UNIVERSIDADE DE SÃO PAULO ESCOLA POLITÉCNICA

DEPARTAMENTO DE ENGENHARIA MECÂNICA

ANÁLISE DINÂMICA E PROJETO DE CONTROLE DE UM QUADRICÓPTERO

Felipe Mehsen Tufaile

São Paulo

2017

UNIVERSIDADE DE SÃO PAULO ESCOLA POLITÉCNICA

DEPARTAMENTO DE ENGENHARIA MECÂNICA

ANÁLISE DINÂMICA E PROJETO DE CONTROLE DE UM QUADRICÓPTERO

Trabalho de Formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de Graduação em Engenharia

Felipe Mehsen Tufaile

Prof. Dr. Décio Crisol Donha

Área de Concentração:
Engenharia Mecânica

São Paulo
2017

Catalogação-na-publicação

Mehsen Tufaile, Felipe
Análise Dinâmica e Projeto de Controle de um Quadricóptero/F. Tufaile – São Paulo, 2017.

160 p.

Trabalho de Formatura – Universidade de São Paulo Escola Politécnica. Departamento de Engenharia Mecânica.

1.Controle Moderno. 2.Quadricópteros. 3.Controle Clássico. 4.Controle de Sistemas. I.Universidade de São Paulo.Escola Politécnica. Departamento de Engenharia Mecânica.II.t

RESUMO

Na última década, o uso de quadricópteros tem crescido muito devido à vasta aplicação em atividades comerciais, possibilidade de testar novas teorias de controle, por substituir o ser humano em tarefas consideradas perigosas e até mesmo por ser utilizado para entretenimento pessoal. Por essas razões, este trabalho tem por objetivo estudar a dinâmica e controle de um quadricóptero, com o intuito de projetar um sistema de controle capaz de estabilizar os movimentos de rolagem, arfagem e guinada. Para tanto, serão analisadas algumas técnicas de controle dentro de duas vertentes: controle clássico e controle moderno. Por meio do controle moderno, será realizado o estudo de um sistema MIMO (*Multiple-Input Multiple-Output*) no domínio do tempo. Dessa forma, serão utilizadas técnicas de alocação de polos, análise de controlabilidade e observabilidade, projeto de um observador de estados e análise de estabilidade. Ao final desta parte do trabalho será obtido, portanto, um controlador de ordem completa capaz de estabilizar o quadricóptero. Em segundo lugar, por meio da teoria de controle clássico será realizado o estudo de um conjunto de sistemas SISO (*Single-Input Single-Output*) no domínio da frequência. Dessa forma, serão utilizadas funções de transferência e técnicas de análise de resposta em frequência, com o objetivo de sintetizar um controlador PID. Por fim, o trabalho irá envolver a construção de um protótipo, no qual serão colocados em prática os algoritmos de controle mencionados. Tais algoritmos serão sintetizados com auxílio do ambiente de programação MATLAB® e os resultados finais serão comparados, ao término, com o intuito de avaliar o desempenho de cada controlador.

Palavras-chave: *Controle de sistemas dinâmicos*. *Quadricópteros*. *Controle Clássico*. *Controle Moderno*.

ABSTRACT

For the past decades, the use of quadcopters have become extremely popular for its vast application in commercial activities, for testing new control techniques, for substituting humans in activities considered dangerous and even for personal entertainment. Driven by such motivation, this paper studies some techniques applied for developing control systems for quadcopters stabilization, more specifically this work has the purpose of studying and developing a control algorithm for stabilizing pitch, roll and yaw. For that matter, the purpose of this work will be accomplished by investigating two major control system theories: the modern control theory and the classical control theory. Through the modern control theory, it will be analyzed a multiple-input multiple-output (MIMO) system in the discrete-time domain. Hence, it will be covered the studies of pole-placement, controlability and observability and stability analysis in order to obtain a full-state controller. For the second part of the study, it will be analyzed a set of single-input single-output (SISO) systems in the frequency domain. Therefore, a PID controller will be synthetized through the use of transfer functions and analyses in the frequency domain. Furthermore, the work will be concluded by testing the two controllers developed in a prototype, which will be designed by the student, in order to compare the performance of each algorithm. During this part of the work, all the algorithm developed will be programmed using MATLAB®'s programming environment.

Keywords: *Dynamic Systems Control. Quadcopters. Classical Control Theory. Modern Control Theory*

Lista de ilustrações

Lista de tabelas

Lista de símbolos

O_{XYZ}	Sistema de coordenadas do referencial fixo
$O_{X'Y'Z'}$	Sistema de coordenadas do referencial móvel
u_X	Eixo X do referencial fixo
u_Y	Eixo Y do referencial fixo
u_Z	Eixo Z do referencial fixo
$u_{X'}$	Eixo X' do referencial móvel
$u_{Y'}$	Eixo Y' do referencial móvel
$u_{Z'}$	Eixo Z' do referencial móvel
U	Velocidade linear na direção X', $[m/s]$
V	Velocidade linear na direção Y', $[m/s]$
W	Velocidade linear na direção Z', $[m/s]$
P	Velocidade angular em relação ao eixo "$u_{X'}$", $[rad/s]$
Q	Velocidade angular em relação ao eixo "$u_{Y'}$", $[rad/s]$
R	Velocidade angular em relação ao eixo "$u_{Z'}$", $[rad/s]$
X	Posição na direção X', $[m]$
Y	Posição na direção Y', $[m]$
Z	Posição na direção Z', $[m]$
ϕ	Ângulo de rolagem $[rad]$
θ	Ângulo de arfagem $[rad]$
ψ	Ângulo de guinada $[rad]$
$\dot{\phi}$	Taxa de variação temporal do ângulo de rolagem, $[rad/s]$
$\dot{\theta}$	Taxa de variação temporal do ângulo de arfagem, $[rad/s]$
$\dot{\psi}$	Taxa de variação temporal do ângulo de guinada, $[rad/s]$

$\dot{U}$	Aceleração linear na direção X', $[m/s^2]$
$\dot{V}$	Aceleração linear na direção Y', $[m/s^2]$
$\dot{W}$	Aceleração linear na direção Z', $[m/s^2]$
P	Velocidade angular em relação ao eixo "$u_{X'}$", $[rad/s]$
Q	Velocidade angular em relação ao eixo "$u_{Y'}$", $[rad/s]$
R	Velocidade angular em relação ao eixo "$u_{Z'}$", $[rad/s]$
$\dot{P}$	Aceleração angular em relação ao eixo "$u_{X'}$", $[rad/s^2]$
$\dot{Q}$	Aceleração angular em relação ao eixo "$u_{Y'}$", $[rad/s^2]$
$\dot{R}$	Aceleração angular em relação ao eixo "$u_{Z'}$", $[rad/s^2]$
$\dot{X}$	Derivada no tempo da posição X, $[m/s]$
$\dot{Y}$	Derivada no tempo da posição Y, $[m/s]$
$\dot{Z}$	Derivada no tempo da posição Z, $[m/s]$
R_T	Matriz de rotação
R_ϕ	Matriz de rotação em torno do eixo "$u_{X'}$"
R_θ	Matriz de rotação em torno do eixo "$u_{Y'}$"
R_ψ	Matriz de rotação em torno do eixo "$u_{Z'}$"
F_{ext}	Força externa resultante $[N]$
F_g	Força gravitacional $[N]$
F_p	Força nos propulsores $[N]$
F_{p_x}	Força gerada pelo propulsor na direção X' $[N]$
F_{p_y}	Força gerada pelo propulsor na direção Y' $[N]$
F_{p_z}	Força gerada pelo propulsor na direção Z' $[N]$
M_{ext}	Momento externo resultante $[N.m]$
M_x	Momento ao redor do eixo "$u_{x'}$" $[N.m]$
M_y	Momento ao redor do eixo "$u_{y'}$" $[N.m]$

M_z	Momento ao redor do eixo "$u_{z'}$" $[N.m]$
I_m	Matriz de momento de inércia do quadricóptero $[kg.m^2]$
I_{XX}	Momento de inércia ao redor do eixo "$u_{X'}$" $[kg.m^2]$
I_{YY}	Momento de inércia ao redor do eixo "$u_{Y'}$" $[kg.m^2]$
I_{ZZ}	Momento de inércia ao redor do eixo "$u_{Z'}$" $[kg.m^2]$
I_{XY}	Produto de inércia dos eixos "X" e "Y" $[kg.m^2]$
I_{XZ}	Produto de inércia dos eixos "X" e "Z" $[kg.m^2]$
I_{YZ}	Produto de inércia dos eixos "Y" e "Z" $[kg.m^2]$
m	Massa do veículo $[kg]$
υ	Vetor de velocidade linear $[m/s]$
ω	Vetor de velocidade angular $[rad/s]$
$\times$	Indicação de produto vetorial
g	Aceleração da gravidade $[m/s^2]$
d_{cg}	Distância entre centro de massa e eixo do motor $[m]$
K_{TM}	Coeficiente que relaciona empuxo e momento no motor $[m]$
y_{ss}	Rotação em regime permanente $[rad/s]$
τ	Constante de tempo $[s]$
K_T	Coeficiente que relaciona empuxo e rotação no motor $[rad/s^2\mu]$
k_M	Coeficiente que relaciona momento e rotação no motor $[m]$
a_s	Aceleração medida pelo acelerômetro $[kgm/s^2]$
a_x	Aceleração medida pelo acelerômetro na direção u_x $[m/s^2]$
a_y	Aceleração medida pelo acelerômetro na direção u_y $[m/s^2]$
a_z	Aceleração medida pelo acelerômetro na direção u_z $[m/s^2]$
a_{CG}	Aceleração do centro de gravidade CG $[m/s^2]$
CG	Posição do centro de gravidade $[m]$

T_S	Período de amostragem $[s]$
z_0	Cota do véiculo no estado inicial $[m]$
C_T	Coeficiente de empuxo
ρ	Densidade do ar $[kg/m^3]$
Ω_i	Rotação do i-ésimo motor $[rad/s]$
r	Raio da hélice $[m]$
E_1	Empuxo produzido pelo motor 1 $[N]$
E_2	Empuxo produzido pelo motor 2 $[N]$
E_3	Empuxo produzido pelo motor 3 $[N]$
E_4	Empuxo produzido pelo motor 4 $[N]$
$K_{T\,exp}$	Valor experimental da constante K_T
$K_{T\,teo}$	Valor teórico da constante K_T
K_m	Ganho do motor $[N/s(rad/s)]$
τ_m	Constante de tempo do motor $[s]$
$K_{T\,lin}$	Ganho do motor obtido pela linearização da curva correspondente à dinâmica do motor $[N/s(rad/s)]$
r_S	Posição do sensor em relação ao centro de gravidade $[m]$
r_{Sx}	Posição do sensor em relação ao centro de gravidade na direção $u_{x'}$ $[m]$
r_{Sy}	Posição do sensor em relação ao centro de gravidade na direção $u_{y'}$ $[m]$
r_{Sz}	Posição do sensor em relação ao centro de gravidade na direção $u_{z'}$ $[m]$
g_x	Leitura do giroscópio na direção u_x $[rad/s]$
g_y	Leitura do giroscópio na direção u_y $[rad/s]$
g_z	Leitura do giroscópio na direção u_z $[rad/s]$

N_x	Leitura do magnetômetro na direção u_x $[rad/s]$
N_y	Leitura do magnetômetro na direção u_y $[rad/s]$
N_z	Leitura do magnetômetro na direção u_z $[rad/s]$
η	Vetor de estados
$\dot{\eta}$	Derivada do vetor de estados
A	Matriz de estados no tempo contínuo
B	Matriz de entrada no tempo contínuo
C	Matriz de saída
D	Matriz de alimentação direta
U	Vetor do sinal de entrada
$\mathfrak{C}$	Matriz de controlabilidade
$\mathfrak{O}$	Matriz de observabilidade
K	Matriz de ganhos
R	Vetor de referência
L	Matriz de ganho do observador
e	Vetor *erro*
y	Vetor de saída
$\hat{\eta}$	Vetor de estados estimado
$\hat{y}$	Vetor de saída estimado
G	Matriz de estados no tempo discreto
H	Matriz de entrada no tempo discreto
K_d	Matriz de ganhos no tempo discreto
k_p	Ganho proporcional
k_i	Ganho integrativo
k_d	Ganho derivativo

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo, será realizada uma breve introdução acerca da evolução dos veículos quadrirotores, desde de sua origem a partir de aeronaves tripuladas até as versões mais modernas, as quais são remotamente controladas. Além disso, serão também abordados os últimos avanços tecnológicos desenvolvidos na área de controle de sistemas mecânicos e aplicados no controle de quadricópteros.

1.1 Contexto histórico

Em 1907, sete anos antes do início da primeira Guerra Mundial, os irmãos Jacques e Louis Bréguet, na França, realizaram a primeira tentativa de construir uma aeronave capaz de funcionar com asas rotativas, o chamado *Gyroplane No.1*, ilustrado na Fig.1. Construído com a ajuda do professor Charles Richet, o veículo era formado por estruturas abertas (objetivando maior leveza), um assento para o piloto e um motor de combustão interna de 50 HP de potência (localizados no centro do veículo) e quatro rotores formados por quatro pares de pás (8,1 metros cada), localizados nas extremidades.

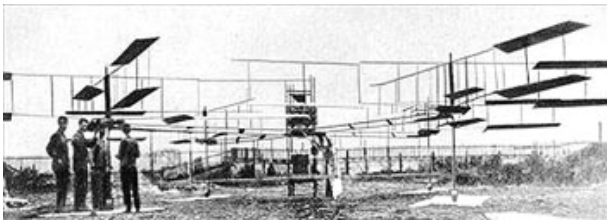


Figura 1 – Bréguet-Richet *Gyroplane No.1.*, (CTIE MONASH UNIVERSITY, 2002)

Durante o trabalho, Jacques e Louis Bréguet realizaram diversos testes, utilizando vários formatos de asas, provando ser possível produzir um veículo com asas rotativas que fosse capaz de realizar voo vertical. Entretanto, apesar haver decolado, a aeronave não permaneceu suspensa por muito tempo devido à sua instabilidade (SHOSA, 2015). Alguns anos após a tentativa dos irmãos Bréguet, Etienne Oemichen, um outro engenheiro Francês, conduziu estudos similares na construção de uma aeronave com asas rotativas. Oemichen conseguiu obter uma aeronave satisfatoriamente estável, capaz de manter voo por pouco mais de catorze minutos e percorrer uma distância de 360 metros (DOMINGUES, 2009). Para tanto, ele utilizou um balão preenchido com gás

hidrogênio, o qual tinha a função de proporcionar estabilidade e sustentação, além de quatro rotores e oito hélices, conforme ilustrado na Fig.2.

Na mesma época, o exército estadunidense também demonstrava grande interesse no projeto de aeronaves capazes de realizar voo vertical. Em 1921, George De Bothezat, contratado pelo *US Army Air Corps*, construiu uma aeronave capaz de pairar à 100 metros do solo (DOMINGUES, 2009). Entretanto, o veículo, ilustrado na Fig.3, também apresentava grandes problemas de estabilidade.

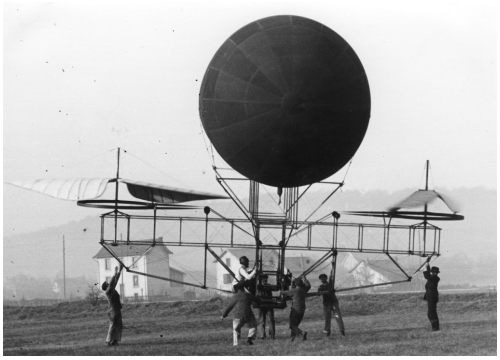


Figura 2 – Oemichen No.2, 1920s (CHALONS-TOURISME, 2016)

Figura 3 – Quadrirotor projetado para o *US Army Air Corps* em 1922 (THE VERTICAL MAGAZINE, 2012)

Pouco mais de 30 anos após o desenvolvimento do quadrirotor de De Bothezat, surge um dos primeiros quadrirotores controlados através da variação do torque nos rotores, o *Convertawings Model A*, apresentado na Fig. 4. O modelo era constituído por estruturas tubulares de aço e alumínio, dois motores e vários sistemas de polias e correias trapezoidais para realizar a variação diferencial da rotação (LAMBERMONT, 1958). Embora o projeto tenha sido um sucesso, a falta de incentivo financeiro fez com que seu desenvolvimento fosse abandonado.

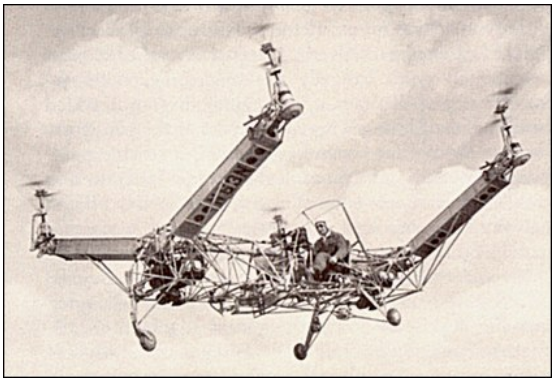


Figura 4 – *Convertawings Model A*, (AVIASTAR, 2011)

No início da década de 80, a necessidade de projetar uma aeronave capaz de realizar decolagens verticais em pequenos espaços, fez com que as empresas *Bell Helicopter* e *Boeing Helicopters* projetassem juntas o *Bell Boeing V-22 Oesprey*, ilustrado na Fig.5. Apesar do primeiro voo realizado por uma aeronave desse modelo ter sido realizado em 1989, as primeiras unidades só foram entregues às Forças Armadas dos Estados Unidos em 2007, por conta de problemas técnicos. Atualmente, a aeronave já foi utilizada no Iraque para operações militares e no Afeganistão, Sudão e Líbia para missões de resgate (BELL HELICOPTER, 2007).

Figura 5 – *Bell Boeing V-22 Osprey*, (BOEING, 2016)

Percebe-se, portanto, que até o fim do século XX, o desenvolvimento de quadrirotores era muito atrelado à demandas militares, sendo necessário grandes investimentos financeiros. Além disso, grande parte dos sistemas de controle desses veículos era mecânico, uma vez que não se dispunha de computadores para projetar sistemas de controle embarcado. No final do século XX e início do século XXI, entretanto, o avanço na microeletrônica e consequente desenvolvimento de microcontroladores e dispositivos eletrônicos de baixo custo fizeram com que a tecnologia e investimento necessários para desenvolver um quadrirotor fossem mais acessíveis. Com isso, inúmeros

projetos de quadrirotores, passaram a ser desenvolvidos utilizando sistema de controle embarcados.

Por essas razões, grande parte do desenvolvimento de quadrirotores foi direcionada ao projeto de veículos aéreos não tripulados (VANTs), os quais, por não necessitarem de uma tripulação (controlados através de ondas de rádio) e por utilizarem componentes eletrônicos, possuem dimensões reduzidas. Um dos exemplos mais populares desses tipos de veículos é o quadricóptero, o qual será foco do estudo realizado neste trabalho. A Fig. 6 ilustra exemplos de alguns modelos de quadricópteros e suas aplicações.

Figura 6 – Quadricópteros e suas aplicações: a) Sistemas de monitoramento (BROWN, 2016); b) Competições estudantis (RAYTHEON, 2015); c) Missões de resgate (ADAM, 2015)

1.2 Motivação

A partir do desenvolvimento dos quadricópteros, os quadrirotores, até então utilizados especialmente para fins militares, sofreram grande mudança quanto à sua aplicabilidade. cada vez esses veículos são vistos realizando atividades como fotografar paisagens em ângulos desafiadores, monitorar tubulações de transporte de petróleo em plataformas, mapear áreas atingidas por desastres naturais, permitir sensoreamento em locais de difícil acesso etc. Os motivos principais dessa tendência são: sua versatilidade, simplicidade construtiva e capacidade de embarcar diversos tipos de sensores e câmeras. Mais recentemente, devido ao desenvolvimento da tecnologia de inteligência artificial, novas aplicações e habilidades têm sido conferidas aos quadricópteros. Entre as principais, destacam-se:

- Aperfeiçoamento de Movimentos: Desenvolvimento de quadricópteros capazes de aprender ações a partir da coleta e análise de dados de um histórico de movimentos. Um exemplo direto dessa habilidade pode ser ilustrado pelo movimento do quadricóptero desenvolvido por Raffaello D'Andrea, o qual, consegue repetir, de forma aprimorada, uma mesma trajetória estabelecida (D'ANDREA, 2016).

A Fig.7 ilustra o aperfeiçoamento do movimento do quadricóptero ao tentar seguir a trajetória definida pelo caminho em vermelho.

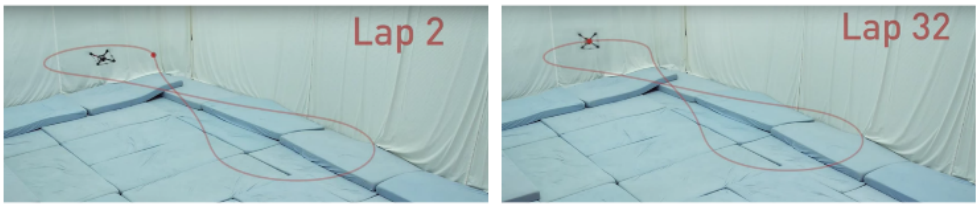


Figura 7 – a) Quadricóptero em sua 2ª volta; b) Quadricóptero em sua 32ª volta; (D'ANDREA, 2016)

- Trabalho Colaborativo: Pesquisas realizadas no desenvolvimento de sistemas de controle que permitem que os quadricópteros realizem tarefas de forma cooperativa. Entre as aplicações mais importantes pode-se destacar o transporte de cargas com peso elevado e o mapeamento do interior de construções, com o objetivo de detectar presença de compostos radioativos (INTERNATIONAL BUSINESS TIMES, 2016), conforme pode ser visto na Fig.8.

Figura 8 – a) Quadricópteros transportando carga, desenvolvido pelo *GRASP Lab na Penn University* (POPULAR SCIENCE, 2010), b) Quadricóptero realizando mapeamento do interior de um galpão (INTERNATIONAL BUSINESS TIMES, 2016)

- Quadricópteros Anfíbios: Quadricópteros desenvolvidos para navegar tanto no ar quanto submerso, conforme ilustrado na Fig.9. A implantação da tecnologia pode ajudar no monitoramento de regiões de vazamento de óleo e em atividades de regaste (BLESCH, 2016).

