocidade linealizado Motor 2');

%%
% * Codigo para Processar os Dados Função de Transferência para Motor 1
temp=dlmread(ArquivoVariavel_M2, ',',0,0);
EntradaVariavel2= temp(:,1);

```



```

SaidaVariavelTicks= temp(:,2);
QtdeDados = size(EntradaVariavel2,1);
QtdeValidacao = round(QtdeDados/3);
QtdeAvaliacao = round(QtdeDados*2/3);

M2_SaidaVariavel=60./(SaidaVariavelTicks.*Periodo_Tick*Resolucao_Sensor)
; % Conversão de velocidade em ticks de tempo para RPM
M2_EntradaVariavel=(EntradaVariavel2-M2_LimiteInf)./(M2_LimiteSup-
M2_LimiteInf); % Normalizando o PWM

M2_AmostrasV=iddata(M2_SaidaVariavel(1:QtdeValidacao,1),
M2_EntradaVariavel(1:QtdeValidacao,1),1/Amostragem_Var,'InputName','
PWM','OutputName','Velocidade','OutputUnit','RPM','Name','Amostras
Validação');
M2_AmostrasA=iddata(M2_SaidaVariavel(1:QtdeDados,1),M2_EntradaVariavel
(1:QtdeDados,1),1/Amostragem_Var,'InputName','PWM','OutputName','
Velocidade','OutputUnit','RPM','Name','Amostras Experimentação');
[M2_AmostrasVd,M2_TAv]=detrend(M2_AmostrasV,0);
[M2_AmostrasAd,M2_TAd]=detrend(M2_AmostrasA,0);
%%
% Offset removido
M2_TAd

%%
% * Metodo 1 ARX

M2_ftObtida=arx(M2_AmostrasAd,[M2_na M2_nb M2_nk])

compare(M2_AmostrasA,M2_ftObtida,compareOptions('InputOffset',M2_TAd.
InputOffset,'OutputOffset',M2_TAd.OutputOffset))
compare(M2_AmostrasV,M2_ftObtida,compareOptions('InputOffset',M2_TAd.
InputOffset,'OutputOffset',M2_TAd.OutputOffset))

error=resid(M2_AmostrasAd,M2_ftObtida);
figure;
plot(error)
grid minor;
%%
figure;
step(M2_ftObtida,stepDataOptions('StepAmplitude',1-M2_TAd.InputOffset))
grid minor;
%%
stepinfo(M2_ftObtida)

%%
% * Metodo 2 TF Estimate

M2_ftObtida2=tfest(M2_AmostrasAd,M2_np,M2_nz,M2_ndelay,'Ts',1/
Amostragem_Var)
compare(M2_AmostrasA,M2_ftObtida2,compareOptions('InputOffset',M2_TAv.

```

```

    InputOffset , 'OutputOffset' ,M2_TAv.OutputOffset));
compare(M2_AmostrasV,M2_ft0btida2,compareOptions('InputOffset',M2_TAv.
    InputOffset,'OutputOffset',M2_TAv.OutputOffset))
error=resid(M2_AmostrasAd,M2_ft0btida2);
figure;
plot(error)
grid minor;
%%
figure;
step(M2_ft0btida2,stepDataOptions('StepAmplitude',1-M2_TAd.InputOffset))
;
grid minor;
%%
stepinfo(M2_ft0btida2)

[M2_num,M2_den,Ts] = tfdata(M2_ft0btida,'v');
[M2_num2,M2_den2] = tfdata(M2_ft0btida2,'v');
M2_simulacao.time=0:1/700:size(M2_EntradaVariavel,1)*(1/700)-0.001;
M2_simulacao.signals.values=M2_EntradaVariavel;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%close all hidden;

%% Motor 3
%
%%
% * Variaveis entrada dos dados da curva de velocidade
ArquivoCurva_M3='M3_Curva.txt';