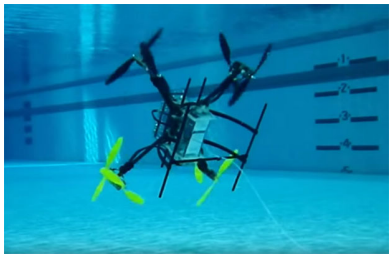


Figura 9 – Quadricóptero desenvolvido pela *Rutgers University*, nos Estados Unidos, adaptado ao ambiente aéreo e submerso (BLESCH, 2016)

- Detecção e detonação de minas terrestres: Quadricópteros construídos com a finalidade de detectar anormalidades no solo que indiquem presença de minas terrestres e detoná-las. O projeto desenvolvido por Massoud Hassani e sua equipe é ilustrado na Fig. 10.

Figura 10 – Quadricópteros desenvolvidos para detectar e detonar minas terrestres (THE VERGE, 2016)

A partir das aplicações mencionadas, pode-se concluir que a importância dessa classe de quadrirotores no avanço da tecnologia é bastante evidente. Logo, a partir desse trabalho, pretende-se construir um quadricóptero e desenvolver seu sistema de controle, tanto pela teoria do controle moderno quanto pela teoria do controle clássico.

1.3 Estado da arte

Conforme mencionado anteriormente, uma das aplicações mais dos quadricópteros é servir como plataforma de testes de novas técnicas de controle. Neste capítulo, portanto, serão mencionadas as técnicas mais recentes aplicadas na tentativa de obter sistemas de controle mais sofisticados, permitindo que estes veículos realizem movimentos mais rápidos e de forma mais precisa.

Um primeiro estudo que deve ser destacado é o desenvolvimento de um algoritmo de controle baseado na técnica de controle Linear Quadrático, por Eryk Nice, em 2004. Em seu trabalho, Nice realizou o controle da posição do quadricóptero por meio da estimativa de sua posição e uso da observação humana para cancelamento do erro de integração (NICE, 2004). O veículo construído como plataforma de teste do algoritmo de controle pode ser visto na Fig.11.

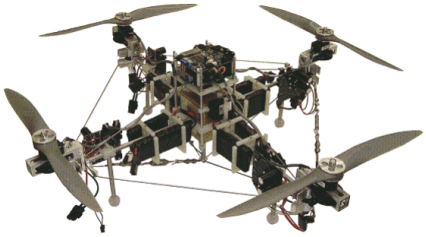


Figura 11 – *AFV* (NICE, 2004)

Ainda no mesmo ano, um estudo comparativo entre a técnica de controle linear quadrática e controle PID, aplicados à micro quadricópteros para vôos em locais fechados, foi realizado por Bouabdallah *et al.*. Por meio da pesquisa, tornou-se evidente a dificuldade em se obter um modelo matemático satisfatório para aplicação da técnica linear quadrática por meio da determinação de matrizes de ponderação. Dessa forma, o desempenho do controlador PID apresentou um resultado mais satisfatório. Entretanto, como a teoria acerca da técnica linear quadrática possui potencial para síntese de controladores muito mais poderosos, ficou evidente a influência dos modelos matemáticos nos sistemas de controle linear quadrático. Portanto, o maior desempenho do controlador PID é justificado pela maior tolerância da técnica de controle para com as incertezas do modelo matemático (BOUABDALLAH *et al.*, 2004). O micro veículo VTOL (decolagem e aterrissagem vertical), construído por Bouabdallah *et al.*, pode ser visto na Fig. 12.

Figura 12 – Micro *VTOL* construído por Bouabdallah *et al.* para análise das técnicas de controle PID e linear quadrática (BOUABDALLAH *et al.*, 2004)

Dois anos mais tarde, Abdellah Mokhtari desenvolveu um quadricóptero com sistema linear de controle H∞ para controle externo, combinando a incerteza do modelo matemático da planta com realimentação robusta linearizada para o controle interno. Para síntese do modelo matemático da dinâmica do sistema foi admitido uma planta não-linear, sendo as incertezas provenientes da variação do momento de inércia e da carga transportada pelo veículo. A partir do trabalho, foi possível analisar que escolhendo as matrizes de peso do sistema de forma criteriosa pode-se obter um controle mais robusto, capaz de superar as diferenças entre o modelo real e teórico da dinâmica do veículo (MOKTHARI, 2006).

Já em 2009, utilizando a teoria de sistema auto-ajustável à interferências (STFIS) baseado na técnica de ordem zero de otimização em tempo real de Takagi-Sugeno, (ZEMALACHE *et al.*, 2009) obtiveram sucesso em projetar um sistema de controle robusto capaz de se auto-estabilizar em condições de falha de um dos atuadores do veículo e presença de forças de arraste provenientes do meio. Para tanto, foi utilizado o modelo de quadricóptero XSF, o qual pode ser visto na Fig.13.

Figura 13 – XSF (ZEMALACHE *et al.*, 2009)

Em 2011, (BERTRAND *et al.*, 2011) projetaram um sistema de controle hierárquico para um veículo aéreo não tripulado, com base na teoria de perturbações singulares. O controle da posição e altitude do quadricóptero foram realizados por meio da implementação de leis de controle que levam em consideração a escala de tempo dos movimentos de translação e orientação de forma separada. Assim, a estabilização da orientação do veículo é considerada em uma escala de tempo mais rápida que sua translação (BERTRAND *et al.*, 2011).

Como pode ser visto, existe uma variedade muito grande de sistemas de controle que têm sido desenvolvidos por meio da aplicação de diferentes técnicas, objetivando obter controle mais precisos dos movimentos e posicionamento dos quadricópteros. Contudo, uma dificuldade comum enfrentada nos trabalhos analisados é a dificuldade em lidar com as incertezas do modelo matemático do veículo, o que leva muitos pesquisadores a projetar algoritmos capazes contornar esse problema.

1.4 Estrutura do trabalho

Conforme mencionado, este trabalho tem por objetivo projetar, implementar e comparar dois sistemas de controle, baseados na teoria de controle moderno e controle clássico, que permitam estabilizar os movimentos de arfagem, rolagem e guinada, além de controlar a posição do veículo no espaço.

Tendo em vista os objetivos a serem atingidos, será primeiramente desenvolvido a modelagem matemática da dinâmica de voo do quadricóptero. Sendo assim, no capítulo dois, serão apresentados os sistemas não-linear e linearizado, admitindo-se as hipóteses apropriadas. Tais hipóteses deverão garantir que o sistema linear possua comportamento suficientemente similar ao sistema não-linear para um determinado ponto de operação.

O capítulo três, por sua vez, irá apresentar a seleção dos componentes utilizados para construção do protótipo do veículo. Dessa forma, será abordado nesse capítulo as características principais dos componentes elétricos (atuadores, bateria, microcontroladores, sensores e ESCs), bem como o critério de sua seleção. Além disso, também será apresentado um modelo virtual do protótipo, construído com o objetivo de calcular algumas propriedades mecânicas de interesse como os momentos de inércia e a posição do centro de massa.

Em seguida, no capítulo quatro, será dado início aos projetos dos sistemas de controle do veículo, sendo primeiramente desenvolvido o sistema de controle com base na teoria do controle moderno. Portanto, será apresentado, neste capítulo, o modelo dinâmico do veículo em espaço de estado, o estudo da controlabilidade e da observabilidade do sistema, a alocação de polos do sistema, o estudo da estabilidade do sistema em malha fechada, e o projeto de um observador de estados.

No capítulo seguinte, será desenvolvido o segundo sistema de controle, baseado na teoria clássica de controle. Logo, no capítulo cinco, serão aplicadas técnicas de controle como análise por funções de transferência, análise das respostas no domínio da frequência (diagrama de Bode) e sintonização dos ganhos derivativo, integral e proporcional.

Após a síntese dos sistemas de controle no domínio do tempo (controle moderno) e no domínio da frequência (Controle clássico), o capítulo seis apresentará a implementação dos controladores, bem como o estudo de um filtro complementar, de modo a possibilitar um sinal de leitura mais confiável.

Por fim, no capítulo sete, serão realizadas comparações no desempenho dos controladores, além de demais observações e conclusões finais.

1.5 Revisão bibliográfica

Nesta seção, serão apresentadas as principais referências utilizadas como embasamento teórico para compreensão da dinâmica de voo do quadricóptero e projeto dos controladores. Cada qual fornece, portanto, uma contribuição fundamental para o desenvolvimento do trabalho.

Em (DOMINGUES, 2009), o princípio de funcionamento e a modelagem matemática da dinâmica do veículo são apresentados, os quais são utilizados como base para formulação do modelo matemático apresentado no segundo capítulo.

Em (PFEIFER, 2013), é apresentado o projeto de construção de um quadricóptero por meio da teoria de controle moderno, utilizando tanto o método de alocação de polos quanto o método linear quadrático.

Em (HENRIQUES, 2011), o autor desenvolve um projeto de controle da atitude de um quadricóptero utilizando apenas sensores de baixo custo como acelerômetro, giroscópio e bússola. Para o sistema de controle, foi utilizado o método linear quadrático de 12 estados.

Em (YOKOTA, 2014), a seleção de componentes e o projeto do sistema de controle de um quadricóptero convencional são utilizados como base para seleção dos componentes elétricos do veículo construído neste trabalho, apresentados no terceiro capítulo. Além disso, a referência também fornece os valores medidos dos parâmetros dos atuadores (ganhos e constantes de tempo), utilizados de forma comparativa neste trabalho.

Em (PACHECO & RESENDE, 2014), é apresentado o projeto teórico de um controlador PID, projetado para atuar como regulador dos movimentos de arfagem, rolagem e guinada do quadricóptero, servindo como referência para o desenvolvimento do controlador projetado no quinto capítulo. Além disso, a obra aborda o uso de um circuito integrado de mensuramento inercial (IMU) para a realimentação da planta, o qual é utilizada no projeto do sistema de medição desenvolvido no sexto capítulo.

Por fim, em (BOUABDALLAH, 2004), o autor realiza um estudo comparativo do desempenho de dua técnicas de controle aplicadas à um quadricóptero: controle linear quadrático (LQ) e controle PID. O estudo contém diversas simulações e testes de desempenho de um mesmo robô, controlado por dois sistemas de controle diferentes. Logo, a obra serve como base para a comparação entre o desempenho dos controladores, desenvolvido no sétimo capítulo.

2 DINÂMICA DO QUADRICÓPTERO

Neste capítulo, será apresentado o princípio de funcionamento do quadricóptero, bem como o sistema de equações que modela a dinâmica de voo do veículo, tanto na forma não-linear quanto na forma linear. A modelagem da dinâmica dos sensores, porém, será reservada para o capítulo três. O resultado do teste do controle linear com o sistema não linear, entretanto, será comentado apenas no capítulo 7.

2.1 Princípio de funcionamento

Conforme mencionado no primeiro capítulo, quadricópteros são veículos constituídos por quatro rotores, fixados nas extremidades de uma estrutura principal, controlados remotamente por um piloto. Para que o sistema possa ser estabilizado, cada um desses atuadores deve ser controlado de forma independente, sendo que rotores vizinhos devem possuir rotação em sentidos opostos, para que o momento que resulta no movimento de guinada do veículo seja nulo quando os rotores estiverem operando em mesma rotação (DOMINGUES, 2009). Respeitando as considerações mencionadas (controle de forma independente dos atuadores e motores vizinhos com rotações opostas) verifica-se possível realizar uma variedade de movimentos, conforme ilustrado na Fig.14, o que permite controlar completamente a atitude do veículo. Na figura em questão, as flechas mais espessas indicam maiores velocidades de rotação e flechas menos espessas, menores. Dessa forma, para que o veículo realize os movimentos de arfagem ou rolagem, é necessário que dois rotores opostos mantenham sua rotação, enquanto os demais sofram alterações graduais.

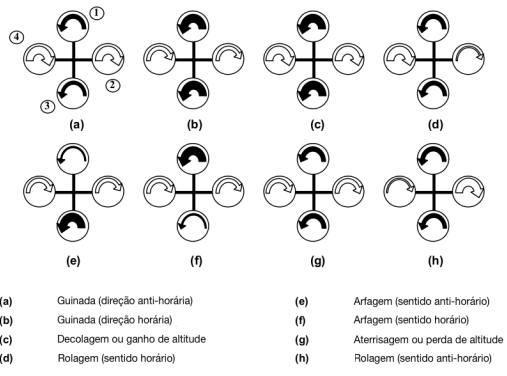


Figura 14 – Esquema de controle de um quadricóptero (PÉREZ-PIÑAR, 2011)

2.2 Hipóteses Simplificadoras

Para obtenção do modelo matemático da dinâmica do quadricóptero, bem como para sua linearização, serão estabelecidas, primeiramente, algumas hipóteses simplificadoras acerca da estrutura física do veículo e da dinâmica dos atuadores. São elas:

- Estrutura do veículo considerada como corpo rígido;
- Referencial móvel ($O_{X'Y'Z'}$) solidário à estrutura do quadricóptero;
- Referencial móvel coincidente com as direções principais de inércia;
- As forças externas devem-se apenas ao empuxo proporcionado pelos atuadores e à ação da gravidade;
- A soma das forças internas ao veículo é nula;
- As forças externas devem-se apenas ao empuxo proporcionado pelos atuadores e à ação da gravidade;
- Assumindo que o veículo irá voar próximo ao solo e sem grandes acelerações, será desprezado qualquer efeito aerodinâmico e termos adicionais como a aceleração Coriolis;
- Será admitido que o veículo irá realizar movimentações que resultem em pequenas variações dos ângulos de Euler. Dessa forma, será considerado que a condição de *"Gimbal Lock"* não será atingida;
- Será admitido que os atuadores possuem dinâmica similares, de modo que suas funções de transferência são as mesmas;
- Será admitido que o ponto de operação dos atuadores correponde à condição em que o empuxo produzido por um atuador equivale a 25% do peso total do veículo, fixando a posição espacial do quadricóptero em uma determinada cota, z_0.

2.3 Modelagem matemática dos movimentos do veículo

Para que seja possível escrever o sistema de equações que regem a dinâmica do quadricóptero, verifica-se necessário definir dois referenciais, um fixo (O_{XYZ}) e um móvel ($O_{X'Y'Z'}$). O referencial móvel, solidário ao quadricóptero, recebe este nome devido à sua mobilidade em relação ao referencial fixo, solidário à terra. Em outras palavras, torna-se possível escrever o movimento do veículo em relação à um observador fixo

na terra. A Fig. 15, ilustra os referenciais mencionados, bem como a convenção dos movimentos angulares do veículo.

Mecanicamente, para definir completamente o estado de um quadricóptero é necessário conhecer 12 variáveis de estado, sendo 6 referentes aos ângulos de Euler [ϕ θ ψ] (ângulos de arfagem, rolagem e guinada, respectivamente) e as respectivas velocidades angulares [P Q R], relacionadas aos eixos [$u_{X'}$ $u_{Y'}$ $u_{Z'}$], e os outros 6 graus de liberdade referentes à posição do centro de massa [X Y Z] e as respectivas velocidades lineares [U V W] em relação ao referencial fixo.

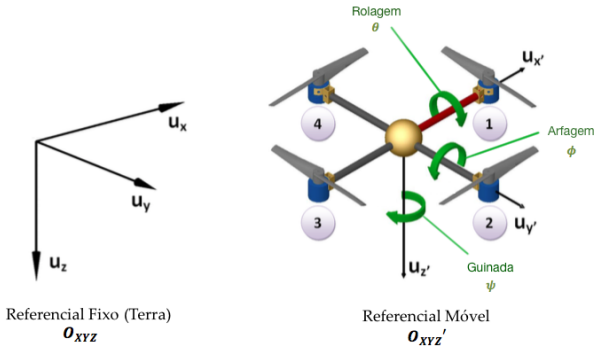


Figura 15 – Sistemas coordenados de referência. Referenciais: fixo e móvel. (DOMINGUES, 2009)

Haja visto a necessidade de descrever a dinâmica do veículo por meio do uso de dois referenciais, será necessário, também, definir uma matriz de rotação, R_T, a qual tem por objetivo descrever a orientação do referencial móvel, $O_{XYZ'}$, em relação ao referencial fixo, O_{XYZ}.

$$R_T(\phi,\theta,\psi) = R_\phi . R_\theta . R_\psi \tag{2.1}$$

Conforme pode ser visto na equação 2.1, a matriz de rotação é formada pela multiplicação de outras três matrizes, na ordem em que são apresentadas, as quais descrevem a rotação do sistema ao redor de um eixo, sendo as matrizes R_ϕ, R_θ e R_ψ descritas a seguir:

$$R_\phi(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \tag{2.2}$$

$$R_\theta(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2.3}$$

$$R_\psi(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

Portanto, a expressão final da matriz de rotação será:

$$R_T(\phi,\theta,\psi) = \begin{bmatrix} \cos(\theta)\cos(\psi) & \cos(\theta)\sin(\psi) & -\sin(\theta) \\ \sin(\phi)\sin(\theta)\cos(\psi) & \cos(\phi)\cos(\psi) & \\ -\cos(\phi)\sin(\psi) & +\sin(\phi)\sin(\theta)\sin(\psi) & \sin(\phi)\cos(\theta) \\ \cos(\phi)\sin(\theta)\cos(\psi) & \sin(\theta)\cos(\phi)\sin(\psi) & \\ +\sin(\phi)\sin(\psi) & -\sin(\phi)\cos(\psi) & \cos(\theta)\cos(\phi) \end{bmatrix} \tag{2.5}$$

2.3.1 Aplicação do TMA e do TMB

Utilizando-se o índice "M" como referência ao sistema de coordenadas do referencial móvel, o índice "F", como referência ao sistema de coordenadas do referencial fixo e aplicando os teoremas do movimento do baricentro (TMB) e movimento angular (TMA), determinam-se as expressões matemáticas que definem o sistema dinâmico do veículo, conforme apresentado a seguir:

$$\frac{d}{dt}[m.\vec{v}]_M + [\vec{\omega}]_M \times [m.\vec{v}]_M = [\vec{F_{ext}}]_M \tag{2.6}$$

$$\frac{d}{dt}[I_m.\vec{\omega}]_M + [\vec{\omega}]_M \times [I_m.\vec{\omega}]_M = [\vec{M_{ext}}]_M \tag{2.7}$$

Sendo as velocidades linear e angular descritas, respectivamente, por:

$$\vec{v} = \begin{bmatrix} U & V & W \end{bmatrix}^t_M \tag{2.8}$$

$$\vec{\omega} = \begin{bmatrix} P & Q & R \end{bmatrix}^t_M \tag{2.9}$$

Além disso, as variáveis m, I_m, $\vec{F_{ext}}$, $\vec{M_{ext}}$, são utilizadas para representar, respectivamente, a massa total do veículo, a matriz de momento de inércia do quadricóptero, a soma das forças externas atuando no veículo e a soma dos momentos externos atuando no veículo. Vale ressaltar que, conforme mencionado, a soma das forças externas corresponde à soma dos empuxos produzidos pelos propulsores, $\vec{F_P}$, com a força gravitacional, $\vec{F_g}$, uma vez que as demais forças serão desprezadas.

$$mR_T \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}_F = mg \begin{bmatrix} -\sin(\theta) \\ \cos(\theta)\sin(\phi) \\ \cos(\theta)\cos(\phi) \end{bmatrix}_M = [\vec{F_g}]_M \tag{2.10}$$

$$\begin{bmatrix} F_{PX} \\ F_{PY} \\ F_{PZ} \end{bmatrix}_M = [\vec{F_P}]_M \tag{2.11}$$

Com isso, a soma das forças externas resultará em:

$$\begin{bmatrix} F_{PX} \\ F_{PY} \\ F_{PZ} \end{bmatrix}_M + mg \begin{bmatrix} -\sin(\theta) \\ \cos(\theta)\sin(\phi) \\ \cos(\theta)\cos(\phi) \end{bmatrix}_M = [\vec{F_{ext}}]_M \tag{2.12}$$

Em seguida, calculando os demais termos da expressão 2.6, tem-se:

$$\frac{d}{dt}[m.\vec{v}]_M = m \begin{bmatrix} \dot{U} \\ \dot{V} \\ \dot{W} \end{bmatrix}_M \tag{2.13}$$

$$[\vec{\omega}]_M \times [m.\vec{v}]_M = m \begin{bmatrix} QW - RV \\ RU - PW \\ PV - QU \end{bmatrix}_M \tag{2.14}$$

Combinando as equações 2.12 e 2.6 o vetor de aceleração que atua no corpo do veículo resulta em:

$$\begin{bmatrix} \dot{U} \\ \dot{V} \\ \dot{W} \end{bmatrix}_M = \frac{1}{m} \begin{bmatrix} F_{PX} \\ F_{PY} \\ F_{PZ} \end{bmatrix}_M + g \begin{bmatrix} -\sin(\theta) \\ \cos(\theta)\sin(\phi) \\ \cos(\theta)\cos(\phi) \end{bmatrix}_M - \begin{bmatrix} QW - RV \\ RU - PW \\ PV - QU \end{bmatrix}_M \tag{2.15}$$

Utilizando as hipóteses de que a estrutura do quadricóptero é rígida, com massa constante, e que seus eixos estão alinhados com os eixos principais de inércia, conclui-se que a matriz de inércia possuirá elementos apenas em sua diagonal principal.

$$[I_m]_M = \begin{bmatrix} I_{XX} & 0 & 0 \\ 0 & I_{YY} & 0 \\ 0 & 0 & I_{ZZ} \end{bmatrix}_M \tag{2.16}$$

Além disso, considerando que a matriz de inércia do sistema não varia com o tempo:

$$\frac{d}{dt}[\vec{\omega}]_M = \begin{bmatrix} \dot{P} \\ \dot{Q} \\ \dot{R} \end{bmatrix}_M \tag{2.17}$$