% Região mais linear
M3_InicioL=5;
M3_FimL=77;
M3_LimiteInf=1040; % Duty Cycle PWM em us
M3_LimiteSup=1770; % Duty Cycle PWM em us
M3_ordemPoly=5; % Ordem do polinomio de linealizacao

%%
% * Variaveis entrada de dados para encontrar Função de Transferência
ArquivoVariavel_M3='M3_Variavel.txt';

M3_ndelay=12;

M3_na=2;
M3_nb=2;
M3_nk=M3_ndelay;

M3_np=3;
M3_nz=2;

```

```

%%
% * Codigo para Processar os Dados Curva de velocidade Motor 3
%

temp=dlmread(ArquivoCurva_M3, ',',0,0);
M3_EntradaCurva=temp(:,1);
SaidaCurvaTicks = temp(:,2);

M3_SaidaCurva=60./(SaidaCurvaTicks.*Periodo_Tick*Resolucao_Sensor); %
    Conversão de velocidade em ticks de tempo para RPM

M3_EntradaCurvaL=M3_EntradaCurva(M3_InicioL:M3_FimL);
M3_SaidaCurvaL=M3_SaidaCurva(M3_InicioL:M3_FimL);
M3_PWM_Lim=(M3_EntradaCurvaL-M3_LimiteInf)/(M3_LimiteSup-M3_LimiteInf)
    ;%% normalizado
x=linspace(0,1,size(M3_PWM_Lim,1));
x=x';

%%
% Polinômio Obtido para linealização da velocidade com o PWM para Motor
    3
M3_PolyVelocidade = polyfit(M3_PWM_Lim,M3_SaidaCurvaL,M3_ordemPoly)

%%
% Relação entre velocidade e PWM para Motor 3
M3_SaidaPoly=polyval(M3_PolyVelocidade,x);
figure
plot(M3_EntradaCurva,M3_SaidaCurva);
grid minor;
xlabel('Duty Cycle PWM (us)');
ylabel('Velocidade (RPM)');
title('Relação entre PWM e velocidade Motor 3');
%%
% Relação entre velocidade e PWM linealizado e normalizado para o Motor
    3
figure;
plot(x,M3_SaidaCurvaL);
hold;
plot(x,M3_SaidaPoly,'red');
grid minor;
xlabel('PWM (Normalizado)');
ylabel('Velocidade (RPM)');
title('Relação entre PWM e velocidade linealizado Motor 3');

%%
% * Codigo para Processar os Dados Função de Transferência para Motor 1
temp=dlmread(ArquivoVariavel_M3, ',',0,0);
EntradaVariavel2= temp(:,1);
SaidaVariavelTicks= temp(:,2);
QtdeDados = size(EntradaVariavel2,1);
QtdeValidacao = round(QtdeDados/3);

```

```

QtdeAvaliacao = round(QtdeDados*2/3);

M3_SaidaVariavel=60./(SaidaVariavelTicks.*Periodo_Tick*Resolucao_Sensor)
; % Conversão de velocidade em ticks de tempo para RPM
M3_EntradaVariavel=(EntradaVariavel2-M3_LimiteInf)./(M3_LimiteSup-
M3_LimiteInf); % Normalizando o PWM

M3_AmostrasV=iddata(M3_SaidaVariavel(1:QtdeValidacao,1),
M3_EntradaVariavel(1:QtdeValidacao,1),1/Amostragem_Var,'InputName','
PWM','OutputName','Velocidade','OutputUnit','RPM','Name','Amostras
Validação');
M3_AmostrasA=iddata(M3_SaidaVariavel(QtdeValidacao+1:QtdeDados,1),
M3_EntradaVariavel(QtdeValidacao+1:QtdeDados,1),1/Amostragem_Var,'
InputName','PWM','OutputName','Velocidade','OutputUnit','RPM','Name',
'Amostras Experimentação');
[M3_AmostrasVd,M3_TAv]=detrend(M3_AmostrasV,0);
[M3_AmostrasAd,M3_TAd]=detrend(M3_AmostrasA,0);
%%
% Offset removido
M3_TAd

%%
% * Metodo 1 ARX

M3_ftObtida=arx(M3_AmostrasAd,[M3_na M3_nb M3_nk])

compare(M3_AmostrasA,M3_ftObtida,compareOptions('InputOffset',M3_TAd.
InputOffset,'OutputOffset',M3_TAd.OutputOffset))
compare(M3_AmostrasV,M3_ftObtida,compareOptions('InputOffset',M3_TAd.
InputOffset,'OutputOffset',M3_TAd.OutputOffset))

error=resid(M3_AmostrasAd,M3_ftObtida);
figure;
plot(error)
grid minor;
%%
figure;
step(M3_ftObtida,stepDataOptions('StepAmplitude',1-M3_TAd.InputOffset))
grid minor;
%%
stepinfo(M3_ftObtida)

%%
% * Metodo 2 TF Estimate

M3_ftObtida2=tfest(M3_AmostrasAd,M3_np,M3_nz,M3_ndelay,'Ts',1/
Amostragem_Var)
compare(M3_AmostrasA,M3_ftObtida2,compareOptions('InputOffset',M3_TAv.
InputOffset,'OutputOffset',M3_TAv.OutputOffset));
%compare(M3_AmostrasV,M3_ftObtida2,compareOptions('InputOffset',M3_TAv.

```

```

    InputOffset , 'OutputOffset' , M3_TAv.OutputOffset))
error=resid(M3_AmostrasAd , M3_ft0btida2);
figure;
plot(error)
grid minor;
%%
figure;
step(M3_ft0btida2 , stepDataOptions('StepAmplitude' , 1-M3_TAd.InputOffset))
;
grid minor;
%%
stepinfo(M3_ft0btida2)

[M3_num , M3_den , Ts] = tfdata(M3_ft0btida , 'v');
[M3_num2 , M3_den2] = tfdata(M3_ft0btida2 , 'v');
M3_simulacao.time=0:1/700:size(M3_EntradaVariavel , 1)*(1/700) -0.001;
M3_simulacao.signals.values=M3_EntradaVariavel;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%close all hidden;

%% Motor 4
%
%%
% * Variaveis entrada dos dados da curva de velocidade
ArquivoCurva_M4='M4_Curva.txt';

% Região mais linear
M4_InicioL=5;
M4_FimL=77;
M4_LimiteInf=1040; % Duty Cycle PWM em us
M4_LimiteSup=1770; % Duty Cycle PWM em us
M4_ordemPoly=5; % Ordem do polinomio de linealizacao

%%
% * Variaveis entrada de dados para encontrar Função de Transferência
ArquivoVariavel_M4= 'M4_Variavel.txt';

M4_ndelay=12;

M4_na=2;
M4_nb=2;
M4_nk=M4_ndelay;

M4_np=2;
M4_nz=1;

```

```

%%
% * Codigo para Processar os Dados Curva de velocidade Motor 4
%

temp=dlmread(ArquivoCurva_M4, ',',0,0);
M4_EntradaCurva=temp(:,1);
SaidaCurvaTicks = temp(:,2);

M4_SaidaCurva=60./(SaidaCurvaTicks.*Periodo_Tick*Resolucao_Sensor); %
    Conversão de velocidade em ticks de tempo para RPM

M4_EntradaCurvaL=M4_EntradaCurva(M4_InicioL:M4_FimL);
M4_SaidaCurvaL=M4_SaidaCurva(M4_InicioL:M4_FimL);
M4_PWM_Lim=(M4_EntradaCurvaL-M4_LimiteInf)/(M4_LimiteSup-M4_LimiteInf)
    ;%% normalizado
x=linspace(0,1,size(M4_PWM_Lim,1));
x=x';

%%
% Polinômio Obtido para linealização da velocidade com o PWM para Motor
    4
M4_PolyVelocidade = polyfit(M4_PWM_Lim,M4_SaidaCurvaL,M4_ordemPoly)