Calculando os demais termos da expressão 2.7, chega-se em:

$$\frac{d}{dt}[I_m\vec{\omega}]_M = \begin{bmatrix} I_{XX}\dot{P} \\ I_{YY}\dot{Q} \\ I_{ZZ}\dot{R} \end{bmatrix}_M \tag{2.18}$$

$$[\vec{\omega}]_M \times [I_m.\vec{\omega}]_M = \begin{bmatrix} (I_{ZZ} - I_{YY})QR \\ (I_{XX} - I_{ZZ})RP \\ (I_{YY} - I_{XX})PQ \end{bmatrix}_M \tag{2.19}$$

Logo, a aplicação do teorema do momento angular resulta em:

$$\begin{bmatrix} \dot{P} \\ \dot{Q} \\ \dot{R} \end{bmatrix}_M = \begin{bmatrix} \frac{M_X}{I_{XX}} \\ \frac{M_Y}{I_{YY}} \\ \frac{M_Z}{I_{ZZ}} \end{bmatrix}_M - \begin{bmatrix} \frac{(I_{ZZ}-I_{YY})QR}{I_{XX}} \\ \frac{(I_{XX}-I_{ZZ})RP}{I_{YY}} \\ \frac{(I_{YY}-I_{XX})PQ}{I_{ZZ}} \end{bmatrix}_M \tag{2.20}$$

2.3.2 Análise cinemática

As demais equações do sistema serão obtidas por meio da análise cinemática do movimento do quadricóptero. Sendo assim, será realizado a derivada das velocidades, anulgar e linear, com o intuito de obter as respectivas velocidades. Para tanto, será primeiro definido o vetor $\vec{r}$, o qual representa a posição do sistema de coordenadas móvel em relação ao sistema de coordenadas fixo.

$$\vec{r} = X\vec{i} + Y\vec{j} + Z\vec{k} \tag{2.21}$$

Derivando o vetor apresentado na equação 2.19 com relação ao tempo, obtém-se o vetor da velocidade do referencial móvel em relação ao referencial fixo.

$$\dot{\vec{r}} = \dot{X}\vec{i} + \dot{Y}\vec{j} + \dot{Z}\vec{k} \tag{2.22}$$

Logo, para determinar a velocidade do veículo em relação ao referencial fixo deve-se utilizar aplicar a matriz de rotação.

$$\begin{bmatrix} U \\ V \\ W \end{bmatrix}_M = R_T \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix}_F \tag{2.23}$$

Uma vez que a matriz R_T é ortogonal, ou seja, $R_T.{R_T}^t = I$, onde I é a matriz identidade, conclui-se que a equação 2.20 pode ser rescrita resultando em:

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix}_F = {R_T}^t \begin{bmatrix} U \\ V \\ W \end{bmatrix}_M \tag{2.24}$$

Substituindo a matriz de rotação transposta, a expressão 2.24 resulta em:

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix}_F = \begin{bmatrix} \cos(\theta)\cos(\psi) & \begin{matrix}\sin(\phi)\sin(\theta)\cos(\psi)\\ -\cos(\phi)\sin(\psi)\end{matrix} & \begin{matrix}\cos(\phi)\sin(\theta)\cos(\psi)\\ +\sin(\phi)\sin(\psi)\end{matrix} \\ \cos(\theta)\sin(\psi) & \begin{matrix}\cos(\phi)\cos(\psi)\\ +\sin(\phi)\sin(\theta)\sin(\psi)\end{matrix} & \begin{matrix}\sin(\theta)\cos(\phi)\sin(\psi)\\ -\sin(\phi)\cos(\psi)\end{matrix} \\ -\sin(\theta) & \sin(\phi)\cos(\theta) & \cos(\theta)\cos(\phi) \end{bmatrix} \begin{bmatrix} U \\ V \\ W \end{bmatrix}_M \tag{2.25}$$

Por fim, verifica-se que para determinar o vetor posição do quadricóptero em relação ao referencial fixo, por meio de integração, é necessário conhecer as posições angulares ϕ, θ e ψ.

Estas últimas variáveis, por outro lado, variam com o tempo, sendo que suas taxas de variação ($\dot{\phi}$, $\dot{\theta}$ e $\dot{\psi}$) dependem do vetor de velocidade angular $\vec{\omega}$. Portanto, para o último conjunto de equações pode ser determinado analisando a relação entre a taxa de variação da posição angular entre os referenciais fixo e móvel e o vetor $\vec{\omega}$. Contudo, a determinação desta relação é mais complicada e deverá ser descrita conforme equação 2.26.

$$\begin{bmatrix} P \\ Q \\ R \end{bmatrix}_M = R_T \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}_F + R_\phi R_\theta \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix}_F + R_\phi \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix}_F \tag{2.26}$$

Substituindo os termos da equação anterior pelas termos apresentados nas equações 2.5, 2.2 e 2.3, obtém-se:

$$\begin{bmatrix} P \\ Q \\ R \end{bmatrix}_M = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\theta\cos\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}_F \tag{2.27}$$

Em seguida, manipulando algebricamente a equação 2.27, chega-se em:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}_F = \begin{bmatrix} 1 & \tan\theta\sin\phi & \tan\theta\cos\phi \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} P \\ Q \\ R \end{bmatrix}_M \tag{2.28}$$

A expressão 2.28, em conjunto com as expressões 2.15, 2.20 e 2.25 formam o sistema de equações, não lineares, que descrevem a dinâmica do quadricóptero. Através da solução simultânea das 12 equações obtidas, tem-se conhecimento das 12 variáveis de estado do problema de controle. Entretanto, convém realizar alguma observações importantes.

Primeiramente, sabe-se que o desempenho de um sistema de controle será tão melhor quanto melhor o modelo matemático desenvolvido para descrever sua dinâmica. Em outras palavras, modelos matemáticos muito simplificados, consequência do uso de hipóteses que distanciam muito o modelo matemático da dinâmica real, resultam em desempenho ruim. No caso do modelo matemático descrito nesse capítulo, a dinâmica do veículo só será válida para a condição sob a qual foi modelado, conforme hipóteses mencionadas no início do capítulo.

Um outro aspecto importante pode ser observado na equação 2.28, onde o termo $\cos(\theta)$ aparece no denominador. Isso indica que quando θ for igual a 90 graus, ocorrerá uma indeterminação do sistema. Tal fenômeno é conhecido como *"gimbal lock"*. Para contornar esse problema, utilizam-se equações diferenciais quaternárias. Entretanto, como o estudo desenvolvido nesse trabalho não prevê estudar o movimento do quadricóptero pra grandes variações dos ângulos de Euler, a condição de *"gimbal lock"* não deverá ser atingida durante uma manobra do veículo.

2.4 Modelagem matemática dos atuadores

Conforme mencionado anteriormente, os motores são posicionados em um quadricóptero, de forma que motores vizinhos possuam rotação em sentidos contrários, como pode ser visto na Fig. 16. Na figura, verifica-se que os motores ímpares, posicionados possuem rotação no sentido horário, enquanto que os motores pares possuem rotação no sentido anti-horário. Dessa forma, para uma mesma velocidade de rotação, o momento produzido pelos motores se anulam.

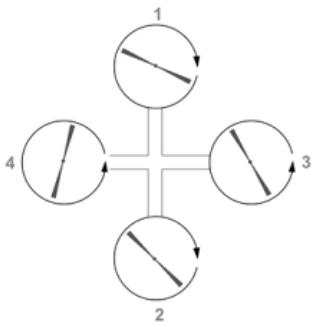


Figura 16 – Esquema de configuração dos atuadores

Com base no esquema apresentado na Fig. 15 e denominando o empuxo produzido pelo i-ésimo atuador como E_n, o vetor $\vec{F_P}$, o qual representa o empuxo produzido pelo conjunto de atuadores, pode ser escrito da seguinte forma:

$$[\vec{F_P}]_M = \begin{bmatrix} F_{Px} \\ F_{Py} \\ F_{Pz} \end{bmatrix}_M = \begin{bmatrix} 0 \\ 0 \\ -(E_1+E_2+E_3+E_4) \end{bmatrix}_M \tag{2.29}$$

De acordo com (DOMINGUES, 2009) o torque produzido por um motor é proporcional ao quadrado de sua rotação, conforme apresentada na expressão a seguir:

$$E_i = C_T \frac{4\rho r^4}{\pi^2} {\Omega_i}^2 \tag{2.30}$$

Onde, C_T é coeficiente de empuxo, ρ densidade do ar, r raio da hélice, Ω_i rotação do i-ésimo propulsor. Deve-se destacar que o valor do coeficiente C_T bem como o valor da densidade do ar ρ são encontrados na refereência (DOMINGUES, 2009).

Ainda de acordo com a mesma referência, fixando-se o fluído (ar) e a hélice propulsora, a equação 2.27 pode ser reescrita substituindo os termos constantes por K_T.

$$E_i = K_T {\Omega_i}^2 \tag{2.31}$$

De forma análoga, o momento, $[\vec{M_P}]_M$, produzido pelos motores pode ser calculado utilizando-se os empuxos conforme expressão a seguir:

$$[\vec{M_P}]_M = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix}_M = \begin{bmatrix} (E_4-E_2)d_{cg} \\ (E_1-E_3)d_{cg} \\ (E_1+E_3-E_2-E_4)K_{TM} \end{bmatrix}_M \tag{2.32}$$

Onde d_{cg} é a distância entre o motor e o centro de gravidade e K_{TM} é a constante que relaciona o momento e o empuxo produzido por um motor. O termo K_{TM}, por sua vez, é obtido pela divisão entre as constantes K_T e "K_M", sendo que este último termo é calculado conforme expressão a seguir:

$$K_M = C_P \frac{4\rho r^5}{\pi^3} \tag{2.33}$$

Substituindo as variáveis apresentados nas expressões 2.30 e 2.33 pelos valores obtidos na referência mencionada (c_T = 0,1154, c_P = 0,0743, ρ = 1.225 $[kg/m^3]$ e r = 0,127 $[m]$), chega-se aos seguintes valores de K_T, K_M e K_{TM}:

$$K_T = 0,1154 \frac{4(1,225)(0,127)^4}{\pi^2} = 1,46.10^{-5}[kg.m/rad^2] \tag{2.34}$$

$$K_M = 0,0743 \frac{4(1,225)(0,127)^5}{\pi^3} = 3,8.10^{-7}[kg.m^2/rad^2] \tag{2.35}$$

$$K_{TM} = \frac{3,8.10^{-7}}{1,46.10^{-5}} = 0,026[m] \tag{2.36}$$

2.5 Sistema linear e sistema linearizado

Com o intuito de obter um sistema de equações que possa ser utilizado mais facilmente, será realizado, nessa seção, a linearização do sistema por meio da expansão em séries de Taylor.

Conforme visto na seção 2.3, as variáveis de interesse de interesse do problema são:

$$\begin{bmatrix} X & Y & Z & U & V & W & \phi & \theta & \psi & P & Q & R \end{bmatrix}^t \tag{2.37}$$

Dessa forma, o sistema matemático que descreve a dinâmica do veículo pode ser escrito conforme a expressão a seguir:

$$
\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \\ \dot{U} \\ \dot{V} \\ \dot{W} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{P} \\ \dot{Q} \\ \dot{R} \end{bmatrix} =
\begin{bmatrix}
(\cos\theta\cos\psi)U + (\sin\psi\sin\theta\cos\psi - \cos\phi\sin\psi)V + (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi)W \\
(\cos\theta\sin\psi)U + (\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi)V + (\sin\theta\cos\phi\sin\psi - \sin\phi\cos\psi)W \\
-(sin\theta)U + (\sin\phi\cos\theta)V + (\cos\theta\cos\phi)W \\
\frac{F_{p_x}}{m} - g\sin(\theta) - (QW - RV) \\
\frac{F_{p_y}}{m} + g\cos(\theta)\sin(\phi) - (RU - PW) \\
\frac{F_{p_z}}{m} + g\cos(\theta)\cos(\phi) - (PV - QU) \\
P + (\tan\theta\sin\phi)Q + (\tan\theta\cos\phi)R \\
(\cos\phi)Q - (\sin\phi)R \\
(\frac{\sin\phi}{\cos\theta})Q + (\frac{\cos\phi}{\cos\theta})R \\
\frac{(\Omega_4^2 - \Omega_2^2)K_T d_{cg}}{I_{XX}} - \frac{(I_{ZZ} - I_{YY})QR}{I_{XX}} \\
\frac{(\Omega_1^2 - \Omega_3^2)K_T d_{cg}}{I_{YY}} - \frac{(I_{XX} - I_{ZZ})RP}{I_{YY}} \\
\frac{(\Omega_1^2 + \Omega_3^2 - \Omega_2^2 - \Omega_4^2)K_{TM}}{I_{ZZ}} - \frac{(I_{YY} - I_{XX})PQ}{I_{ZZ}}
\end{bmatrix} \tag{2.38}
$$

Utilizando a expansão em série de Taylor, apresentada a seguir, o sistema definido em 2.38, pode ser linearizado em torno de um ponto de operação,$[X_0]$ e $[U_0]$, os quais

serão definidos no próximo capítulo.

$$\begin{bmatrix} \dot{y}_i \\ \vdots \\ \dot{y}_n \end{bmatrix} = \begin{bmatrix} f_i(x_{10},\cdots,x_{j0}) + (\frac{\partial f_i(x_1,\cdots,x_j)}{\partial x_1})(x_1 - x_{10}) \\ +\cdots+ (\frac{\partial f_i(x_1,\cdots,x_j)}{\partial x_j})(x_j - x_{10}) + T.O.S. \\ \vdots \\ f_n(x_{10},\cdots,x_{j0}) + (\frac{\partial f_n(x_1,\cdots,x_j)}{\partial x_1})(x_1 - x_{10}) \\ +\cdots+ (\frac{\partial f_n(x_1,\cdots,x_j)}{\partial x_j})(x_j - x_{10}) + T.O.S. \end{bmatrix} \tag{2.39}$$

$$\begin{bmatrix} X_0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & z_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^t \tag{2.40}$$

$$\begin{bmatrix} U_0 \end{bmatrix} = \begin{bmatrix} \Omega_1^0 & \Omega_2^0 & \Omega_3^0 & \Omega_4^0 \end{bmatrix}^t \tag{2.41}$$

Portanto, assumindo que os motores possuem a mesma dinâmica, ou seja, $\Omega_1^0 = \Omega_2^0 = \Omega_3^0 = \Omega_4^0 = \Omega_0$, o sistema 2.38 pode ser escrito na forma linear conforme expressõa a seguir:

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \\ \dot{U} \\ \dot{V} \\ \dot{W} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{P} \\ \dot{Q} \\ \dot{R} \end{bmatrix} = \begin{bmatrix} U \\ V \\ W \\ -g\theta \\ +g\phi \\ \frac{4[(\Omega_0)^2]K_T + 2\Omega_0[\Omega_1+\Omega_2+\Omega_3+\Omega_4]K_T}{m} + g \\ P \\ Q \\ R \\ \frac{2[\Omega_4-\Omega_2]K_T d_{cg}\Omega_0}{I_{XX}} \\ \frac{2[\Omega_1-\Omega_3]K_T d_{cg}\Omega_0}{I_{YY}} \\ \frac{2[\Omega_1+\Omega_3-\Omega_2-\Omega_4]K_{TM}\Omega_0}{I_{ZZ}} \end{bmatrix} \tag{2.42}$$

Como pode ser visto na equação 2.42, a complexidade matemática do sistema linearizado é muito menor. Dessa forma, a aplicação das teorias de controle utilizando

o sistema linearizado tornam-se mais simples. Entretanto, a validade do modelo matemático é restringida a determinada faixa de operação, próxima do ponto de operação definido durante a linearização. Além disso, deve-se mencionar que a linearização do sistema não garante desempenho, apenas estabilidade. Logo, para sistema cujo modelo matemático não apresenta bom desempenho, deve-se realizar técnicas alternativas de controle, como controle adaptativo (CABRAL, 2016).

3 SELEÇÃO DE COMPONENTES

Neste capítulo, será apresentado o protótipo construído como plataforma de teste dos sistema de controle projetados, bem como a seleção dos componentes utilizados. Também serão realizados neste capítulo a identificação da dinâmica dos atuadores, o equacionamento da dinâmica dos sensores e a construção de um modelo virtual do protótipo, utilizado para determinar a matriz de inércia e posição do centro de gravidade do veículo.

3.1 Seleção dos atuadores

Para realizar a escolha dos motores, foram consultados sites amadores de veículos radio controlados, os quais contém uma quantidade vasta de informação sobre os modelos comumente utilizados, além das referências (DOMINGUES, 2009), (PFEIFER, 2013) e (YOKOTA, 2014). A partir das informações obtidas e de uma estimativa inicial do massa do veículo completo (1,2 kg), foi escolhido o motor mais barato capaz de contrabalancear, em condição moderada de trabalho, 25% do peso do veículo. As características dos motores escolhidos são detalhadas na Tab. 1. A Fig. 17 ilustra o motor utilizado.

Tabela 1 – Características do motor DC (*Brushless*)

Kv:	1000RPM/V
Eficiência Máxima:	80%
Corrente de Eficiência Máxima:	4-10A (>75%)
Corrente Máxima:	13A para 60S
Potência Máxima:	150W
Massa:	52,7g
Dimensões:	28mm de diâmetro
Diâmetro do eixo:	3,2 mm
Número de polos:	14

Figura 17 – Ilustração do Motor CC (*Brushless*) utilizado como atuador.

3.1.1 Seleção da hélice

O modelo de hélice escolhido é o mesmo que o utilizado em (DOMINGUES, 2009), com o intuito de aproveitar o valor calculado para as constantes K_M, K_T e K_{TM}, além de aproveitar outras análises desenvolvidas na referência como o valor do empuxo gerado a partir de uma determinada rotação. A Fig. 18 ilustra o modelo de hélice (EPP 1045) escolhido para construção dos atuadores, enquanto a descrição de suas características é fornecida na Tab. 2.

Figura 18 – Ilustração das hélices (modelo EPP 1045) utilizadas no protótipo.

Tabela 2 – Características da Hélice

Modelo:	EPP 1045 CW/CCW
Dimensões:	10x4,5”
Material:	Carbono-Nylon
Massa:	28g

3.1.2 Seleção dos controladores eletrônicos de velocidade (ESC)

Para realizar o controle de velocidade dos motores, foram utilizados quatro controladores eletrônicos de velocidade (ESC), os quais permitem que seja variada a velocidade dos motores de forma mais precisa e eficiente, por meio do envio de sinais eletrônicos pelo microcontrolador. Na construção do protótipo foram utilizados quatro

ESCs com o "*Firmware*"SimonK, os quais, de acordo com algumas referências, são mais rápidos no controle de veículos multi-rotores. A ilustração do componente escolhido é apresentada na Fig. 19 enquanto que as características dos componentes são apresentadas na Tab. 3.

Figura 19 – Ilustração do ESC: "*30A SimonK Firmware ESC*"

Tabela 3 – Características dos *Eletronic Speed Controllers*

Firmware:	*SimonK*
Corrente Máxima:	30 A
Corrente de Pico:	40 A (10s)
Tensão de alimentação:	5V
Massa:	25g
Dimensões:	50x25x8mm

É importante ressaltar que os ESCs selecionados possuem a tecnologia de proteção contra baixa tensão, que evita que a bateria possar ser danificada caso sua carga seja drenada para valores inferiores a um valor mínimo. No caso dos ESCs selecionados, o valor de tensão mínimo é de 3.4 V por célula. Caso alguma célula permaneça com sua carga inferior à 3.4 V por 2 segundos, o fornecimento de tensão aos motores é interrompido por meio do dispositivo LVC (Interrupção para Baixas Tensões).

3.2 Ensaio dos atuadores

Após selecionados os componentes do sistema de atuação, foram realizados ensaios com o intuito de modelar a dinâmica dos atuadores, bem como validar os dados reais do conjunto motor-hélice, por meio da comparação com os resultados encontrados em referências. Com isso, foram analisadas as medidas de empuxo em função das corrente e da rotação do motor. Para realização dos ensaios, foi desenvolvida uma bancada de testes utilizando sensores óticos (LEDs emissores e receptores de onda infravermelha), uma fonte de 12 V de tensão com limitação de corrente em 30 A, um atuador (Motor Brushless, ESC e hélice) e um microcontrolador (Arduino UNO). A

bancada construída pode ser vista na Fig. 20, sendo que o projeto do circuito elétrico encontra-se em anexo a este relatório.

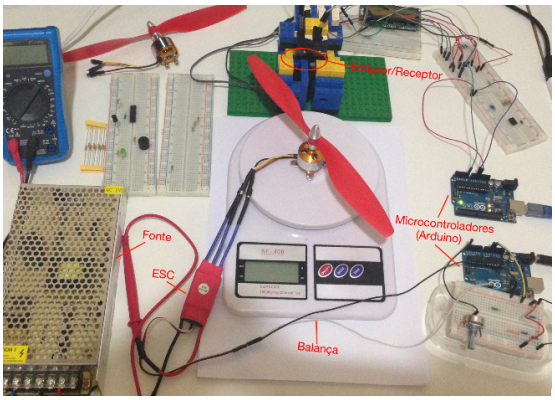
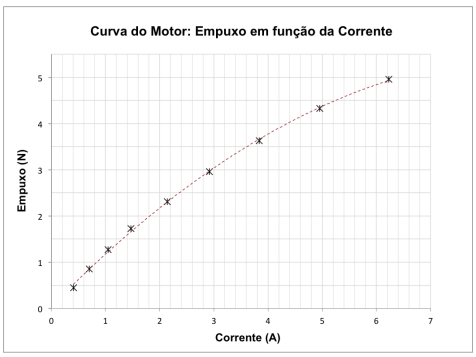


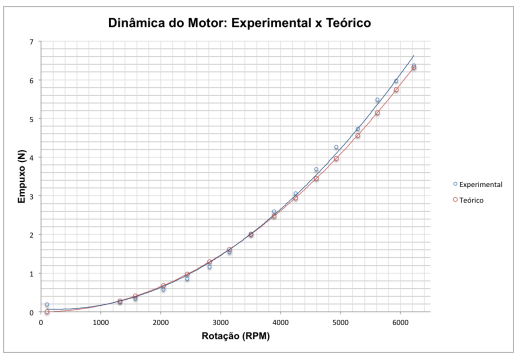
Figura 20 – Bancada de testes utilizada para identificação da dinâmica do motor

O princípio de funcionamento do equipamento de medição é bastante simples e baseia-se na interrupção da transmissão de sinais de onda infra-vermelho entre o emissor e receptor. Cada vez que a hélice passa entre os emissores e os receptores, conforme pode ser visto na Fig.20, o sinal é interrompido, sendo esta interrupção detectada pelo microcontrolador. Além disso, o microcontrolador também tem por função enviar um sinal específico ao ESC, configurando uma determinada rotação. Portanto, variando-se o sinal enviado pelo microcontrolador ao ESC, varia-se a resposta do motor.