%%
% Relação entre velocidade e PWM para Motor 4
M4_SaidaPoly=polyval(M4_PolyVelocidade,x);
figure
plot(M4_EntradaCurva,M4_SaidaCurva);
grid minor;
xlabel('Duty Cycle PWM (us)');
ylabel('Velocidade (RPM)');
title('Relação entre PWM e velocidade Motor 4');
%%
% Relação entre velocidade e PWM linealizado e normalizado para o Motor
    4
figure;
plot(x,M4_SaidaCurvaL);
hold;
plot(x,M4_SaidaPoly,'red');
grid minor;
xlabel('PWM (Normalizado)');
ylabel('Velocidade (RPM)');
title('Relação entre PWM e velocidade linealizado Motor 4');

%%
% * Codigo para Processar os Dados Função de Transferência para Motor 1
temp=dlmread(ArquivoVariavel_M4, ',',0,0);
EntradaVariavel2= temp(:,1);
SaidaVariavelTicks= temp(:,2);
QtdeDados = size(EntradaVariavel2,1);
QtdeValidacao = round(QtdeDados/3);
QtdeAvaliacao = round(QtdeDados*2/3);

```

```

M4_SaidaVariavel=60./(SaidaVariavelTicks.*Periodo_Tick*Resolucao_Sensor)
; % Conversão de velocidade em ticks de tempo para RPM
M4_EntradaVariavel=(EntradaVariavel2-M4_LimiteInf)./(M4_LimiteSup-
M4_LimiteInf); % Normalizando o PWM

M4_AmostrasV=iddata(M4_SaidaVariavel(1:QtdeValidacao,1),
M4_EntradaVariavel(1:QtdeValidacao,1),1/Amostragem_Var,'InputName','
PWM','OutputName','Velocidade','OutputUnit','RPM','Name','Amostras
Validação');
M4_AmostrasA=iddata(M4_SaidaVariavel(QtdeValidacao+1:QtdeDados,1),
M4_EntradaVariavel(QtdeValidacao+1:QtdeDados,1),1/Amostragem_Var,'
InputName','PWM','OutputName','Velocidade','OutputUnit','RPM','Name',
'Amostras Experimentação');
[M4_AmostrasVd,M4_TAv]=detrend(M4_AmostrasV,0);
[M4_AmostrasAd,M4_TAd]=detrend(M4_AmostrasA,0);
%%
% Offset removido
M4_TAd

%%
% * Metodo 1 ARX

M4_ftObtida=arx(M4_AmostrasAd,[M4_na M4_nb M4_nk])

compare(M4_AmostrasA,M4_ftObtida,compareOptions('InputOffset',M4_TAd.
InputOffset,'OutputOffset',M4_TAd.OutputOffset))
compare(M4_AmostrasV,M4_ftObtida,compareOptions('InputOffset',M4_TAd.
InputOffset,'OutputOffset',M4_TAd.OutputOffset))

error=resid(M4_AmostrasAd,M4_ftObtida);
figure;
plot(error)
grid minor;
%%
figure;
step(M4_ftObtida,stepDataOptions('StepAmplitude',1-M4_TAd.InputOffset))
grid minor;
%%
stepinfo(M4_ftObtida)

%%
% * Metodo 2 TF Estimate

M4_ftObtida2=tfest(M4_AmostrasAd,M4_np,M4_nz,M4_ndelay,'Ts',1/
Amostragem_Var)
compare(M4_AmostrasA,M4_ftObtida2,compareOptions('InputOffset',M4_TAv.
InputOffset,'OutputOffset',M4_TAv.OutputOffset));
compare(M4_AmostrasV,M4_ftObtida2,compareOptions('InputOffset',M4_TAv.
InputOffset,'OutputOffset',M4_TAv.OutputOffset))

```





## APÊNDICE C – IMPLEMENTAÇÃO NO TIVA C PARA OBTENÇÃO DOS DADOS DO CONJUNTO MOTOR, HÉLICE E ESC