A partir dos dados obtidos, foram estudadas a variação do empuxo em função da corrente enviada ao motor e a variação do empuxo em função da rotação, permitindo determinar o consumo médio de corrente para que o VANT permaneça pairado e o modelo matemático da dinâmica do atuador. A Fig. 21(a) ilustra a curva do empuxo em função da corrente enviado ao motor enquanto que a Fig. 21(b) ilustra a curva do empuxo em função da rotação. A Tab. 4 detalha o conjunto de valores obtidos por meio do teste realizado em um dos atuadores.



(a) Empuxo em função da corrente



(b) Empuxo em função da rotação

Figura 21 – Curvas do atuador obtidas através de ensaio

Analisando os gráficos da Fig. 21, verifica-se que a máxima rotação do motor é próxima de 650 rad/s (6220 rpm) enquanto que o máximo empuxo produzido é próximo de 6,4 N. Além disso, verifica-se que o motor não produz resposta para rotação inferiores à 10,3 rad/s (98,34 RPM) o que define sua ”Zona Morta”. De forma semelhante, observa-se que para rotações superiores à 650 rad/s (6220 rpm), a variação no empuxo produzido é muito pequena. Tal fenômeno é conhecido como saturação do motor. Fica evidente, portanto, que a faixa de funcionamento do motor é delimitada por um limite inferior (”zona morta”) e um limite superior (região de saturação do motor).

Ainda com relação à Fig. 21(b), comparando a curva experimental (curva azul), com a curva teórica levantada a partir da Eq. 2.30 (curva vermelha), verifica-se uma correspondência satisfatória. Além disso, calculando o valor experimental da constante K_T, $K_{Texp} = 1,55x10^{-05}$, verifica-se um erro de apenas 4% do valor teórico, $K_{Tteo} = K_T = 1,49x10^{-05}$.

Tabela 4 – Dados obtidos por meio do ensaio de um dos atuadores

Empuxo (g)	Empuxo (N)	PWM (μs)	Rotação (rad/s)
20	0,20	1070	10,30
27	0,26	1080	138,15
35	0,34	1100	165,20
59	0,58	1150	213,87
87	0,85	1200	255,56
119	1,17	1250	294,81
157	1,54	1300	329,26
204	2,00	1350	366,88
265	2,60	1400	407,37
312	3,06	1450	444,80
376	3,69	1500	481,17
435	4.27	1550	516,25
483	4,74	1600	553,66
560	5,49	1650	587,74
609	5,97	1700	621,16
650	6,38	1750	651,43

Considerando a estimativa da massa total do veículo em 1,2 kg e assumindo que em condição de regime cada atuador será responsável por produzir $\frac{1}{4}$ do empuxo necessário para que o veículo seja capaz de pairar (aproximadamente 3 N de empuxo por atuador), a rotação em que cada motor deverá trabalhar será aproximadamente 445 rad/s. Portanto, o ponto de operação de cada motor irá corresponder à rotação de 445

rad/s (Ω_0 = 445 rad/s). Uma vez determinado o ponto de operação do sistema, pode-se determinar os parâmetros das funções de transferência que descrevem a dinâmica do motor para condições de operação próximas desse ponto.

Neste trabalho, assim como em muitas referências, a dinâmica do motor será descrita por meio de uma função de transferência de primeira ordem, assumindo que a constante de tempo mecânica do motor é muito maior que a constante de tempo elétrica. Sendo assim, dinâmica do motor será descrita conforme expressão a seguir:

$$G(s) = \frac{E(s)}{\Omega(s)} = \frac{K_m}{\tau_m s + 1} \tag{3.1}$$

Resta, portanto, determinar o valor da constante de tempo τ_m e do ganho K_m.

Para determinar o valor da constante de tempo τ_m, serão analisadas as medidas de rotação em função do tempo, até que os motores alcancem a condição de regime, conforme ilustrado na Fig. 22.

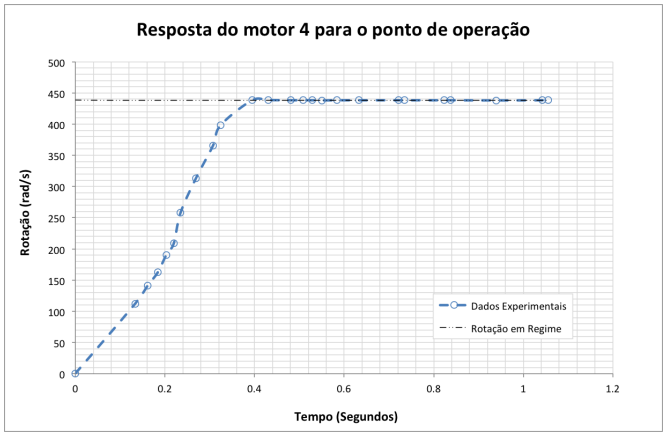


Figura 22 – Resposta do motor ao sinal de entrada correspondente ao ponto de operação

Admitindo que, para os ensaios dos motores, a condição inicial dos motores seja o repouso e que o sistema se comporta como um sistema de primeira ordem, a constante de tempo pode ser calculada a partir da equação a seguir:

$$\frac{y(\tau)}{y_{ss}} = 0,632 \tag{3.2}$$

Onde "y_{ss}" o valor da rotação na condição de regime.

Utilizando a Eq.3.2 na curva da Fig.22, chega-se ao valor da constante de tempo igual 265 milisegundos (0,265 segundos).

O ganho, por sua vez, pode ser determinado calculando-se o coeficiente angular da reta tangente à curva do motor, no ponto de operação, conforme apresentado na Fig. 23. Com isso, o valor obtido para o ganho, $K_{T\,lin}$, é igual 0.0159 $\frac{N}{s(rad/s)}$.

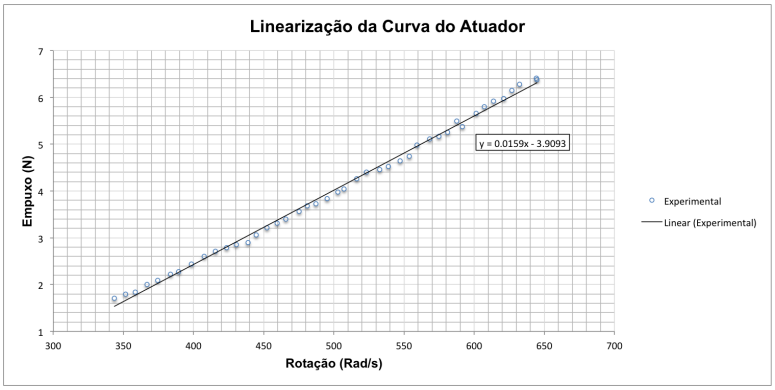


Figura 23 – Curva linearizada da dinâmica do motor no ponto de operação (Empuxo em função da rotação)

3.3 Seleção da bateria

Para escolha da bateria adequada para construção do protótipo foram consultados alguns sites sobre veículos rádio controlados, os quais contam com diversas regras práticas para seleção de baterias, de acordo com a forma de uso e propósito do veículo. A seguir, será mencionado o método prático encontrado na referência (OSCARLIANG, 2017). É importante ressaltar, porém, que a escolha de uma bateria do tipo LiPo foi priorizada, uma vez que essas baterias são recarregáveis, leves e de custo compatível com o propósito do projeto.

- Cálculo estimado da corrente necessária para funcionamento do VANT:

  O primeiro dado importante que deve ser estimado é a corrente média que deverá ser fornecida pela bateria, a qual pode ser calculada com base nos dados obtidos no ensaio dos motores. Conforme mencionado, no ponto de operação, cada motor deverá fornecer, aproximadamente, 3N de empuxo. Com isso, analisando o gráfico da Fig. 21(a), verifica-se que cada motor irá consumir, em média, 3,5 A de corrente. Além disso, de acordo com os sites mencionados, deve-se adicionar 1,0 A à corrente total fornecida aos motores para contabilizar o consumo de

corrente pelos demais componentes eletrônicos. Dessa forma, o consumo total de corrente será próximo de 15 A.

- Cálculo da corrente máxima fornecida pela bateria:

  O próximo passo no método mencionado consiste em determinar a taxa de descarga da bateria de modo que esta seja capaz de fornecer a corrente exigida. De acordo com a referência, a máxima corrente fornecida pela bateria pode ser calculada multiplicando-se a capacidade da bateria (em Ah) por sua taxa de descarga (em C). Com isso, para determinação da taxa de descarga, a equação a seguir, deve ser satisfeita.

$$Consumodecorrente(A) < Capacidade(Ah).Taxadedescarga(C) \quad (3.3)$$

  Escolhendo, por exemplo, uma bateria com 5200 mAh de capacidade e 10 C de taxa de descarga, verifica-se que a corrente máxima que pode ser fornecida pela bateria será 52 A. Isto significa que a corrente total exigida pelos componentes do VANT poderá atingir o valor máximo de 52 A sem que a bateria seja danificada.

- Seleção do número de células:

  O terceiro passo para seleção da bateria é determinar seu número de células, o qual está diretamente relacionado à tensão produzida pela bateria. Usualmente, baterias utilizadas em quadricópteros possuem 3 células, as quais fornecem, juntas, 11.1 Volts (3.7 Volts/célula).

- Custo da bateria:

  O último passo, de acordo com a referência, é analisar o custo da bateria. Baterias com maiores capacidades, maiores taxa de descarga e maior número de células são mais caras e, geralmente, possuem maior massa. Portanto, o custo da bateria torna-se bastante relevante para sua seleção, uma vez que este pode elevar demasiadamente o custo total do protótipo.

Levando em consideração os itens mencionados anteriormente, foi selecionado a bateria 5200 mAh, 10 C, 3S (3 células), da marca Multistar. As características gerais da bateria e sua ilustração podem ser vistos na Tab. 5 e na Fig. 24, respectivamente.

Figura 24 – Ilustração da bateria escolhida

Tabela 5 – Características gerais da bateria escolhida

Modelo:	Multistar
Capacidade Mínima:	5200 mAh
Configuração:	3S1P/11V/3 Células
Taxa de Descarga:	10C
Pico de descarga:	20C (10s)
Massa:	331g
Dimensões:	142x49x22mm
Plug de carga:	JST-XH
Plug de descarga:	XT60

3.4 Estrutura do quadricóptero

A escolha da estrutura do quadricóptero foi feita levando-se em consideração dois principais aspectos: massa total da estrutura e custo. Dessa forma, foi priorizado a estrutura dji F450, a qual é razoavelmente leve, possui custo reduzido e é facilmente encontrada em sites de revenda de componentes para quadricópteros. A estrutura dji F450 pode ser vista na Fig. 25.

Figura 25 – Estrutura dij F450

Um outro aspecto importante na escolha da estrutura é a distância entre os eixos de rotação dos motores diametralmente opostos. Como cada hélice possui 10” (254 mm), a distância entre os centro de rotação deve ser maior que 254 mm, para evitar que as hélices colidam durante operação. Como pode ser visto na Fig. 26, as dimensões da dij F450 atendem ao requisito mencionado.

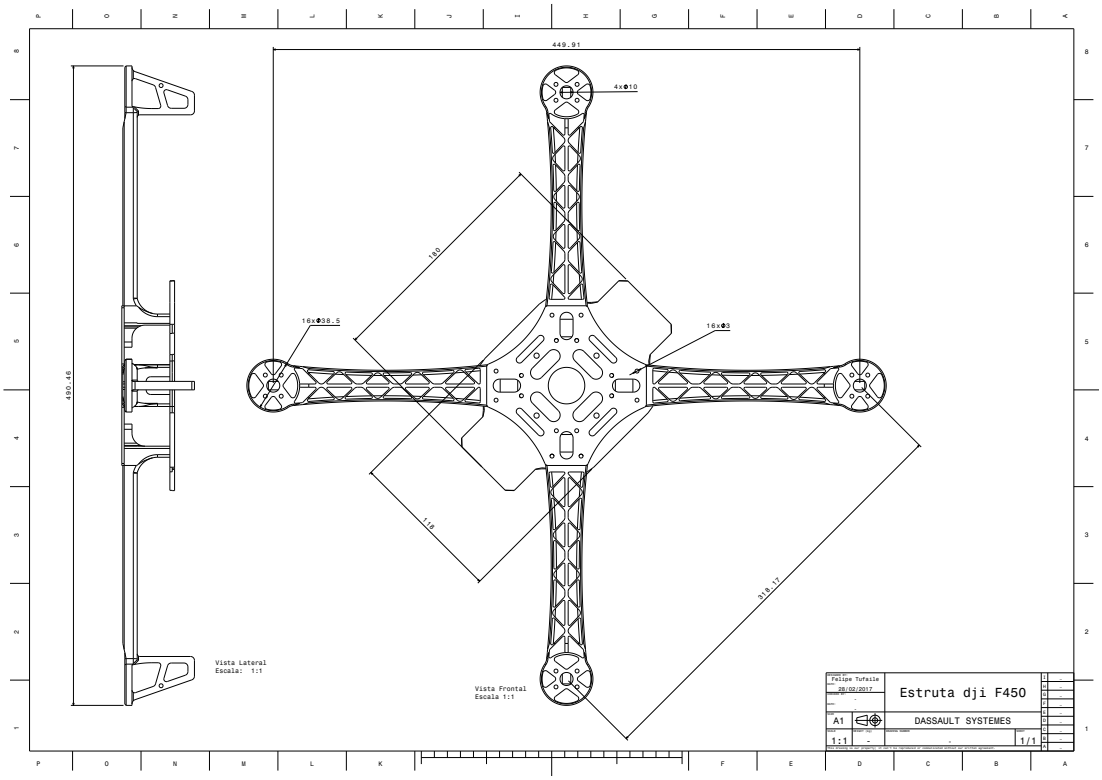
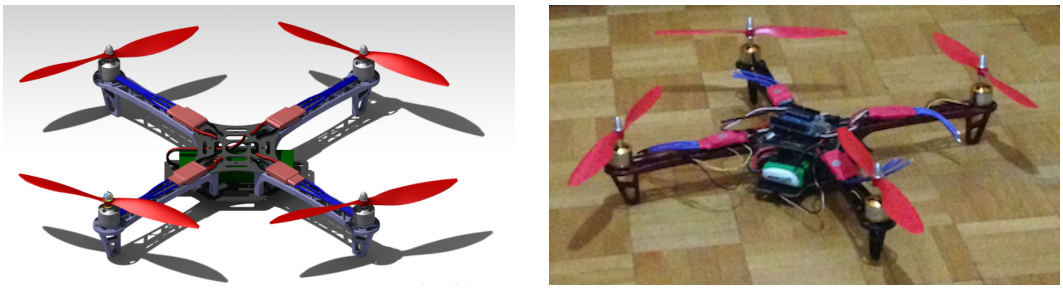


Figura 26 – Dimensões da estrutura dij F450

Com base nas dimensões do protótipo construído, foi possível construir um modelo virtual do quadricóptero, o qual permite calcular os valores dos momento. O modelo

virtual (construído com auxílio do *software* CATIA®) e o protótipo construído podem ser visto na Fig. 27.



(a) Modelo Virtual (b) Modelo Real

Figura 27 – Modelo virtual e protótipo construído

Os valores dos momentos de inércia e do centro de massa, calculados pelo "*software*" CATIA®, são apresentados na Tab. 6.

Vale destacar que o modelo virtual foi construído de forma que o referencial móvel seja coincidente com as direções principais de inércia. Logo, verifica-se que os produtos de inércia da estrutura são nulos, validando a hipótese levantada no capítulo anterior.

Tabela 6 – Momentos de inércia e posição do centro de massa do quadricóptero

Grandeza	Valor	Unidade
Massa	1,026	kg
Ixx	$11x10^{-}3$	kgm^2
Ixy	0	kgm^2
Ixz	0	kgm^2
Iyy	$11x10^{-}3$	kgm^2
Iyz	0	kgm^2
Izz	$20x10^{-}3$	kgm^2

3.5 Seleção dos sensores

Os sensores são dispositivos responsáveis por medir os valores reais das variáveis de estado e que serão utilizadas para atualizar o sinal de controle. Seu papel no sistema de controle é de fundamental importância, uma vez que sem esses dispositivos não seria possível ter conhecido do estado real de uma determinada planta. No caso específico do projeto de controle desenvolvido nesse trabalho, serão utilizados três sensores (um acelerômetro, um giroscópio e um magnetômetro), os quais serão assunto

dessa seção.

3.5.1 Acelerômetro

O acelerômetro é um dispositivo capaz de medir a aceleração de um corpo em relação à aceleração da gravidade, indicando o valor de 1 ”g” quando posicionados para cima e 0 ”g” quando inclinados 90 graus. Por essa razão, o acelerômetro possibilita que seja medido a inclinação de um corpo de acordo com a aceleração medida.

De acordo com (PFEIFER, 2013), tomando (r_S) como uma posição genérica do acelerômetro em relação ao centro de gravidade, o vetor de aceleração do sensor referido ao seu ponto de medição, a_S, pode ser expresso conforme equação a seguir:

$$a_S = a_{CG} + \omega \times r_S + \omega \times (\omega \times r_S) \tag{3.4}$$

Onde a_{CG} é a aceleração do corpo em relação ao centro de gravidade e ω o vetor da velocidade angular do corpo, conforme visto no capítulo dois. A aceleração de Coriollis não foi considerada na equação 3.4, pois, admitiu-se a hipótese de que o VANT não desempenhará grandes velocidades, conforme mencionado no capítulo dois.

Rearranjando a equação 3.4 e substituindo pelos termos apresentados no capítulo dois, chega-se em:

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \frac{1}{m} \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} + g \begin{bmatrix} -\sin\theta \\ \cos\theta \sin\phi \\ \cos\theta \cos\phi \end{bmatrix} + \begin{bmatrix} \dot{Q}r_{S_z} - \dot{R}r_{S_y} \\ \dot{R}r_{S_x} - \dot{P}r_{S_z} \\ \dot{P}r_{S_y} - \dot{Q}r_{S_x} \end{bmatrix} + \begin{bmatrix} Q(Pr_{S_y} - Qr_{S_x}) - R(Rr_{S_x} - Pr_{S_z}) \\ R(Qr_{S_z} - Rr_{S_y}) - P(Pr_{S_y} - Qr_{S_x}) \\ P(Rr_{S_x} - Pr_{S_z}) - Q(Qr_{S_z} - Rr_{S_y}) \end{bmatrix} \tag{3.5}$$

Linearizado a expressão acima utilizando a expansão em série de Taylor, obtém-se:

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = g \begin{bmatrix} -\theta \\ \phi \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{Q}r_{S_z} - \dot{R}r_{S_y} \\ \dot{R}r_{S_x} - \dot{P}r_{S_z} \\ \dot{P}r_{S_y} - \dot{Q}r_{S_x} \end{bmatrix} \tag{3.6}$$

Idealmente, os acelerômetros devem ser posicionados no centro de gravidade de um VANT, para evitar que estejam submetidos à acelerações centrípetas. Dessa forma, será admitido que o acelerômetro será posicionado o mais próximo possível do centro

de gravidade: coordenada $r_S = [0 \quad 0 \quad r_{S_z}]$. Logo, a equação 3.6 pode ser resumida conforme expressão a seguir:

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = g \begin{bmatrix} -\theta \\ \phi \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{Q} r_{S_z} \\ -\dot{P} r_{S_z} \\ 0 \end{bmatrix} \tag{3.7}$$

3.5.2 Giroscópio

Diferentemente do acelerômetro, o qual é utilizado para medir aceleração, o giroscópio é utilizado para medir velocidade angular. Com base no princípio físico da conservação do momento angular, o giroscópio é capaz de indicar a velocidade de rotação de um objeto por meio do cálculo do deslocamento de uma massa (dentro do giroscópio) a medida que o veículo inclina em relação aos eixos (Fig. 28) . O deslocamento medido é convertido em um baixo valor de corrente, o qual é amplificado e lido pelos microcontroladores.

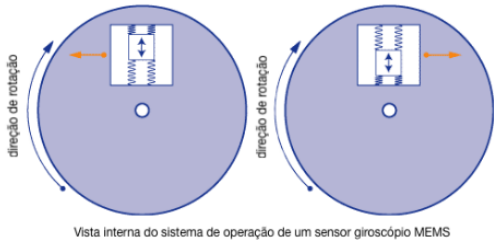


Figura 28 – Princípio de funcionamento de um giroscópio eletrônico do tipo MEMS (Sistema Micro-eletrônico), (RONZO, 2013).

Em termos gerais, o giroscópio é capaz de medir a taxa de variação angular ao redor dos eixos fixos ao quadricóptero, conforme as equações a seguir:

$$g_x = \frac{d\phi}{dt} = P \tag{3.8}$$

$$g_y = \frac{d\theta}{dt} = Q \tag{3.9}$$

$$g_z = \frac{d\psi}{dt} = R \tag{3.10}$$

3.5.3 Magnetômetro

Os magnetômetros são dispositivos capazes de medir a densidade de fluxo magnético criado por um determinado campo. Logo, sendo a densidade de fluxo magnético diretamente proporcional a intensidade do seu campo, os magnetômetros são capazes de detectar variações no campo magnético da terra (JAIN, 2012). Por essa razão, esses dispositivos funcionam como bússolas, indicando a inclinação de um objeto em relação aos polos magnéticos terrestre.

A determinação da orientação de um objeto por meio de um magnetômetro é realizada a partir do cálculo das expressões a seguir:

$$N_x = \cos\psi \tag{3.11}$$

$$N_y = -\sin\psi \tag{3.12}$$

$$N_z = 0 \tag{3.13}$$

Verifica-se, portanto, que a partir do uso do magnetômetro, é possível determinar a posição angular ψ. Após linearização das expressões acima, obtém-se:

$$\begin{bmatrix} N_x \\ N_y \\ N_z \end{bmatrix} = \begin{bmatrix} 1 \\ -\psi \\ 0 \end{bmatrix} \tag{3.14}$$

3.5.4 Sensores selecionados

Objetivando minimizar a massa do protótipo e reduzir o custo de construção, foi utilizado um único componente elétrico que une os três sensores mencionados: acelerômetro, giroscópio e magnetômetro. Tal componente foi encontrado no site da empresa *Sparkfun*, a qual, além de ser indicada por diversos autores, disponibiliza guias de instalação, pacotes de bibliotecas e *datasheets* para download.

O componente escolhido foi, portanto, o IMU (Unidade de Medição Inercial) "*SparkFun 9DoF Sensor Stick*", conforme pode ser visto na Fig. 29. Como o próprio nome do componente sugere, o sensor possui 9 graus de liberdade, uma vez que é possível medir as três componentes da aceleração, da rotação e da posição angular. Além disso, o componente possui em seu circuito o chip LSM9DS1 (sistema sensor de movimento), o qual é equipado com uma interface digital. Cada um dos LSM9DS1 é capaz de suportar uma ampla faixa de espectro, conforme apresentado na Tab. 7.

Figura 29 – IMU *SparkFun 9DoF Sensor Stick* da empresa SparkFun

Tabela 7 – Características do sensor IMU *SparkFun 9DoF Sensor Stick* (SPARKFUN, 2017)

Grandeza	Descrição
Chip	LSM9DS1
Fonte de Alimentação	3.3 V
Faixa Giroscópio	$\pm$245, $\pm$500, $\pm$2000 °/s
Faixa Aceleração	$\pm$2, $\pm$4, $\pm$8, $\pm$16g
Faixa Magnetômetro	$\pm$4, $\pm$8, $\pm$12, $\pm$16 gauss

A ficha de dados do circuito integrado escolhido, bem como o seu esquema de instalação é encontrado em anexo a este relatório.

4 CONTROLE MODERNO

Neste capítulo, será projetado um sistema de controle a partir da teoria de controle moderno. Para tanto, o modelo dinâmico obtido no capítulo dois será escrito em espaço de estados, afim de remodelar a dinâmica da planta por meio da alocação de polos. Também será estuado nesse capítulo a estabilidade do sistema de controle em malha fechada. Além disso, o capítulo será divido em duas partes, sendo a primeira referente ao projeto do controlador no domínio do tempo contínuo e a segunda ao projeto do controlador no domínio do tempo discreto, sendo este último será obtido a partir do projeto no tempo contínuo.

4.1 Projeto do controlador no tempo contínuo

4.1.1 Sistema em espaço de estados

Conforme apresentado no capítulo dois, o modelo matemático linear da dinâmica do quadricóptero é dado pela Eq. 2.42. A partir desse sistema, define-se o vetor de estado conforme a expressão a seguir:

$$\left[\eta\right] = \begin{bmatrix} X & Y & Z & U & V & W & \phi & \theta & \psi & P & Q & R \end{bmatrix}^t \tag{4.1}$$

Logo, o vetor de entrada é definido como:

$$\left[U\right] = \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \\ \Omega_4 \end{bmatrix} \tag{4.2}$$

Com isso, as matrizes [A] e [B] podem ser definidas utilizando os vetores de estado e de entrada, de forma a satisfazer a expressão a seguir:

$$\dot{\eta} = A\eta + BU \tag{4.3}$$

Dessa forma, a matriz [A] resulta em:

$$\left[A\right] = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.4}$$

Enquanto que a matriz [B], resulta em:

$$\left[B\right] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ K_{T\,lin}/m & K_{T\,lin}/m & K_{T\,lin}/m & K_{T\,lin}/m \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -K_{T\,lin}d_{cg}/I_{xx} & 0 & K_{T\,lin}d_{cg}/I_{xx} \\ K_{T\,lin}d_{cg}/I_{yy} & 0 & -K_{T\,lin}d_{cg}/I_{yy} & 0 \\ K_{T\,lin}K_{TM}/I_{zz} & -K_{T\,lin}K_{TM}/I_{zz} & K_{T\,lin}K_{TM}/I_{zz} & -K_{T\,lin}K_{TM}/I_{zz} \end{bmatrix} \tag{4.5}$$

As matrizes [C] e [D], por outro lado, são definidas a partir da dinâmica do sensor escolhido. Conforme apresentado, o sensor em questão mede as três componentes da aceleração, da rotação e da posição angular. Tendo-se conhecimento das grandezas medidas pelo sensor, o vetor de saída será formado de acordo com a equação a seguir:

$$\left[Y\right] = \begin{bmatrix} X & Y & Z & a_x & a_y & N_y & g_x & g_y & g_z \end{bmatrix}^t \tag{4.6}$$

Deve-se ressaltar que as variáveis de estado X, Y e Z não são diretamente medidas, sendo necessário integrar as componentes da aceleração linear: a_x, a_y e a_z. Dessa forma, verifica-se a necessidade de projetar um observador de estado, o qual têm por função estimar as variáveis que não podem ser diretamente medidas. Uma vez definido o vetor de saída, o modelo matemático linear da dinâmica dos sensores é apresentado no equacionamento a seguir:

$$\begin{bmatrix} X \\ Y \\ Z \\ a_x \\ a_y \\ N_y \\ g_x \\ g_y \\ g_z \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ -g\theta + \dot{Q}r_{sz} \\ g\phi - \dot{P}r_{sz} \\ -\psi \\ P \\ Q \\ R \end{bmatrix} \tag{4.7}$$

Logo, com base na expressão 4.7, as matrizes [C] e [D] são definidas conforme as expressões a seguir:

$$\left[C\right] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.8}$$

$$\left[D\right] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ K_{T\,lin} d_{cg} r_{sz}/I_{yy} & 0 & -K_{T\,lin} d_{cg} r_{sz}/I_{yy} & 0 \\ 0 & K_{T\,lin} d_{cg} r_{sz}/I_{xx} & 0 & -K_{T\,lin} d_{cg} r_{sz}/I_{xx} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (4.9)$$

4.1.2 Estudo da controlabilidade

A controlabilidade de um sistema de controle refere-se à capacidade de se conduzir o sistema a um determinado estado, dentro de um intervalo de tempo finito, o que não significa que após atingido o estado desejado, este será mantido (HENRIQUE, 2011). Matematicamente, para que um sistema seja completamente controlável, é necessário que o posto da matriz de controlabilidade, matriz "$\mathfrak{C}$", apresentada a seguir, seja igual ao número de variáveis de estado do sistema, no caso 12.

$$\mathfrak{C} = [A \quad AB \quad A^2B \quad ... \quad A^{n-1}B] \quad (4.10)$$

Analisando a expressão 4.10, verifica-se que a matriz de controlabilidade é construída utilizando-se a matriz [B], matriz referente ao sinal proveniente dos atuadores. Isso evidencia o fato de que o projeto de seleção dos atuadores deve resultar em uma matriz de controlabilidade com posto igual ao número de variáveis de estado. Caso o posto seja inferior ao número de variáveis de estado, deve-se adicionar atuador(es) ao sistema. Utilizando o "*software*" MATLAB®, verifica-se que, a matriz de controlabilidade possui, de fato, posto 12. Logo, o sistema é completamente controlável.

4.1.3 Estudo da observabilidade

A observabilidade, por sua vez, refere-se a possibilidade de determinar-se o estado de um sistema, em tempo finito, apenas a partir de suas saídas. Em outras palavras, as saídas do sistema devem ser suficientes para reconstruir seu estado. Matematicamente,

um sistema será considerado totalmente observável se o posto da matriz de observabilidade, "$\mathfrak{O}$", apresentada a seguir, for numericamente igual à quantidade de variáveis de estado. Logo, o posto da matriz "$\mathfrak{O}$" deverá ser 12.

$$\mathfrak{O} = [C \quad CA \quad CA^2 \quad ... \quad CA^{n-1}] \tag{4.11}$$

Resolvendo a equação 4.11 com auxílio do MATLAB®, verifica-se que o posto da matriz de observabilidade é de fato 12, sendo o sistema, portanto, totalmente observável.

Uma outra maneira de obter um sistema completamente observável, sem o uso de um observador de estado, seria utilizar sensores de posição (GPS) e sensores de altitude (barômetro ou sonar), o quais mediriam diretamente as posições X, Y e Z.

4.1.4 Controle por alocação de polos

Em um sistema de controle em malha aberta, conforme representado no digrama da Fig. 30, o sinal de saída não possui influência no sinal de entrada. Portanto, não há meios de estabilizar um sistema instável. Por essa razão e sabendo que a dinâmica do quadricóptero é bastante instável é necessário realizar o projeto de um controlador em malha fechada, o qual utiliza o sinal de saída para construir um novo sinal de entrada. Dessa forma, a dinâmica do sistema de controle será alterada de acordo com o diagrama de blocos ilustrado na Fig. 31.

Uma vez definida a dinâmica do sistema de controle em malha-fechada, pode-se alterar a dinâmica original do veículo por meio da alocação dos seus polos. Com base nessa técnica, a matriz de estado do sistema é reescrita por meio da introdução de uma matriz de ganho [K], fazendo que a dinâmica do sistema de controle seja reconfigurada conforme ilustrado no diagrama de blocos da Fig.32. A partir da figura, verifica-se que o sinal de entrada passa a ser definido de acordo com a equação $U = K(R - \eta)$, o que indica o fechamento da malha por meio de uma realimentação negativa.

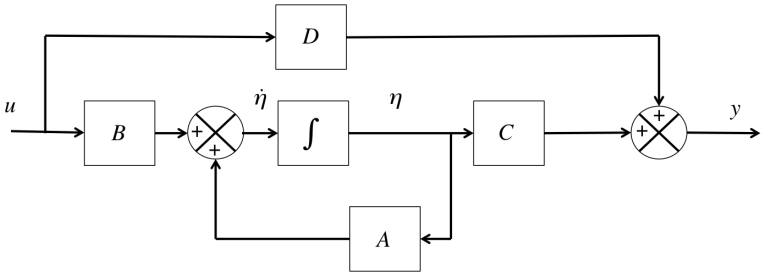


Figura 30 – Diagrama de blocos do sistema em malha aberta

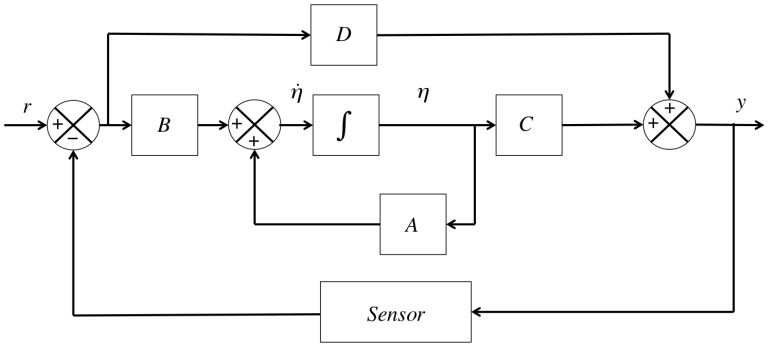


Figura 31 – Diagrama de blocos do sistema em malha fechada

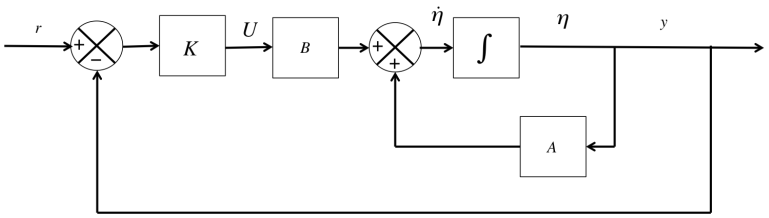


Figura 32 – Diagrama de blocos do sistema em malha fechada

Dessa forma, a nova equação do sistema possuirá a matriz de estados [A - BK], conforme demonstrado pelo equacionamento a seguir:

$$\dot{\eta} = A\eta + BU \tag{4.12}$$

$$\dot{\eta} = A\eta + BK(R - \eta) \tag{4.13}$$

$$\dot{\eta} = (A - BK)\eta + BKR \tag{4.14}$$

Após determinada a expressão do sistema de controle em malha fechada, apresentada pela equação 4.14, os novos polos do sistema foram definidos de forma que a dinâmica do veículo satisfizesse os requisitos de tempo de acomodação e percentual de sobressinal apresentadas na Tab. 8. Os polos obtidos por meio da aplicação dos requisitos mencionados são detalhados na mesma tabela.

Tabela 8 – Polos do sistema em malha aberta e malha fechada

Variavel de Estado	*Tempo de Acomodação*	*Percentual de Sobressinal*	*Polos (Malha Aberta)*	*Polos (Malha Fechada)*
X	1,5s	2%	0	-5,0000 + 4,0153i
Y	1,5s	2%	0	-5,0000 + 4,0153i
Z	1,5s	2%	0	-5,0000 + 4,0153i
U	1,5s	2%	0	-5,0000 - 4,0153i
V	1,5s	2%	0	-5,0000 - 4,0153i
W	1,5s	2%	0	-5,0000 - 4,0153i
ϕ	1,0s	1%	0	-5,7143 + 3,8982i
θ	1,0s	1%	0	-5,7143 + 3,8982i
ψ	1,0s	1%	0	-5,7143 + 3,8982i
P	1,0s	1%	0	-5,7143 - 3,8982i
Q	1,0s	1%	0	-5,7143 - 3,8982i
R	1,0s	1%	0	-5,7143 - 3,8982i

4.1.5 Observador de estados

Conforme mencionado anteriormente, as variáveis de estado referentes à posição linear do veiculo (X, Y e Z) não podem ser diretamente medidas pelo sensor. Dessa forma, torna-se necessário o uso de um observador de estados para estimar seus valores. Neste capítulo, será apresentado, portanto, o projeto do observador de estados do sistema de controle.

Adicionando a dinâmica do observador de estado à dinâmica do sistema de controle sem observador de estado, representado na Fig. 32, o sistema de controle possuíra a dinâmica representada pela Fig. 33. Analisando esta última figura, Observa-se, portanto, que a partir dos sinais de entrada e saída (vetores U e y, respectivamente) o observador de estado constrói o vetor de estados estimado $\hat{\eta}$. Uma vez representada a dinâmica do observador de estado, deve-se, então, calcular a matriz de ganho [L] para que o erro do observador convirja para zero assintoticamente. Matematicamente, definindo o erro do observador como $e = \hat{\eta} - \eta$, verifica-se que:

$$\dot{\eta} = A\eta + BU + L(y - \hat{y}) \tag{4.15}$$

$$\dot{e} = (A - LC)e \tag{4.16}$$

Logo, a matriz [L] deve ser tal que a dinâmica do observador seja mais rápida que a dinâmica do sistema. Aplicando o critério de que a dinâmica do observador de estado seja dez vezes mais rápida que a dinâmica do sistema, obtém-se os polos do observador apresentados na Tab. 9.

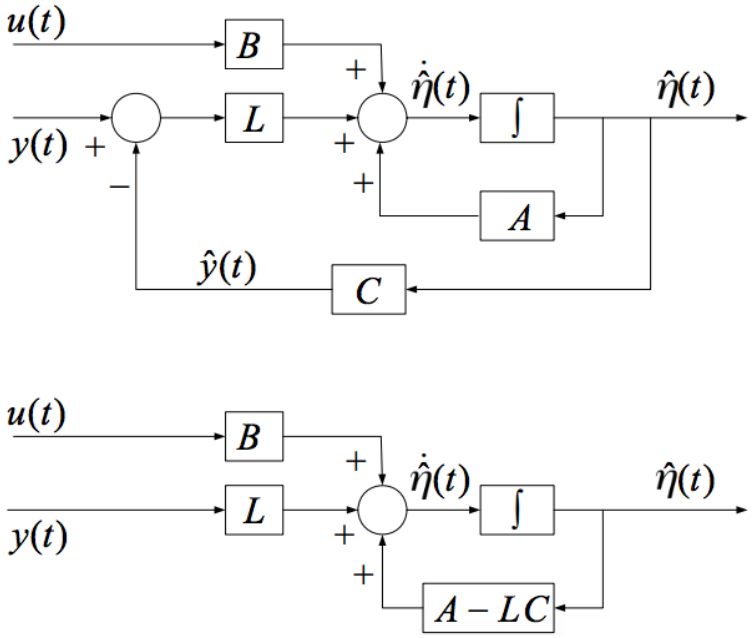


Figura 33 – Diagrama de blocos do sistema em malha fechada com observador de estados

Tabela 9 – Polos do observador de estados

Variavel de Estado	*Polos Observador*
X	-0,5000 + 4,0153i
Y	-0,5000 + 4,0153i
Z	-0,5000 + 4,0153i
U	-0,5000 - 4,0153i
V	-0,5000 - 4,0153i
W	-0,5000 - 4,0153i
ϕ	-0,5714 + 3,8982i
θ	-0,5714 + 3,8982i
ψ	-0,5714 + 3,8982i
P	-0,5714 - 3,8982i
Q	-0,5714 - 3,8982i
R	-0,5714 - 3,8982i

Uma vez definido os polos do observador de estado, a matriz [L] pode ser calculada utilizando o comando "*place*" no "*software*" MATLAB®.

4.1.6 Análise de Estabilidade

Haja visto que a dinâmica do quadricóptero é instável, um dos objetivos do sistema de controle é estabilizá-la. Portanto, a analise da estabilidade do sistema de controle projetado é fundamental. Por essa razão, será apresentado, nesse seção, a análise de estabilidade por meio do estudo da posição dos polos no plano imaginário.

Conhecendo a equação temporal que descreve a dinâmica do quadricóptero, chamando genericamente de f(t), uma maneira de verificar matematicamente sua estabilidade é calcular o limite de f(t) para um intervalo de tempo infinito:

$$y = \lim_{t \to \infty} f(t) \tag{4.17}$$

Sendo assim, supondo um sistema dinâmico de segunda ordem genérico, que possua polos complexos "$a+bi$" e "$a-bi$" , sua função de transferência seria descrita da seguinte forma:

$$F(s) = \frac{m}{(s-a)^2 + b^2} \tag{4.18}$$

Em seguida, aplicando a transformada inversa de Laplace, a Eq.4.18 resulta em:

$$f(t) = e^{at} \sin(bt) \tag{4.19}$$

Substituindo a expressão 4.19 na expressão 4.17 e considerando que o termo $\sin(bt)$ é limitado entre os valores -1 e 1, verifica-se que para que a resposta "y" não seja infinita, a parcela real "a" deve ser negativa. Em outras palavras, para que um sistema seja estável no domínio de tempo contínuo, todos os polos devem estar localizados no semi-plano negativo do plano complexo.

Analisando o posicionamento dos polos do sistema em malha fechada no plano imaginário (Fig. 34), verifica-se que todos os polos encontram-se no semi-plano negativo.

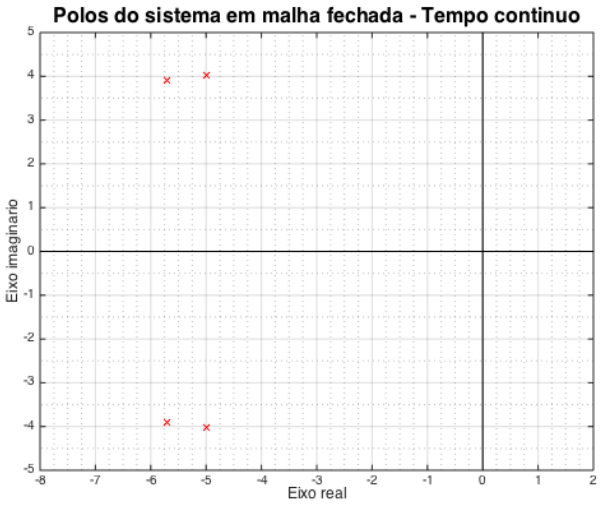


Figura 34 – Polos do sistema em malha fechada (tempo contínuo)

4.2 Projeto do controlador no tempo discreto

Sabendo-se que um dos objetivos desse trabalho é implementar o sistema de controle projetado em um protótipo, é imprescindível definir o sistema de controle no tempo discreto, domínio no qual é realizada a comunicação digital. Por essa razão, neste capítulo será apresentado a tradução do sistema em espaço de estado do tempo contínuo para o tempo discreto.

4.2.1 Fundamentação teórica

Em um sistema de controle escrito no domínio do tempo discreto, as variáveis de estado podem mudar apenas em instantes de tempo específicos. Portanto, definindo-se um intervalo de tempo, t_d, suficientemente pequeno, as variáveis de estado só poderão mudar nos instantes múltiplos inteiros de t_d. Escolhendo corretamente o período de discretização t_d, pode-se assumir que o intervalo de tempo entre dois instantes é suficientemente pequeno para que a informação, durante este intervalo, possa ser obtida por meio de interpolação.

Por essa razão, enquanto que no domínio do tempo contínuo a dinâmica do sistema é escrita por meio de equações diferenciais, no domínio de tempo discreto, a dinâmica do sistema é descrita por equações de diferença. Além disso, a transferência de informações em sistemas definidos no tempo discreto ocorre, na maioria das vezes, na forma digital, sendo necessário conversão da informação analógica em digital e vice versa (OGATA, 2016).

Ao aplicar a discretização do tempo no modelo definido em espaço de estados, verifica-se a necessidade de definir as matrizes [G], de forma que o sistema no tempo contínuo e no tempo discreto sejam equivalentes:

$$\begin{aligned} \dot{\eta}(t) &= A\eta(t) + Bu(t) \\ y(t) &= C\eta(t) + Du(t) \end{aligned} \tag{4.20}$$

$$\begin{aligned} \eta((k+1)T) &= G(T)\eta(kT) + H(T)u(kT) \\ y(kT) &= C\eta(kT) + Du(kT) \end{aligned} \tag{4.21}$$

De acordo com (OGATA, 2016), utilizando a solução da expressão 4.20 para calcular os valores do estado η nos instantes kT e $(k+1)t_d$, chega-se em:

$$\eta((k+1)T) = e^{A(k+1)T}\eta(0) + e^{A(k+1)T}\int_0^{(k+1)T} e^{-A\tau}Bu(\tau)d\tau \tag{4.22}$$

$$\eta(kT) = e^{AkT}\eta(0) + e^{AkT}\int_0^{kT} e^{-A\tau}Bu(\tau)d\tau \tag{4.23}$$

Em seguida, multiplicando a expressão 4.23 pelo termo e^{AT} e substituindo o valor de $\eta(kT)$ na expressão 4.22, obtém-se:

$$\eta((k+1)T) = e^{AT}\eta(kT) + \int_0^{T} e^{-A\lambda}d\lambda Bu(kT) \quad \lambda \in [0, kT) \tag{4.24}$$

Logo, verifica-se que as matrizes [G] e [H] resultam em:

$$G(T) = e^{AT} \tag{4.25}$$

$$H(T) = \int_0^{T} e^{-A\lambda}d\lambda B \tag{4.26}$$

onde $\lambda = (k+1)T - \tau$ e $d\lambda = -d\tau$.