```

#include "padrao.h"
#include <stdlib.h>
#include "drivers\serial.h"
#include "utils\cmdline.h"

void init_system() {

    SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_PLL | SYSCTL_XTAL_16MHZ |
                   SYSCTL_OSC_MAIN);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    ROM_FPUEnable();
    ROM_FPULazyStackingEnable();

}

#define TESTE_STEP 0

void init_console() {
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);

    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);

    ROM_GPIOPinConfigure(GPIO_PA0_UORX);
    ROM_GPIOPinConfigure(GPIO_PA1_UOTX);
    ROM_GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);

    UARTClockSourceSet(UART0_BASE, UART_CLOCK_PIOSC);
    UARTStdioConfig(0, 115200, 16000000);

}

uint16_t *amostras;
uint16_t qtd_amostras =3000, pwm_periodo=1770, freq_amostragem=700, \
freq_contador=10000, pwm_periodo_min=1000, para_amostras=2000;

#define AMOST_SCALE 1
volatile uint16_t pwm_atual=0;

#define FREQ_AMOSTRAGEM(a) (TimerLoadSet64(WTIMER1_BASE, SysCtlClockGet
    ()/a))
#define FREQ_CONTADOR(a) (TimerLoadSet(WTIMER0_BASE, TIMER_A, (
    SysCtlClockGet()/a))
#define ELAPSE_TICKS 0xFFFFFFFFFFFFFFFF

```

```

#define ELAPSE_SCALE 1
int flagtesta =1;

uint16_t velocidades
[60]={1371,1456,1617,1540,1643,1768,1750,1528,1751,1310,1277,1408,
\1552,1526,1719,1531,1475,1532,1620,1259,1666,
1324,1499,1384,1442,1594,1338,1395,1353,1351,\
1420,1708,1495,1460,1343,1754,1462,1689,1570,\
1446,1706,1658,
1492,1673,1717,1473,1424,1560,1719,1615,1446,1632,\
1746,1532,1531,1412,1287,1345,1298,1491};

void PWM_PERIODO(double a) {
    PWMPulseWidthSet(PWM0_BASE, PWM_OUT_0,(unsigned long)
        PWMGenPeriodGet( PWM0_BASE, PWM_OUT_0)*a/0.005);
    pwm_atual=(a*1000000);
}

void init_timerAmostragem(void) {
    SysCtlPeripheralEnable(SYSCTL_PERIPH_WTIMER1);
    TimerConfigure(WTIMER1_BASE, TIMER_CFG_PERIODIC_UP);
    FREQ_AMOSTRAGEM(freq_amostragem);
    TimerPrescaleSet(WTIMER1_BASE, TIMER_A, AMOST_SCALE);
}

void start_timerAmostragem() {
    TimerIntEnable(WTIMER1_BASE, TIMER_TIMA_TIMEOUT);
    IntEnable(INT_WTIMER1A);
    TimerEnable(WTIMER1_BASE, TIMER_A|TIMER_B);
    IntMasterEnable();
}

void start_timerCount() {
    TimerEnable(WTIMER0_BASE, TIMER_A);
}

void init_timerCounter(void) {

    SysCtlPeripheralEnable(SYSCTL_PERIPH_WTIMER0);
    TimerConfigure(WTIMER0_BASE, TIMER_CFG_SPLIT_PAIR |
        TIMER_CFG_A_CAP_TIME_UP);
    TimerLoadSet(WTIMER0_BASE, TIMER_A, (SysCtlClockGet()/freq_contador)
        );
    TimerIntEnable(WTIMER0_BASE, (TIMER_TIMA_TIMEOUT | TIMER_CAPA_EVENT
        ));
    TimerPrescaleSet(WTIMER0_BASE, TIMER_A, AMOST_SCALE);
    IntEnable(INT_WTIMER0A);
    TimerEnable(WTIMER0_BASE, TIMER_A);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);
    GPIOPinConfigure(GPIO_PC4_WTOCCP0);
    GPIOPinTypeTimer(GPIO_PORTC_BASE, GPIO_PIN_4);
}

```