4.2.2 Intervalo de discretização

Conforme mencionado no tópico anterior, um passo importante na discretização de um sistema é a determinação do período de discretização, t_d. De acordo com o teorema de Nyquist-Shannon, este intervalo de tempo deve ser pequeno suficiente para que nenhuma informação seja perdida durante o processo de discretização, permitindo que um determinado sinal seja completamente reconstruído.

Ainda de acordo com o teorema, um sinal contínuo, cujo espectro é nulo fora do intervalo [-ω_0, ω_0] é descrito de forma única por seus valores, em pontos equidistantes, se a frequência de amostragem do sinal, ω_s, for maior que o dobro da frequência do sistema, ω_0 (TU BERLIN, 2014). Entretanto, geralmente utiliza-se uma frequência de amostragem 5 ou 10 vezes maior que a frequência do sistema.

Com base na consideração feita no parágrafo anterior, a referência (PFEIFER, 2013) utiliza um intervalo de discretização $t_d = 0.02s$, o qual também é utilizado neste trabalho.

4.2.3 Sistema em espaço de estados

Uma vez realizado as considerações necessárias sobre o sistema no domínio de tempo discreto, o próximo passo é determinar as matrizes [G(T)] e [H(T)], utilizando as equações 4.25 e 4.26. Entretanto, o procedimento será realizado com auxílio do comando "*c2d*"no ambiente de programação MATLAB®. Aplicando o comando mencionado, as matrizes [G] e [H] obtidas são:

$$\Big[G\Big] = \begin{bmatrix} 1.00 & 0 & 0 & 0.01 & 0 & 0 \\ 0 & 1.00 & 0 & 0 & 0.01 & 0 \\ 0 & 0 & 1.00 & 0 & 0 & 0.01 \\ 0 & 0 & 0 & 1.00 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.00 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.00 \end{bmatrix} \tag{4.27}$$

$$\Big[H\Big] = \begin{bmatrix} 0 & -8.386x10^{-6} & 0 & 8.386x10{-}6 \\ 1.4625x10^{-5} & 0 & -8.898x10^{-6} & 0 \\ 9.295x10^{-7} & -5.33x10^{-7} & 5.655x10^{-7} & -5.33x10{-}7 \\ 0 & -0.1677x10^{-3} & 0 & 1.677x10{-}3 \\ 2.925x10^{-3} & 0 & -1.780x10^{-3} & 0 \\ 1.859x10^{-4} & -1.07x10^{-4} & 1.131x10^{-4} & -1.07x10^{-4} \end{bmatrix} \tag{4.28}$$

Vale ressaltar que as matrizes [C] e [D] não se alteram:

$$\left[C\right] = \begin{bmatrix} 0 & -9.81 & 0 & 0 & 0 & 0 \\ 9.81 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.29}$$

$$\left[D\right] = \begin{bmatrix} -0.0023 & 0 & 0.0014 & 0 \\ 0 & -0.0013 & 0 & 0.0013 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.30}$$

4.2.4 Sistema em malha fechada

Conforme mencionado, em um sistema de controle digital o sinal de entrada analógico deve ser convertido em um sinal digital por meio de um amostrador e um conversor analógico/digital. Dessa forma, o microcontrolador é capaz de processar uma sequência numérica por meio de algoritmos e rotinas de cálculo, gerando uma nova sequência numérica (OGATA, 2016). Após calculada uma nova sequência numérica, é necessário transformar esta informação digital de volta para analógica, permitindo que o atuador a interprete. Este segundo procedimento, por sua vez, é realizado por um conversor digital/analógico e por um segurador. A sequência de procedimentos descritos são ilustrados no diagrama de blocos da Fig. 35, a qual representa as etapas de um sistema de controle digital em malha fechada.

Redesenhando a dinâmica de controle representada na Fig. 32, utilizando as matrizes [G] e [H] definidas no domínio do tempo discreto, verifica-se que a dinâmica de controle é similar, conforme apresentado na Fig. 36.

Resta, portanto, determinar a matriz de ganho K_d, a qual pode ser calculada a partir dos novos polos do sistema em malha fechada, no domínio do tempo discreto. Por sua vez, os polos dos sistema em malha fechada no domínio do tempo discreto podem ser

determinados a partir dos polos do sistema no domínio do tempo contínuo, utilizando-se a expressão a seguir:

$$s_{discreto} = e^{s_{continuo}T} \tag{4.31}$$

Com isso, os novos polos do sistema no domínio do tempo discreto resultaram nos polos apresentados na Tab. 10.

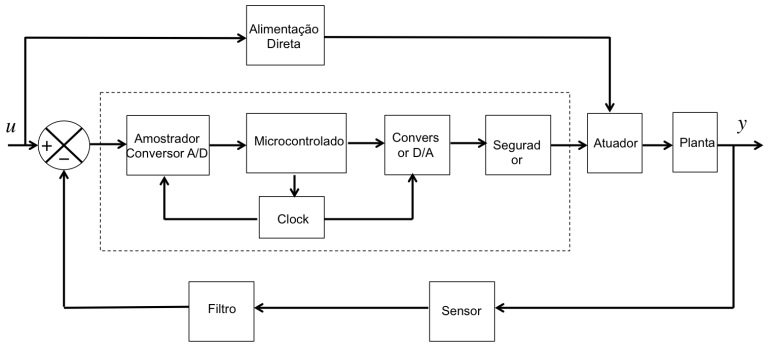


Figura 35 – Diagrama de blocos do sistema em malha fechada (Controlador Digital)

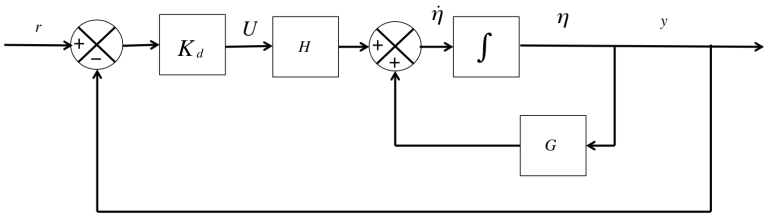


Figura 36 – Diagrama de blocos do sistema em malha fechada

Tabela 10 – Polos do sistema em malha fechada - Tempo Discreto

Variavel de Estado	*Polos (Malha Fechada)*
X	-0,9019 + 0,0726i
Y	-0,9019 + 0,0726i
Z	-0,9019 + 0,0726i
U	-0,9019 - 0,0726i
V	-0,9019 - 0,0726i
W	-0,9019 - 0,0726i
ϕ	-0,8893 + 0,0695i
θ	-0,8893 + 0,0695i
ψ	-0,8893 + 0,0695i
P	-0,8893 - 0,0695i
Q	-0,8893 - 0,0695i
R	-0,8893 - 0,0695i

4.2.5 Análise de Estabilidade

Conforme visto, no domínio do tempo discreto, os polos podem ser determinados por meio da transformação apresentada na equação 4.31. Utilizando esta transformação em um polo complexo genérico, no tempo contínuo, "a + bi", o polo correspondente, no tempo discreto, será:

$$z = e^{T(a+bi)} = e^{Ta}e^{Tbi} = e^{Ta}e^{(Tbi+2\pi k)} \tag{4.32}$$

Dessa forma, pode-se concluir que os polos no domínio da frequência que diferem de inteiros múltiplos da frequência de amostragem $2\pi/T$ são mapeados na mesma posição no plano "z", e que valores negativos de "a" (lado esquerdo do plano complexo) resultam $|z| = e^{Ta} < 1$, (OGATA, 2016). Com isso, concluí-se que para que um sistema seja estável no domínio de tempo discreto, os polos devem ser localizados no interior de uma circunferência de raio unitário.

Tendo em vista a conclusão obtida no parágrafo anterior e plotando os polos no plano imaginário, conforme ilustrado na Fig. 37, verifica-se que o sistema projetado no domínio do tempo discreto é estável, uma vez que todos os polos encontram-se no interior de uma circunferência de raio unitário.

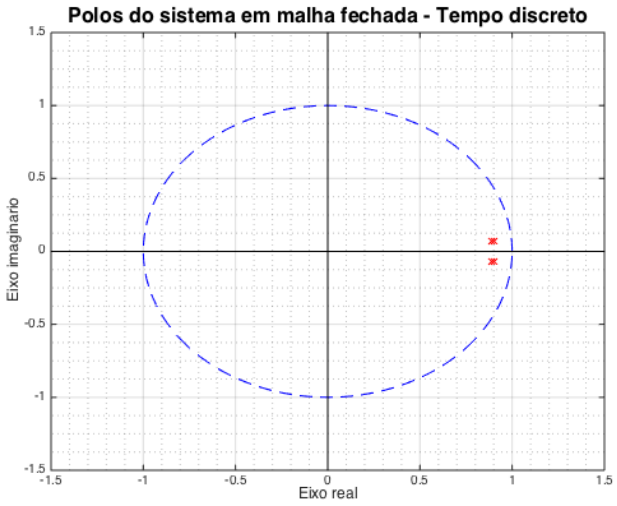


Figura 37 – Polos do sistema em malha fechada (tempo discreto)

4.3 Simulação do controle discreto

Conforme mencionado, o sistema de controle desenvolvido nas seções anteriores têm por objetivo regular as variáveis de estado relacionadas aos ângulos de Euler (posição angular e velocidade angular) além de permitir que uma determinada trajetória seja seguida, sendo está última definida por meio das variáveis de estado X, Y e Z. Tendo em vista a função do controlador, uma trajetória genéria foi criada utilizando o "*software*" MATLAB®, com o objetivo de analisar, por meio de simulação numérica do sistema de controle discreto, o desempenho do controlador projetado. Os resultados da simulação podem ser vistos nas Fig. 38, 39 e 40.

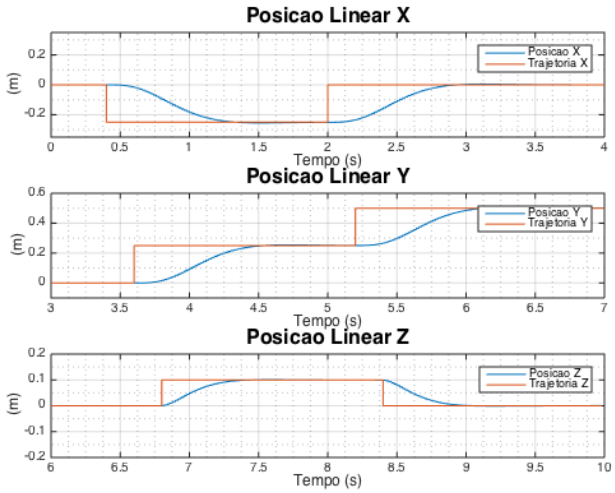


Figura 38 – Resposta do sistema em malha fechada: posições lineares

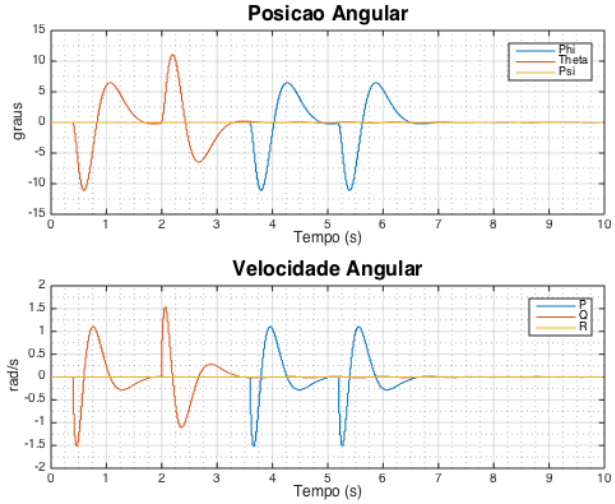


Figura 39 – Resposta do sistema em malha fechada: posições e velocidades angulares

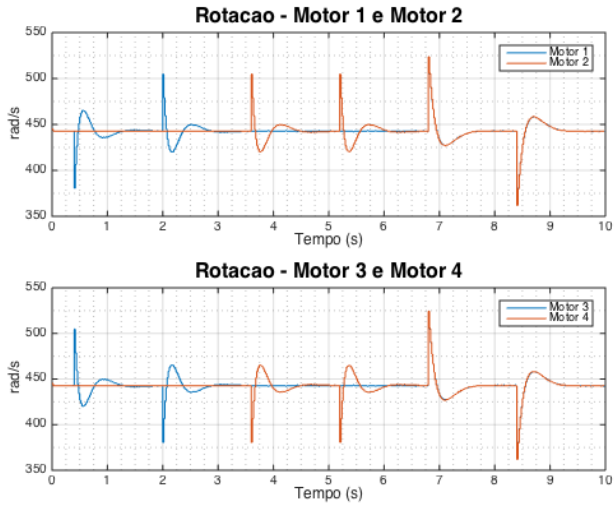


Figura 40 – Sinais de entrada no controle discreto

Como pode ser observado na Fig. 38, o sistema de controle é capaz de fazer com que o veículo (linha azul) siga a trajetória (linha vermelha). Observa-se que a condição de regime é alcançada após, aproximadamente, 1,0 segundo, para cada variação de posição definida pela trajetória, o que satisfaz a restrição imposta na seção 4.4.

A partir da Fig.39, verifica-se que o sistema de controle de fato regula as posições e velocidades angulares do veículo, fazendo com que as variáveis de estado ϕ, θ, ψ, P, Q e R tendam a zero na condição de regime. Além disso, é possível observar que a condição de regime é alcançada após, aproximadamente, 1,5 segundos, satisfazendo o critério imposto na seção 4.4.

Por fim, analisando a Fig.40, verifica-se que a velocidade máxima e mínima desempenhada por qualquer um dos atuadores está dentro da faixa operacional definida durante a linearização da dinâmica dos atuadores. Dessa forma, concluí-se que os atuadores estarão operando dentro das condições utilizadas como hipótese para modelagem matemática de sua dinâmica.

5 Controle clássico

O controle clássico é uma outra teoria, amplamente utilizada, para projetar um sistema de controle. Por meio dessa teoria, desenvolve-se um controlador PID (controlador proporcional, integrativo e derivativo) o qual é projetado a partir de um sistema com apenas uma entrada e uma saída, sistemas SISO (Uma entrada e uma saída).

Entretanto, dado que a dinâmica de voo do quadricóptero corresponde a um sistema com várias entradas e saídas, sistema MIMO (*Múltiplas entradas e múltiplas saídas*, será necessário analisar uma função de transferência para cada atuador e saída. Sendo assim, dado a existência de 4 atuadores e 9 saídas (leituras do sensor), é de se esperar que o modelo de controle clássico irá resultar em 36 funções de transferência. Entretanto, conforme será visto, cada atuador controla apenas algumas variáveis de estado do problema, não possuindo efeito algum em outras variáveis. Por essa razão, apenas faz sentido estudar 20 funções de transferência das 36 totais.

Além disso, é importante mencionar que dado a limitação da teoria de controle clássica, o sistema de controle deve ser desenvolvido separadamente para cada movimento do quadricóptero, não sendo possível desenvolver um sistema de controle em que um mesmo atuador controle mais de uma variável.

5.1 Determinação das funções de transferência

Conforme mencionado, o sistema de controle deve ser projetado para controlar os movimentos do quadricóptero de forma separada. Sendo assim e tendo em vista que o principal objetivo do projeto de controle é estabilizar o veículo, no projeto do controlador por meio da teoria clássica, será realizado o controle dos ângulos de Euler, ϕ e θ.

- Atuador 1

  Com relação ao atuador 1, conforme nomenclatura detalhada na Fig. 15, as variáveis que podem ser controladas são: deslocamento vertical ”z”, aceleração na direção ”x”, posição angular ”ψ”, velocidade angular ”Q”e velocidade angular ”R”. As respectivas funções de transferência são ilustradas a seguir:

$$FT_{Z1} = \frac{0.009583}{s^2} \tag{5.1}$$

$$FT_{ax1} = \frac{-(0.001843s^2 + 2.308)}{s^2} \tag{5.2}$$

$$FT_{\psi 1} = \frac{-0.01495}{s^2} \tag{5.3}$$

$$FT_{Q1} = \frac{0.2352}{s} \tag{5.4}$$

$$FT_{R1} = \frac{0.01495}{s} \tag{5.5}$$

- Atuador 2

  O atuador 2, por sua vez, é capaz de controlar o deslocamento vertical ”z”, aceleração na direção ”y”, posição angular ”ψ”, velocidade angular ”P”e velocidade angular ”R”. As respectivas funções de transferência são ilustradas a seguir:

$$FT_{Z2} = \frac{0.009583}{s^2} \tag{5.6}$$

$$FT_{ay2} = \frac{-(0.001843s^2 + 2.308)}{s^2} \tag{5.7}$$

$$FT_{\psi 2} = \frac{-0.01495}{s^2} \tag{5.8}$$

$$FT_{P2} = \frac{-0.2352}{s} \tag{5.9}$$

$$FT_{R2} = \frac{-0.01495}{s} \tag{5.10}$$

- Atuador 3

  Uma vez que o atuador 3 encontra-se diametralmente oposto ao atuador 1, as variáveis por ele controlada são as mesmas variáveis controladas pelo atuador 1. Entretanto, algumas funções de transferência possuem sinal oposto.

$$FT_{Z3} = \frac{0.009583}{s^2} \tag{5.11}$$

$$FT_{ax3} = \frac{-(0.001843s^2 + 2.308)}{s^2} \tag{5.12}$$

$$FT_{\psi 3} = \frac{-0.01495}{s^2} \tag{5.13}$$

$$FT_{Q3} = \frac{0.2352}{s} \tag{5.14}$$

$$FT_{R3} = \frac{0.01495}{s} \tag{5.15}$$

- Atuador 4

  De forma similar, estando o atuador 4 posicionado diametralmente oposto ao atuador 2, as variáveis por ele controlada são as mesmas variáveis controladas pelo atuador 2, valendo-se a mesma ressalva feita para o atuador 3 com relação ao sinais de algumas das funções de transferência.

$$FT_{Z4} = \frac{0.009583}{s^2} \tag{5.16}$$

$$FT_{ay4} = \frac{-(0.001843s^2 + 2.308)}{s^2} \tag{5.17}$$

$$FT_{\psi 4} = \frac{-0.01495}{s^2} \tag{5.18}$$

$$FT_{P4} = \frac{-0.2352}{s} \tag{5.19}$$

$$FT_{R4} = \frac{-0.01495}{s} \tag{5.20}$$

5.1.1 Controle dos movimentos de arfagem e rolagem

Conforme mencionado, as variáveis de estado que serão controladas por meio do controle PID serão as variáveis referentes ao ângulo de arfagem e ao ângulo de rolagem. Sendo assim, o sistema de controle projetado com base na teoria clássica consistirá em um regulador de estado.

Uma observação importante a respeito das funções de transferências enunciadas no tópico anterior é que tanto a variável a_x quanto a variável a_y não são variáveis de estado definidas no modelo dinâmico do quadricóptero, conforme ilustrado no capítulo 2. Tais variáveis são necessárias, conforme ilustrado na Eq. 3.7, para determinar as variáveis de estado ϕ e θ. Sendo assim, torna-se necessário determinar uma nova função de transferência, a partir das equações 5.2, 5.4, 5.7, 5.9, 5.12, 5.14, 5.17, 5.19 e 3.7. Com isso, as novas funções de transferência serão:

- Atuador 1

$$FT_{\theta 1} = \frac{0.2352}{s^2} \tag{5.21}$$

- Atuador 2

$$FT_{\phi 2} = \frac{-0.2352}{s^2} \tag{5.22}$$

- Atuador 3

$$FT_{\theta 3} = \frac{-0.2352}{s^2} \tag{5.23}$$

- Atuador 4

$$FT_{\phi 4} = \frac{0.2352}{s^2} \tag{5.24}$$

Observa-se que, apesar da troca do sinal, a função de transferência é a mesma para todos os atuadores. Isso ocorre principalmente por dois motivos: primeiro porque durante a modelagem dos motores, considerou-se que todos possuem a mesma dinâmica e segundo porque, dado a simetria do veículo, a dinâmica dos ângulos ϕ e θ são as mesmas.

Por fim, verifica-se que para ser possível controlar os ângulos ϕ e θ, por meio da teoria clássica, é necessário os quatro atuadores do veículos sejam reservados apenas para esta finalidade. Para que seja possível controlar as demais variáveis, utilizando-se apenas 4 atuadores, deve-se utilizar um outro sistema de controle, por exemplo, um sistema desenvolvido por meio da teoria de controle moderno, conforme o desenvolvido no capítulo 4.

5.1.2 Sintonização do PID

Um controlador PID é formado pela junção de três tipos de controladores: um controlador proporcional, um controlador integrativo e um controlador derivativo. Portanto, durante a sintonização do controlador PID, é necessário determinar um ganho proporcional, integrativo e derivativo.

Nos próximos capítulos, será, portanto, apresentadas diferentes técnicas para determinar os ganhos mencionados.

5.2 Sintonização do PID via MATLAB®("pidTuner")

Uma maneira prática de determinar os ganhos proporcional, integrativo e derivativo é utilizando o comando "*pidTuner()*"do MATLAB®. Ao utilizar a ferramenta, colocando-se a função de transferência no argumento do comando, uma janela irá inicializar onde será exibido a resposta do sistema controlado. O usuário pode então ajustar os ganhos de forma que a resposta produzida atenda os critérios de projeto.

No caso, utilizou-se como critério de projeto sobressinal máximo de 12% e tempo de estabilização de 1,2 segundos. A Fig. 41 ilustra a sintonização do controlador PID para a função de transferência Eq.5.21, utilizando o comando mencionado. Deve-se mencionar que como as demais funções de transferência são iguais ou possuem apenas o sinal oposto, os ganhos possuirão os mesmos valores absolutos.

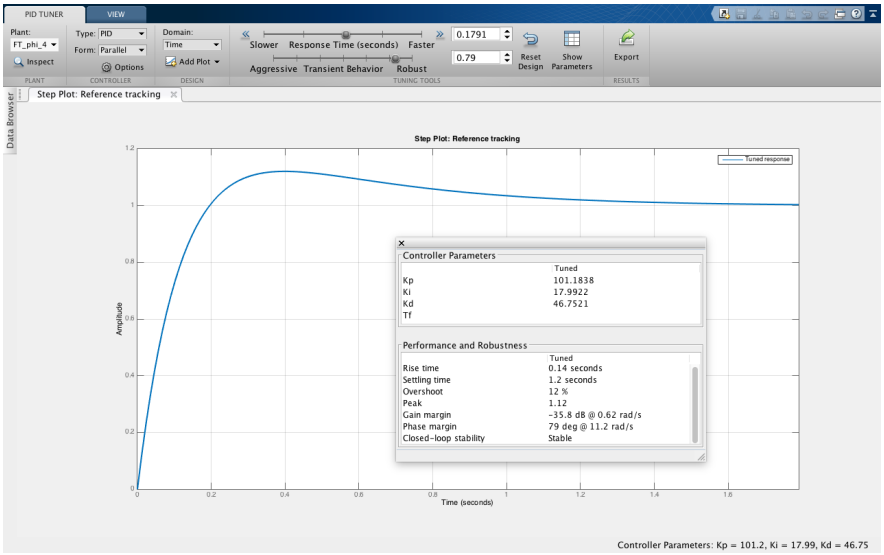


Figura 41 – Sintonização do PID via comando "*pidTuner*"

Como as funções de transferência são similares para todos os atuadores, conforme mencionado, os ganhos serão os mesmos para cada um dos controladores, com exceção dos sinais. Sendo assim, os ganhos calculados podem ser vistos na Tab. 11.

Tabela 11 – Ganhos do PID - "*pidTuner*"

FT	k_P	k_I	k_D
$FT_{\theta 1}$	101,18	17,99	46,75
$FT_{\phi 2}$	-101,18	-17,99	-46,75
$FT_{\theta 3}$	-101,18	-17,99	-46,75
$FT_{\phi 4}$	101,18	17,99	46,75

5.2.1 Sintonização do PID via método de Ziegler-Nichols

Um outro método bastante utilizado para sintonização de um controlador PID é o método desenvolvido por J. Ziegler e N. Nichols, o qual define os ganhos proporcional, integrativo e derivativo a partir de dois parâmetros, ganho crítico e período crítico, determinados a partir da resposta oscilatória pura do sistema. Dessa forma, o primeiro passo do método é determinar o ganho crítico, K_{CR}, que faz com que o sistema produza uma resposta puramente oscilatória. Aplicando o comando "*root-locus*", lugar das raízes, no MATLAB®, o qual devolve as trajetórias dos polos do sistema em malha fechada como função de um ganho k, assumindo realimentação negativa, verifica-se que o ganho crítico corresponde ao ganho para o qual os polos do sistema encontram-se posicionados no eixo imaginário.

Analisando o gráfico da Fig. 42, a qual representa a resposta do sistema em malha fechada para $K_{CR} = 270$, verifica-se que o comportamento do sistema é puramente oscilatório. O parâmetro P_{CR}, por sua vez, pode ser determinado calculando-se o intervalo de tempo entre dois picos consecutivos, conforme ilustrado na figura.

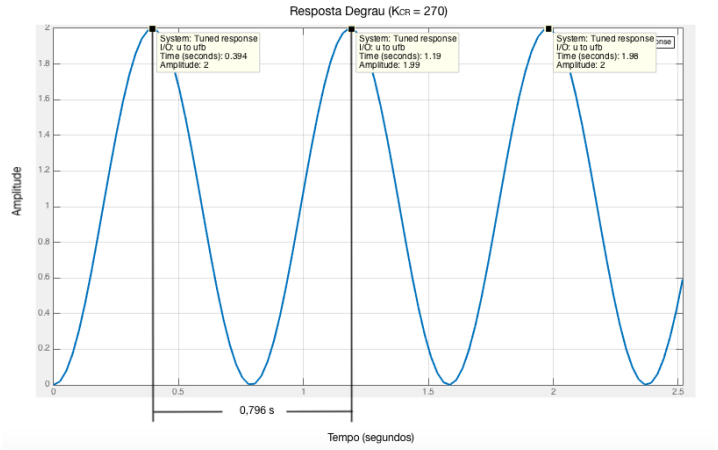


Figura 42 – Resposta degrau da função de transferência, em malha fechada, para $K_{CR} = 270$.

Uma vez determinados os parâmetros P_{CR} e K_{CR}, os ganhos k_P, k_I e k_D podem ser determinados de acordo com a Tab. 12.

Tabela 12 – Método Ziegler-Nichols

Tipo de controle	k_P	k_I	k_D
P	$0{,}50K_{CR}$	-	-
PI	$0{,}45K_{CR}$	$1{,}2k_P/P_{CR}$	-
PID	$0{,}60K_{CR}$	$2k_P/P_{CR}$	$k_P/(8P_{CR})$

Dessa forma, os ganhos dos controladores PIDs sintonizados pode ser visto na Tab. 13.

Tabela 13 – Ganhos do PID - Ziegler-Nichols

FT	k_P	k_I	k_D
$FT_{\theta 1}$	162,00	410,13	16,00
$FT_{\phi 2}$	-162,00	-410,13	-16,00
$FT_{\theta 3}$	-162,00	-410,13	-16,00
$FT_{\phi 4}$	162,00	410,13	16,00

Por fim, simulando a função de transferência $FT_{\theta 1}$, em malha fechada, com controlador PID, para uma entrada degrau unitário, conforme ilustrado na Fig. 43, verifica-se que o tempo decorrido até que o sistema atinja a condição de regime, bem como o sobressinal, são distantes das restrições impostas de 1,2 segundos e 12%, respectivamente.

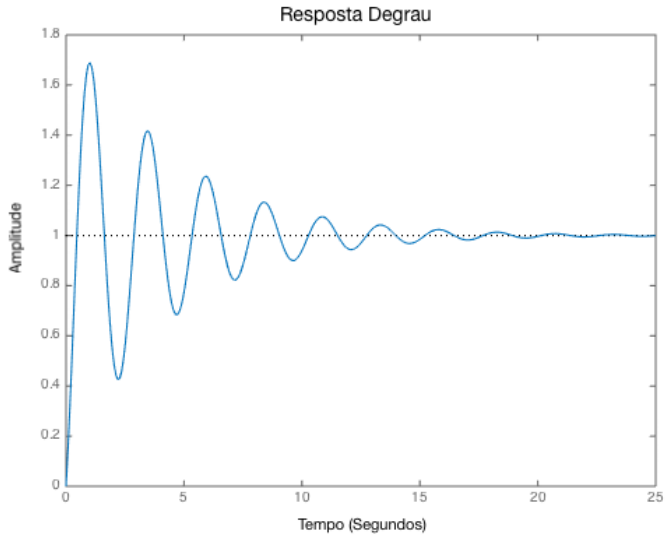


Figura 43 – Resposta degrau unitário da função de transferência $FT_{\theta 1}$, em malha fechada, com controlador PID sintonizado pelo método de Ziegler-Nichols.

5.2.2 Sintonização do PID via índice de desempenho ITAE

O método ITAE (*Integration Time Absolute Error*) é utilizado para configurar um controlador de forma que o tempo de estabilização da resposta seja o mais rápido possível. Isso porque, através do método, atribui-se maior peso para o erro que permanece por um longo período de tempo (SHUAIB *et. al.*, 2007). Matematicamente, o índice de desempenho ITAE é calculado por meio da integral da multiplicação do erro absoluto pelo tempo, conforme equação a seguir:

$$ITAE = \int_0^T t|e(t)|dt \tag{5.25}$$

Para que seja possível calcular o índice de desempenho, a função de transferência do sistema deve estar escrito na forma mais geral:

$$T(s) = \frac{Y(s)}{R(s)} = \frac{b_0}{s^n + b_{n-1}s^{n-1} + ... + b_1 s + b_0} \tag{5.26}$$

Sendo a função de transferência do controlador PID descrita conforme a expressão a seguir:

$$PID(s) = \frac{k_d s^2 + k_p s + k_i}{s} \tag{5.27}$$

Portanto, a função de transferência, em malha fechada, da Eq. 5.21, com controlador será:

$$H(s) = \frac{0,2352k_d s^2 + 0,2352k_p s + 0,2352k_i}{s^3 + 0,2352k_d s^2 + 0,2352k_p s + 0,2352k_i} \tag{5.28}$$

Utilizando os coeficientes ótimos ITAE de terceira ordem, verifica-se que:

$$0,2352k_d = 1,783W_n \tag{5.29}$$

$$0,2352k_p = 2,172W_n^2 \tag{5.30}$$

$$0,2352k_d = W_n^3 \tag{5.31}$$

Além disso, sabendo-se que para um sistema de segunda ordem, o tempo de estabilização pode ser calculado por meio da expressão abaixo e que o amortecimento, ζ, é próximo de 0,7, tem-se que o valor da frequência W_n é igual a $4,76\ rad/s$.

$$T = \frac{4}{\zeta W_n} \tag{5.32}$$

Substituindo o valor de W_n nas expressões Eq. 5.29 - Eq. 5.31 e aplicando-se os mesmos procedimentos para as demais funções de transferência, chega-se aos ganhos descritos na Tab. 14.

Tabela 14 – Ganhos do PID - ITAE

FT	k_P	k_I	k_D
$FT_{\theta 1}$	160,32	307,56	31,59
$FT_{\phi 2}$	-160,32	-307,56	-31,59
$FT_{\theta 3}$	-160,32	-307,56	-31,59
$FT_{\phi 4}$	160,32	307,56	31,59

5.2.3 Sintonização do PID via lugar das raízes

Um outro procedimento bastante utilizado para configurar um controlador PID é o chamado método do lugar das raízes. Através desse método, estuda-se a trajetória dos polos a medida que o ganho é variado.

Escolhendo-se o valor 120 como ganho proporcinal, rearranjando a função de transferência 5.21 conforme expressão abaixo (onde $G(s) = \frac{N}{D}$ a função de transferência em malha aberta), pode-se determinar o ganho integrativo, conforme observado na Fig. 44. Analisando o lugar das raízes do termo $\frac{N}{s(D+k_pN)}$, Fig.44, verifica-se que para que os polos possuam constante de amortecimento inferior a 0,7 (região entre as linhas inclinadas) e possua tempo de estabilização inferior a 1,2 segundos (região fora do semi-círculo), o ganho deve ser superior a 250. Portanto, será escolhido o valor 270 para o ganho.

$$\Phi = 1 + (k_p + \frac{k_i}{s})\frac{N}{D} = 0 \tag{5.33}$$

$$0 = 1 + k_i\frac{N}{s(D + k_pN)} \tag{5.34}$$

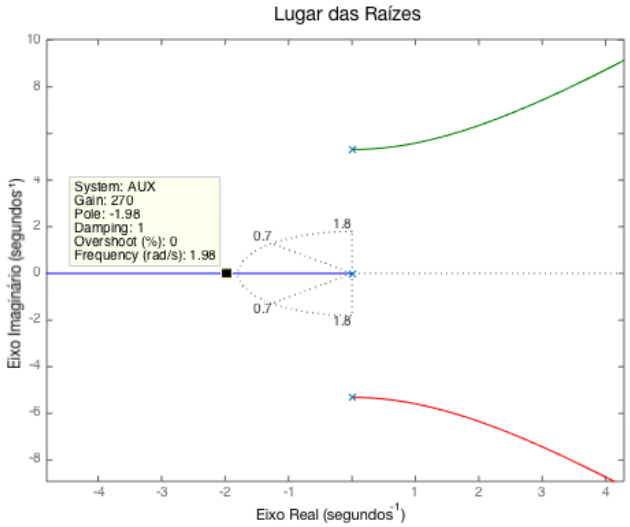


Figura 44 – Lugar das raízes do termo $\frac{N}{s(D+k_pN)}$.

Por fim, rearranjando novamente a equação característica da função de transferência, após adiconados os ganhos proporcional e integrativo, pode-se determinar o ganho derivativo, conforme ilustrado na Fig. 45.

$$0 = 1 + k_d \frac{s^2 N}{sD + (k_p s + k_i)N} \tag{5.35}$$

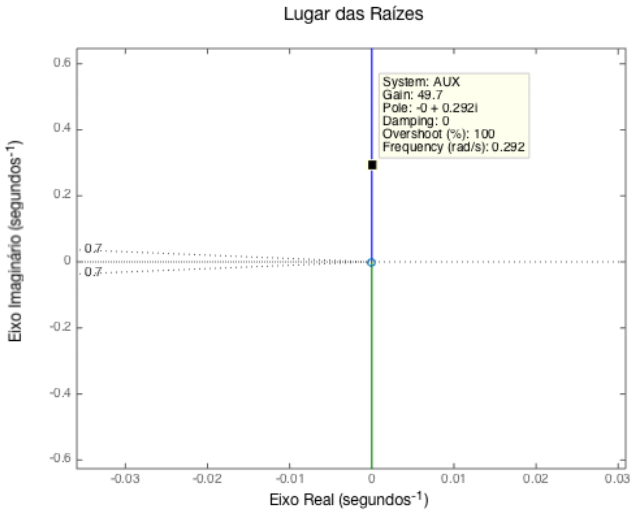


Figura 45 – Lugar das raízes do termo $\frac{s^2 N}{sD+(k_p s+k_i)N}$.

Dessa forma, aplicando-se o mesmo processo para as demais funções de transferência, chega-se aos valores de ganhos descritos na Tab. 15.

Tabela 15 – Ganhos do PID - Lugar das Raízes

FT	k_P	k_I	k_D
$FT_{\theta 1}$	120	270	49,7
$FT_{\phi 2}$	-120	-270	-49,7
$FT_{\theta 3}$	-120	-270	-49,7
$FT_{\phi 4}$	120	270	49,7

5.2.4 Simulação numéricas dos controladores PIDs

De forma a comparar o desempenho dos controladores projetados por meio dos métodos apresentados nos tópicos acima, foram realizadas simulações numéricas de uma entrada degrau unitária para cada um dos controladores e todas as respostas foram plotadas em um mesmo gráfico, conforme pode-se observar na Fig. 46.

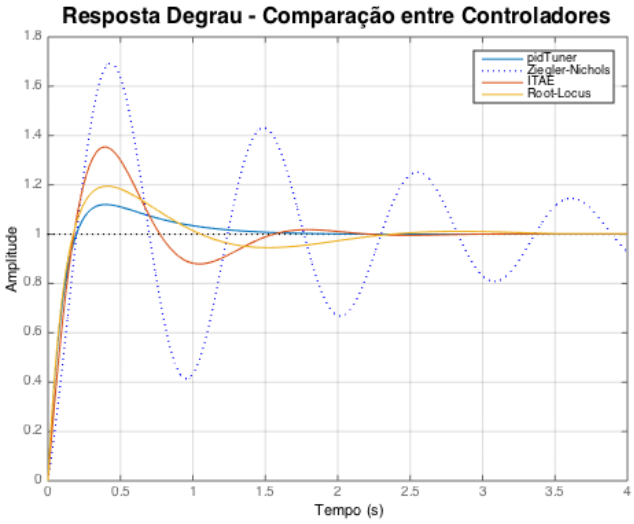


Figura 46 – Comparação entre as respostas à entrada degrau unitário, produzidas pelos controladores PIDs projetados.

Imediatamente observa-se que o melhor desempenho obtido corresponde ao PID projetado por meio comando "*pidTuner*", como era de se esperar, uma vez que através desse comando calibra-se o tempo de estabilização e o percentual de seobressinal, observando o efeito na resposta de forma simultânea.

O pior desempenho é observado para o controlador projetado por meio do método de Zigler-Nichols, haja visto que o tempo de estabilização e o percentual de sobressinal divergem bastante dos valores esperados pelo projeto.

Uma outra maneira de analisar o desempenho dos controladores consiste em traçar o ganho e a fase de cada função de transferência, em função da frequência. Tal visualização é conhecida como diagrama de Bode. Analisando a Fig. 47, a qual ilustra o diagrama de Bode das funções de transferência dos sistema com os controles PIDs projetados nos tópicos anteriores, pode-se levantar algumas conclusões importantes. A primeira conclusão é que os sistemas que mais se aproximam do comportamento característico do amortecimento crítico são os controladores projetados por meio da ferramenta "*pidTuner*"e do método do "lugar das raízes". Isso porque não se observa picos de ganho conforme são observados nos gráficos do método ITAE e do método Ziegler-Nichols.

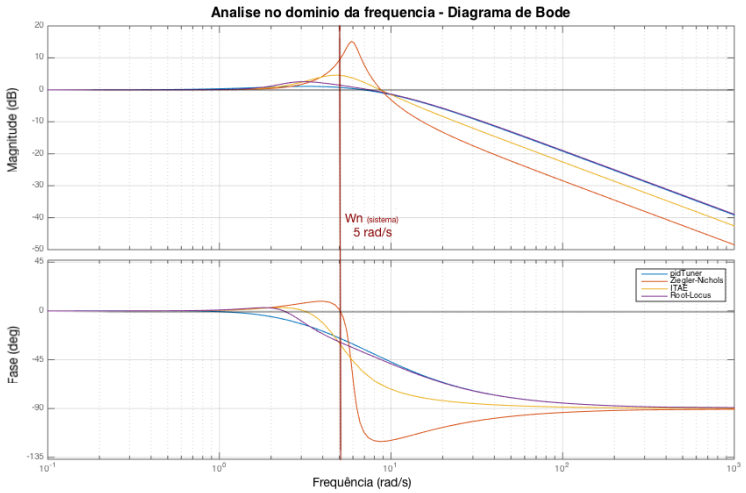


Figura 47 – Comparação entre as respostas à entrada degrau unitário, produzidas pelos controladores PIDs projetados.

Em segundo lugar, analisando o gráfico da fase, verifica-se que apenas o controlador projetado por meio do método de Ziegler-Nichols possui o sinal de resposta em fase com o sinal de entrada, sendo que para todos os outros controladores ocorre um atrasado entre o sinal de entrada e o sinal de saída próximo de 25 graus.

Por fim, verifica-se que para frequências próximas da frequência natural do sistema (5 rad/s), o ganho do sistema é praticamente nulo para os controladores projetados através da ferramenta "*pidTuner*"e do método do "lugar das raízes", enquanto que para os outros dois controladores, ocorre amplificação do sinal de entrada (sistema sub-amortecido), o que pode gerar problemas de instabilidade. Portanto, é de se esperar

que os controladores projetados por meio da ferramenta ”*pidTuner*”e do método do ”lugar das raízes”apresentam melhores resultados.

5.3 Avaliação da estabilidade com base no critério de NYQUIST

Em se tratando da análise de estabilidade, no domínio da frequência, de um sistema de controle, um critério bastante utilizado é o chamado critério de Nyquist. Por meio desse critério, torna-se possível determinar se um sistema será estável em malha fechada, avaliando, para tanto, a equação característica da função de transferência em malha fechada.

Resumidamente, o procedimento para se avaliar a estabilidade do sistema, de acordo com o o critério de Nyquist, pode ser descrito pelos itens a seguir:

- Determinar a equação característica da função de transferência do sistema em malha fechada, $1 + PID(s)G(s) = 0$, onde PID(s) é a função de transferência do controlador e G(s) a função de transferência da planta.

$$T(s) = \frac{PID(s)G(S)}{1 + PID(s)G(s)} \tag{5.36}$$

- Mapear o caminho de ”s”do domínio de ”s”até o domínio de ”PID(s)G(s)”, onde o caminho de ”s”engloba todo semi-plano direito.
- Determinar o número de voltas completas ao redor do ponto -1 + 0j, no domínio ”PID(s)G(s)”.
- Determinar ”P”, o número de polos de $1 + PID(s)G(s)$.
- Determinar o valor de ”Z”, número de zeros de $1 + PID(s)G(s)$, por meio da equação $N = Z - P$.
- Se ”$Z > 0$”, então o sistema é instável.

Analisando o diagrama de Nyquist para cada uma das funções de transferência em malha fechada, Fig. 48 verifica-se que em nenhum dos diagramas o ponto $-1 + 0j$ é circulado, o que indica que N = 0 para todos os casos. Além disso, sabendo P = 0, em todos os casos, conclui-se que Z = 0 em todos os casos. Portanto, verifica-se que os sistemas projetados são estáveis.

Uma outra maneira de verificar a estabilidade dos sistemas controlados seria analisar a equação característica das funções de transferência do sistema em malha fechada e determinar os polos de cada sistema. Ao realizar esse procedimento, nota-se que todos os polos estão posicionados no semi-plano negativo, o que indica estabilidade dos sistemas.