```

}

uint64_t soma_valores=0;
uint32_t qtd_valores=0;

uint32_t count_amostras=0;

uint8_t flag = 0,flag2=0, passo=0;
uint32_t tempo_inicial=0;
uint8_t step=0;
volatile uint32_t tempo_final=0;
void
handler_timerAmostragem(void)
{

    TimerIntClear(WTIMER1_BASE, TIMER_TIMA_TIMEOUT);

    UARTprintf("%u,%u\n",pwm_atual,tempo_final);
    count_amostras++;
    if(step==1) {
        if(count_amostras==(qtd_amostras-para_amostras)) {
            PWM_PERIODO(1000e-6);

        } else if(count_amostras>=qtd_amostras) {
            PWM_PERIODO(1000e-6);
            TimerDisable(WTIMER0_BASE,TIMER_A);
            TimerDisable(WTIMER1_BASE,TIMER_A|TIMER_B);
        }
    }else {
        if(count_amostras%2000 == 0) {
            PWM_PERIODO(velocidades[passo++]*1e-6);
        } else if( passo >= 60) {
            PWM_PERIODO(1000e-6);
            TimerDisable(WTIMER0_BASE,TIMER_A);
            TimerDisable(WTIMER1_BASE,TIMER_A|TIMER_B);
        }
    }
}

uint32_t temporaria=0;
void
handler_timerCount(void)
{

    uint32_t ulStatus = TimerIntStatus(WTIMER0_BASE, true);
    TimerIntClear(WTIMER0_BASE, ulStatus);
    if (ulStatus & TIMER_TIMA_TIMEOUT)
    {

```

```

    while(1); //overflow
} else if (ulStatus & TIMER_CAPA_EVENT) {
    if (flag == 0)
    {
        TimerLoadSet(WTIMER0_BASE, TIMER_A, (SysCtlClockGet()/
            freq_contador));
        tempo_inicial = TimerValueGet(WTIMER0_BASE, TIMER_A); //
            read the capture value
        TimerControlEvent(WTIMER0_BASE, TIMER_A,
            TIMER_EVENT_NEG_EDGE);
        flag=1;
    }
    else
    {
        if(flagtesta==1){
            TimerControlEvent(WTIMER0_BASE, TIMER_A,
                TIMER_EVENT_NEG_EDGE);
            temporaria=TimerValueGet(WTIMER0_BASE, TIMER_A); //-
                tempo_inicial;
            if(temporaria<tempo_inicial){
                temporaria+=(SysCtlClockGet()/freq_contador)-
                    tempo_inicial;
            } else {
                temporaria -=tempo_inicial;
            }

            if(qtd_valores >= (7)) {
                tempo_final=soma_valores/qtd_valores;
                qtd_valores =0;
                soma_valores =0;
            }
            soma_valores+=temporaria;
            qtd_valores++;
        } else {
            TimerControlEvent(WTIMER0_BASE, TIMER_A,
                TIMER_EVENT_NEG_EDGE);
            tempo_final=TimerValueGet(WTIMER0_BASE, TIMER_A); //-
                tempo_inicial;
            if(tempo_final<tempo_inicial){
                tempo_final+=(SysCtlClockGet()/freq_contador)-
                    tempo_inicial;
            } else {
                tempo_final -=tempo_inicial;
            }
            soma_valores+=tempo_final;
            qtd_valores++;
        }

        flag=0;
    }
}

```

```

    }
}

void init_PWM(void) {
    SysCtlPWMClockSet(SYSCTL_PWMDIV_4);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM0);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
    GPIOPinConfigure(GPIO_PB6_MOPWM0);
    GPIOPinConfigure(GPIO_PB7_MOPWM1);

    GPIOPinTypePWM(GPIO_PORTB_BASE, GPIO_PIN_6);
    GPIOPinTypePWM(GPIO_PORTB_BASE, GPIO_PIN_7);
    PWMGenConfigure(PWM0_BASE, PWM_GEN_0, PWM_GEN_MODE_UP_DOWN |
                   PWM_GEN_MODE_NO_SYNC);
    PWMGenPeriodSet(PWM0_BASE, PWM_GEN_0, (SysCtlClockGet() / (200*4)));
    PWM_PERIOD0(1e-3);

    PWMGenEnable(PWM0_BASE, PWM_GEN_0);
    PWMOutputState(PWM0_BASE, PWM_OUT_0_BIT|PWM_OUT_1_BIT, true);
}

int CMD_setDados(int argc, char **argv)
{
    (void)argc;
    (void)argv;
    if(argc != 6) {
        UARTprintf("DADOS;ERRO;FIM;ARG\n");
        return (1);
    }
    uint16_t *temp;

    freq_amostragem=atoi(argv[1]);
    para_amostras=atoi(argv[3]);
    FREQ_AMOSTRAGEM(freq_amostragem);
    FREQ_CONTADOR(freq_contador);
    pwm_periodo=(atoi(argv[4]));
    pwm_periodo_min=(atoi(argv[5]));
    UARTprintf("DADOS;OK;FIM\n");
    return (0);
}

int CMD_getDados(int argc, char **argv) {
    UARTprintf("DADOS\n Amostragem: %d;\nQtde: %d;\nParadaAmostra: %d;\n
               nPWMFIM: %d;\nPWM_INI: %d;\nFIM\n",freq_amostragem, qtd_amostras,
               para_amostras, pwm_periodo, pwm_periodo_min);
    return(0);
}

```

```

int CMD_iniAmostragem(int argc, char **argv) {
    step=0;
    soma_valores=0; qtd_valores=0; flag=0; count_amostras=0; tempo_final=0;
    tempo_inicial=0;
    SysCtlDelay(SysCtlClockGet());
    SysCtlDelay(SysCtlClockGet());
    if (TESTE_STEP != 1) {
        PWM_PERIODO(pwm_periodo*1e-6);
        init_timerCounter();
        init_timerAmostragem();
        start_timerCount();
        SysCtlDelay(SysCtlClockGet());
        SysCtlDelay(SysCtlClockGet());
        //init_timerElapse();
        start_timerAmostragem();
    } else {
        init_timerCounter();
        init_timerAmostragem();
        SysCtlDelay(SysCtlClockGet());
        start_timerCount();
        PWM_PERIODO(pwm_periodo*1e-6);
        start_timerAmostragem();
    }
    return(0);
}

int CMD_testepwm(int argc, char **argv) {
    flagtesta=1;
    PWM_PERIODO(atoi(argv[1])*1e-6);
    init_timerCounter();
    UARTprintf("atualizado pwm %d\n", atoi(argv[1]));
    return 0;
}

int CMD_getVelocidade(int argc, char **argv) {
    init_timerCounter();
    start_timerCount();
    UARTprintf("Velocidade %d\n", tempo_final);
    return 0;
}

int CMD_getCurva(int argc, char **argv) {
    count_amostras=0;
    init_timerCounter();
    start_timerCount();
    int k=0;
    for(k=1000; k<=2000; k=k+10) {
        PWM_PERIODO(k*1e-6);
        SysCtlDelay(SysCtlClockGet()/2);
        UARTprintf("%u,%u\n", pwm_atual, tempo_final);
    }
    PWM_PERIODO(1000*1e-6);
    return 0;
}

```

```

}

int CMD_getStep(int argc, char **argv) {
    count_amostras=0;
    step=1;
    init_timerCounter();
    init_timerAmostragem();
    PWM_PERIODO(pwm_periodo_min*1e-6);
    SysCtlDelay(SysCtlClockGet());
    start_timerCount();
    PWM_PERIODO(pwm_periodo*1e-6);
    start_timerAmostragem();
    return 0;
}

int CMD_getGeral(int argc, char **argv) {
    PWM_PERIODO(1000*1e-6);
    UARTprintf("CURVAVELOCIDADE1\n");
    //CMD_getCurva(NULL, NULL);
    UARTprintf("\nCURVAVELOCIDADE2\n");

    //CMD_getCurva(NULL, NULL);
    pwm_periodo_min=1000;
    UARTprintf("\nSTEP1\n");

    //CMD_getStep(NULL, NULL);

    UARTprintf("\nSTEP2\n");

    //CMD_getStep(NULL, NULL);

    pwm_periodo_min=1040;
    UARTprintf("\nSTEP1LINEAR\n");
    //CMD_getStep(NULL, NULL);
    UARTprintf("\nSTEP2LINEAR\n");
    //CMD_getStep(NULL, NULL);
    pwm_periodo_min=1000;
    UARTprintf("\nAmostragemVariavel1\n");
    passo=0;
    CMD_iniAmostragem(NULL, NULL);

    UARTprintf("\nAmostragemVariavel2\n\n\n");
    passo=0;
    CMD_iniAmostragem(NULL, NULL);
    UARTprintf("\nFIM::\n\n");
    return 0;
}

int CMD_getPeso(int argc, char **argv) {
    count_amostras=0;
    init_timerCounter();
    start_timerCount();

```

```

int k=0;
for(k=1000;k<=1800;k=k+50) {
    PWM_PERIOD0(k*1e-6);
    SysCtlDelay(SysCtlClockGet());SysCtlDelay(SysCtlClockGet());
    SysCtlDelay(SysCtlClockGet()/2);
    UARTprintf("%u,%u\n",pwm_atual,tempo_final);
}
PWM_PERIOD0(1000*1e-6);
return 0;
}

char g_cInput[64];

int main(void) {
    init_system();
    init_console();

    init_timerAmostragem();

    init_timerCounter();
    init_PWM();

    amostras=malloc(sizeof(uint16_t)*2000);

    int32_t i32CommandStatus;
    UARTEchoSet(true);
    while(1) {
        UARTprintf("\n>");
        if(flag2==1) {
            int i = 0;
            SysCtlDelay(SysCtlClockGet());
            UARTprintf("AMOSTRAS;%d;FREQAMOS;%d;FREQCONT;%d;FIM\n",
                qtd_amostras,freq_amostragem,freq_contador);
            for (i=0;i<qtd_amostras;i++) {
                if(i>150 && i<qtd_amostras) {
                    SysCtlDelay(10000);
                }
                if(amostras[i]!=0) {
                    //amostras[i]= (1/((float)7*amostras[i]*1e-4));
                    //UARTprintf("%d;%u\n",i,amostras[i]);
                } else {
                    //UARTprintf("%d;0\n",i);
                }
            }
            UARTprintf("FIM\n\n");
            flag2=0;
        }
        while(UARTPeek('\r') == -1)
        {
            if(flag2==1) {
                int i = 0;
                SysCtlDelay(SysCtlClockGet());

```



```

UARTprintf(" AMOSTRAS;%d;FREQAMOS;%d;FREQCONT;%d;FIM\n",
           qtd_amostras, freq_amostragem, freq_contador);
for (i=0;i<qtd_amostras;i++) {

    SysCtlDelay(10000);

    if(amostras[i]!=0) {
        //amostras[i]= (1.0/((float)7*amostras[i]*1e-4))
        ;
        //UARTprintf("%d;%u\n",i,amostras[i]);
    } else {
        //UARTprintf("%d;0\n",i);
    }
}
flag2=0;
UARTprintf("FIM\n\n");
}
SysCtlDelay(SysCtlClockGet()/1000);
}

UARTgets(g_cInput, sizeof(g_cInput));

i32CommandStatus = CmdLineProcess(g_cInput);
if(i32CommandStatus == CMDLINE_BAD_CMD)
{
    UARTprintf("ERRO!\n");
}

else if(i32CommandStatus == CMDLINE_TOO_MANY_ARGS)
{
    UARTprintf("ERRO!\n");
}
}

//return 0;
}

```