- $$T(s)_{pidTuner} = \frac{11s^2 + 23,8s + 4,23}{s^3 + 11s^2 + 23,8s + 4,23} \tag{5.37}$$

  Polos: $[-8,14; -2,67; -0,20]$

- $$T(s)_{Ziegler-Nichols} = \frac{3,76s^2 + 38,11s + 96,47}{s^3 + 3,76s^2 + 38,11s + 96,47} \tag{5.38}$$

  Polos: $[-0,52 + 5,92i; -0,52 - 5,92i; -2,73]$

- $$T(s)_{ITAE} = \frac{7,43s^2 + 37,71s + 72,35}{s^3 + 7,43s^2 + 37,71s + 72,35} \tag{5.39}$$

  Polos: $[-2,24 + 4,41i; -2,24 - 4,41i; -2,96]$

- $$T(s)_{LugardasRaizes} = \frac{11,29s^2 + 28,23s + 63,51}{s^3 + 11,29s^2 + 28,23s + 63,51} \tag{5.40}$$

  Polos: $[-8,92; -1,18 + 2,39i; -1,18 - 2,39i]$

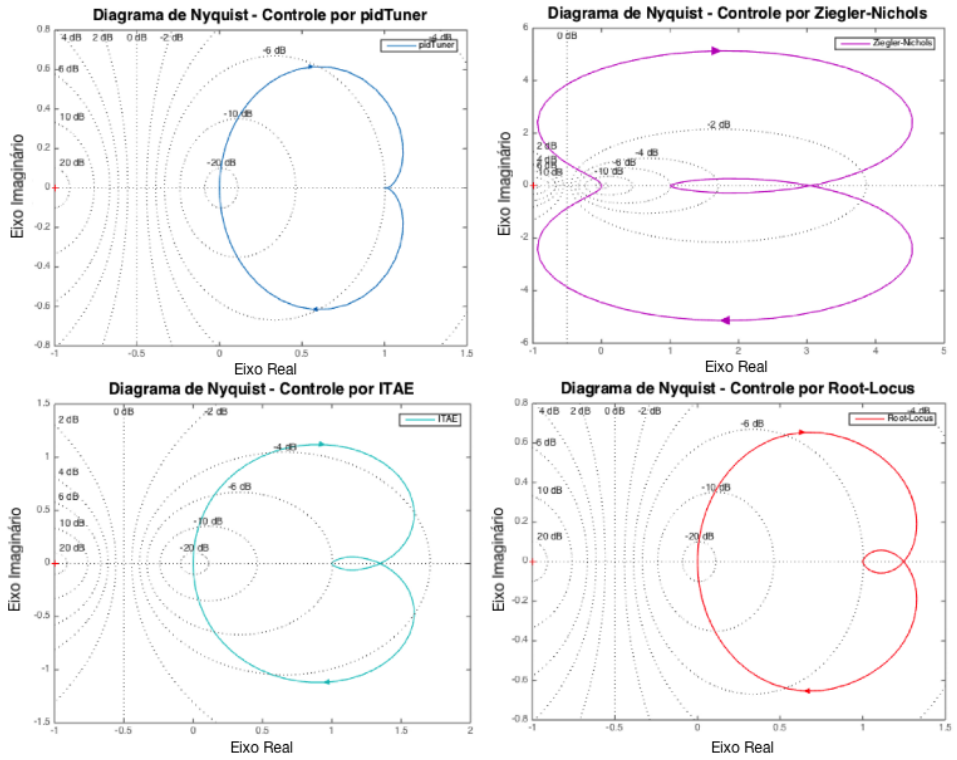


Figura 48 – Diagrama de Nyquist das funções de transferência com controle PID

6 IMPLEMENTAÇÃO DOS SISTEMAS DE CONTROLE

Após projetado os sistemas de controle com base nas teorias de controle moderno e controle clássico, será demonstrado a implementação dos sistemas no protótipo. Dessa forma, será visto, neste capítulo, a configuração do circuito eletrônico do veículo, bem como a lógica de programação utilizada para o controlador moderno e clássico.

6.1 Configuração do circuito eletrônico

Conforme mencionado no capítulo 3, o protótipo é constituído de 4 atuadores (conjunto hélice, motor e ESC), um sensor e uma bateria. Além disso, foi utilizado o microcontrolador Arduino UNO para processar as informações enviadas pelo sensor e enviar comandos aos atuadores. Vale mencionar que devido à problemas técnicos com o sensor mencionado no terceiro capítulo, foi utilizado o sensor MPU6050, o qual é constituído por um giroscópio e acelerômetro. O sensor MPU6050 pode ser visto na Fig. 49.

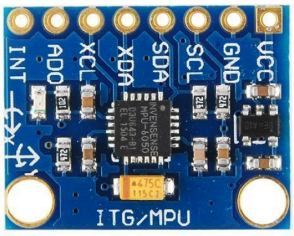


Figura 49 – Sensor MPU6050 (Acelerômetro e Giroscópio)

A configuração do circuito eletrônico pode ser vista no diagrama apresentado na Fig.50. Observa-se que para alimentação do arduino foi utilizado uma bateria separada (6V), como forma de evitar uma eventual sobrecarga do microcontrolador, o que poderia danificar o componente. O protótipo final pode ser visto na Fig.51.

Deve-se ressaltar que, durante os testes, utilizou-se um cabo umbilical para fornecer tensão aos motores ao invés de embarcar a bateria, pois dessa forma o consumo de carga pelos motores é menor devido à menor massa, resultando em maior intervalo de teste entre recargas.

Uma vez finalizado o protótipo, sua massa foi medida com o intuito de validar a massa total estimada durante a etapa de seleção dos componentes. Verificou-se, que o veículo com a bateria embarcada totalizava 1318 gramas, enquanto que sem a bateria,

987 gramas. Como os testes foram realizados sem a bateria, a massa considerada foi 987 gramas.

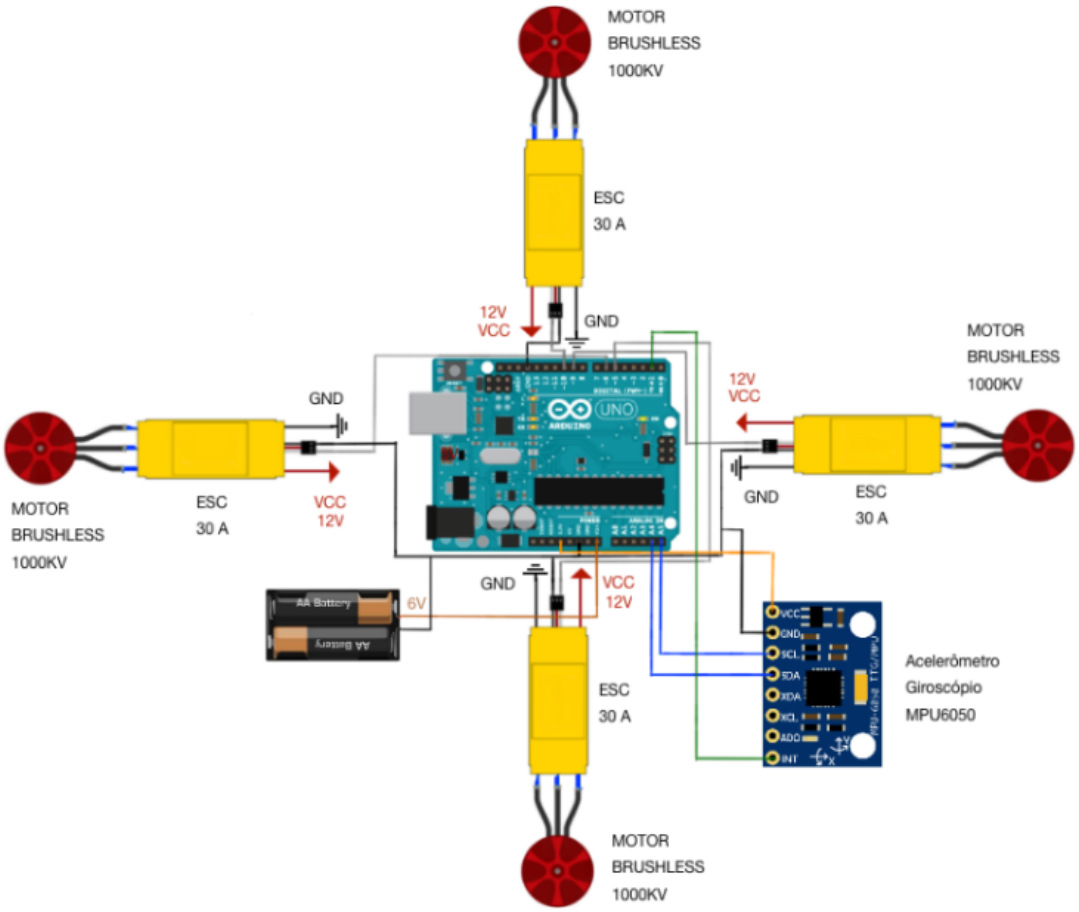


Figura 50 – Circuito eletrônico do protótipo)

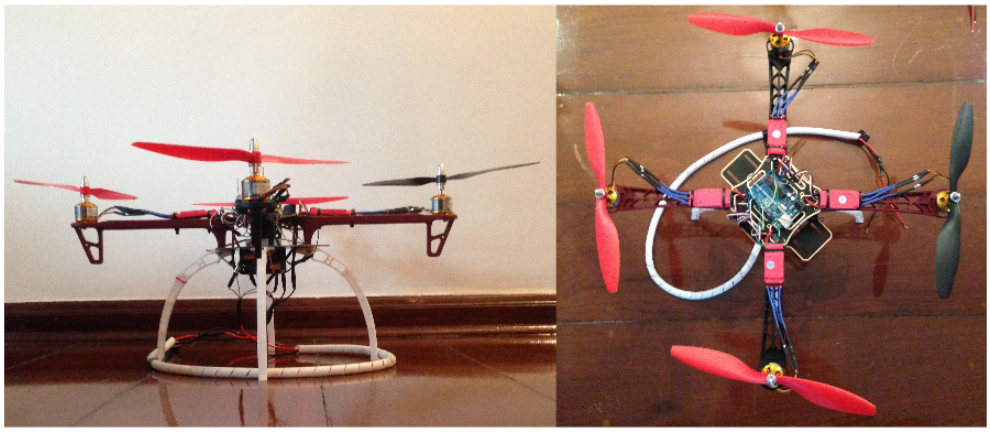


Figura 51 – Imagem do protótipo construído para teste dos sistemas de controle

Levando-se em consideração o teste dos atuadores, realizado no capítulo 3, cada motor deveria trabalhar em 398 rad/s (3802 rpm), para produzir 2,4N de empuxo na condição de operação. Entretanto, durante a realização dos testes, observou-se a necessidade de aumentar a rotação dos motores para 515 rad/s (4918 rpm) para atingir a condição de operação, isto é, condição em que o veículo é capaz de pairar no ar.

6.2 Resposta do motor em função do valor PWM

Para que fosse possível implementar os sistemas de controle utilizando o microcontrolador Arduino, foi necessário realizar o mapeamento da dinâmica do motor em função da largura de pulso enviado pelo microcontrolador. Portanto, foi estabelecido uma relação entre a largura de pulso (medida em milissegundos) enviada pelo Arduino e o empuxo produzido pelo motor, conforme ilustrado no gráfico da Fig.52.

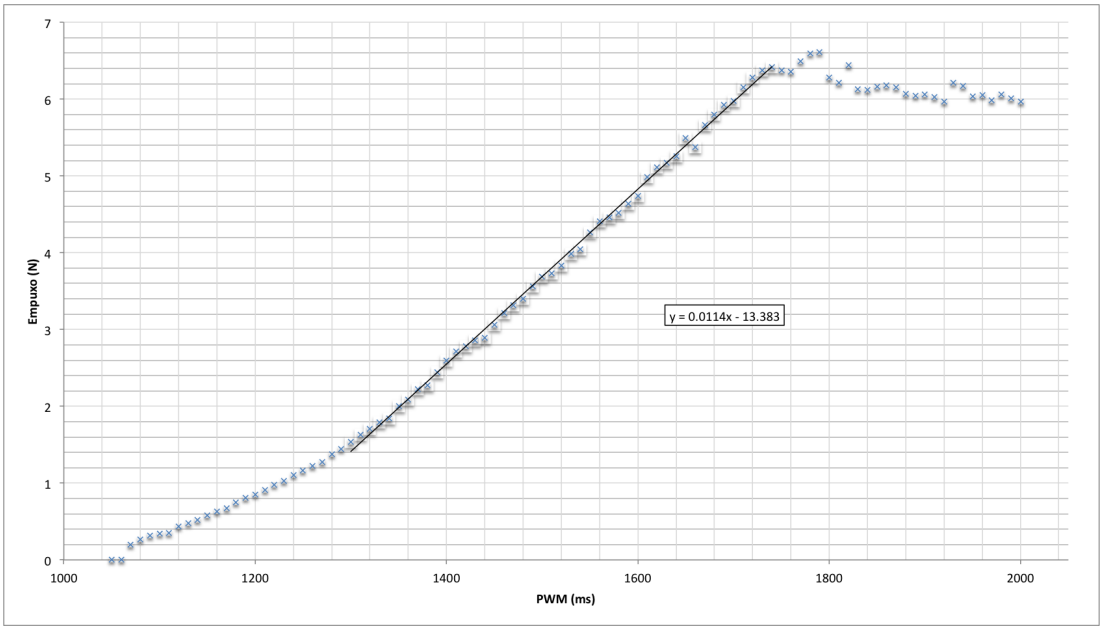


Figura 52 – Relação entre a largura de pulso (PWM) enviada pelo Arduino e o empuxo produzido pelo motor

De acordo com o gráfico, verifica-se que para um motor produzir 4,3 N de empuxo, 515 rad/s de rotação, deve ser enviado uma largura de pulso de 1550 milissegundos. Dessa forma, o ponto de operação, em termos da largura de pulso que deve ser gerada pelo microcontrolador, corresponde à 1550 milissegundos.

6.3 Projeto do filtro complementar

Conforme mencionado no capítulo 3, o acelerômetro fornece ao sistema a medida da aceleração linear enquanto que o giroscópio fornece a medida da velocidade angular. Dessa forma, as variáveis de estado do sistema dinâmico do quadricóptero podem ser calculadas por meio de transformações trigonométricas e integrações dos valores medidos.

Entretanto, apesar dos giroscópios geralmente apresentarem boa leitura da velocidade angular, a integração de seu valor provoca um erro crescente no cálculo da posição angular. Isso porque durante o processo de integração também é integrado o ruído da leitura em estado estacionário. Por outro lado, a leitura do acelerômetro apresenta bastante ruído quando submetido à acelerações quaisquer, porém fornecendo boas leituras em condições estáticas (ROMANO, 2014). Por essas razões, concluí-se que a melhor alternativa para contornar os problemas apresentados pelos sensores é a fusão de suas leituras por meio de um filtro complementar. Dessa forma, aproveita-se a leitura do acelerômetro em condição estática e a leitura do giroscópio em condição dinâmica.

Por meio do filtro complementar, portanto, pondera-se a leitura do acelerômetro com a leitura do giroscópio, conforme expressão a seguir.

$$angulo_{novo} = 0.98(angulo_{antigo} + giro(dt)) + 0.02(acc) \tag{6.1}$$

Sendo, $angulo_{antigo}$ o ângulo medido no instante anterior, $angulo_{novo}$ o ângulo a ser estimado, ”giro”o ângulo medido pelo giroscópio e acc a posição angular obtida através da transformação trigonométrica da leitura das componentes da aceleração linear, conforme expressões a seguir.

$$\theta = ATAN\frac{acc_x}{\sqrt{acc_y^2 + acc_x^2}} \tag{6.2}$$

$$\phi = ATAN\frac{acc_y}{\sqrt{acc_x^2 + acc_x^2}} \tag{6.3}$$

Sabendo-se que em condições estáticas o valor lido pelo giroscópio será muito pequeno, verifica-se que o ângulo estimado será praticamente o valor de entrada ajustado pela leitura do acelerômetro. Por outro lado, para condições dinâmicas, a parcela

ajustada pela leitura do giroscópio passa a ser bastante significativa, dado seu fator multiplicativo.

A título de ilustração, a comparação entre as acelerações estimadas por meio do giroscópio, do acelerômetro e do filtro complementar podem ser vistas no gráfico da Fig.53. A figura ilustra os ângulos medidos pelos sensores durante um dos testes do protótipo.

Observa-se no gráfico que o sinal puro obtido pelo giroscópio e a posição angular obtida através do acelerômetro possuem bastante ruído (curvas vermelha e verde, respectivamente), enquanto que o sinal filtrado por meio do filtro complementar (curva laranja) não apresenta ruído. A curva azul representa a posição angular de rolagem, resultado da integração do sinal lido pelo giroscópio. Observa-se que esse sinal também não apresenta ruído, devido à integração, porém acumula-se erro em regime, como pode ser facilmente observado para instantes maiores que 20 segundos.

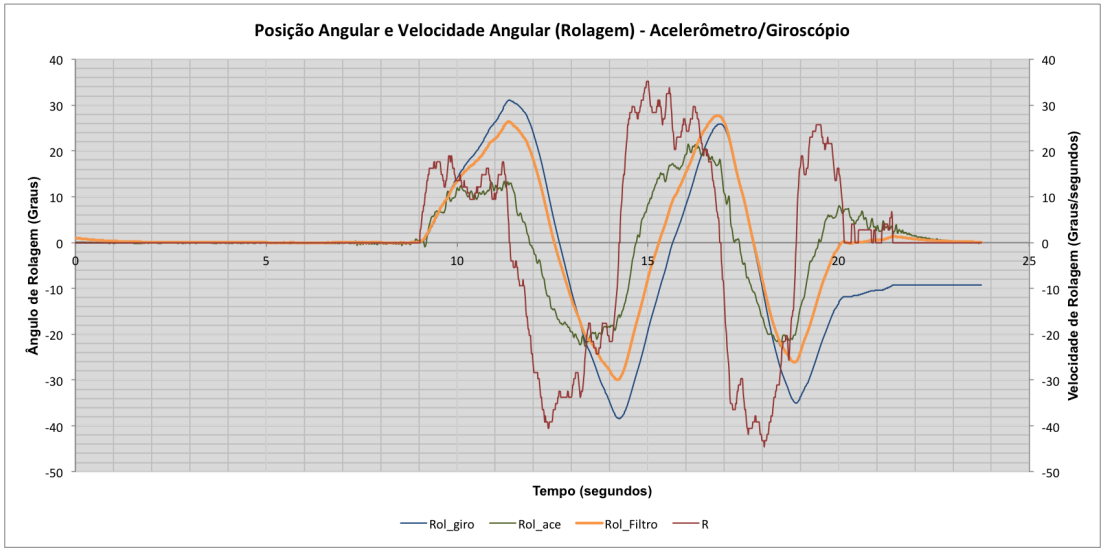


Figura 53 – Aplicação do filtro complementar

Para obtenção do posição angular a partir da velocidade angular medida pelo giroscópio, utilizou-se método de integração as expressões apresentadas a seguir.

$$\begin{aligned} \phi &= \phi + giro_P(dt) \\ \theta &= \theta + giro_Q(dt) \\ \psi &= \psi + giro_R(dt) \end{aligned} \tag{6.4}$$

Sendo, $[\phi, \theta, \psi]$ os ângulos estimados, $[giro_P, giro_Q, giro_R]$ a leitura do giroscópio de cada uma das componentes da velocidade angular e dt o intervalo de tempo entre duas medições consecutivas.

6.4 Implementação do controle moderno

Além das matrizes de ganhos calculadas no capítulo 4, foi necessário utilizar as expressões auxiliares, permitindo estimar as variáveis de estado não medidas, por meio do observador de estados. Dessa forma, as componentes da velocidade linear e da posição linear, foram obtidas a partir da integração da aceleração linear medida pelo acelerômetro, conforme apresentado nas expressões a seguir.

$$\begin{aligned} U &= U + a_x(dt) \\ V &= V + a_y(dt) \\ W &= W + a_z(dt) \end{aligned} \tag{6.5}$$

$$\begin{aligned} X &= X + X(dt) + 0.5(a_x)(dt)^2 \\ Y &= Y + y(dt) + 0.5(a_y)(dt)^2 \\ Z &= Z + z(dt) + 0.5(a_z)(dt)^2 \end{aligned} \tag{6.6}$$

Sendo, $[U,V,W]$ as velocidades lineares estimadas, $[X,Y,Z]$ as posições lineares estimadas e $[a_x,a_y,a_z]$ a leitura do acelerômetro de cada uma das componentes da aceleração linear.

A partir dos cálculos mencionados, nota-se que em conjunto com as variáveis medidas, todas as 12 variáveis de estados estão disponíveis para serem utilizadas na rotina de cálculo dos sistemas de controle, sejam elas obtidas por meio de leitura direta ou estimadas.

Por fim, o programa escrito para controlar o movimento do veículo foi desenvolvido na plataforma disponibilizado pela empresa fabricante do microcontrolador Arduino, e pode ser encontrado na seção de anexos. Além disso, para leitura dos valores medidos pelo sensor MPU6050 através do Arduino,foi utilizado como suporte o programa desenvolvido por (ROWBERG, 2017). Seu código encontra-se disponível para livre uso na página virtual *gitHub*.

6.5 Implementação do controle clássico

Para implementação do controle clássico, por outro lado, foi necessário desenvolver expressões para cálculo do valor correspondente a cada termo do controle PID, conforme descrito pelas expressões a seguir.

$$P = erro(kP); \tag{6.7}$$

$$I = I + (erro(kI))dt; \tag{6.8}$$

$$D = \frac{(angulo_{velho} - angulo_{novo})(kD)}{dt}; \tag{6.9}$$

Sendo kp, ki e kd os ganhos proporcional, integrativo e derivativo, respectivamente, *erro* a diferença entre o ângulo medido e o ângulo de referência e P, I e D as parcelas proporcional, integrativa e derivativa, respectivamente, do ganho total. Vale destacar que a soma parcelas resulta, diretamente no valor que deve ser acrescido ou diminuído do ponto de operação em microssegundos. Isto é, acrescido ou diminuído do valor 1580, que configura a rotação de correspondente ao ponto de operação.

Observa-se que a constante "P"cresce de forma muito mais rápida que suas componentes "I"e "D". Isso porque seu valor é apenas função do ganho multiplicado pelo erro. O fator integrativo, por outro lado, depende do acúmulo do erro para que sua parcela passe a ser significante. Em termos técnicos, isso equivale à remover o erro em regime permanente. O ganho derivativo, por sua vez, tem o papel de antecipar o movimento do quadricóptero e impedir que movimento bruscos sejam realizados, o que evita instabilidade.

7 ANÁLISE COMPARATIVA DOS CONTROLADORES

7.1 Controle moderno

O sistema de controle moderno foi testado utilizando-se o programa descrito no anexo (NOME), conforme mencionado. Os testes foram realizados primeiramente em um suporte fixo (gangorra), de forma a verificar o código desenvolvido por meio da análise de um único movimento (movimento de rolagem do veículo), conforme observado na Fig.54.

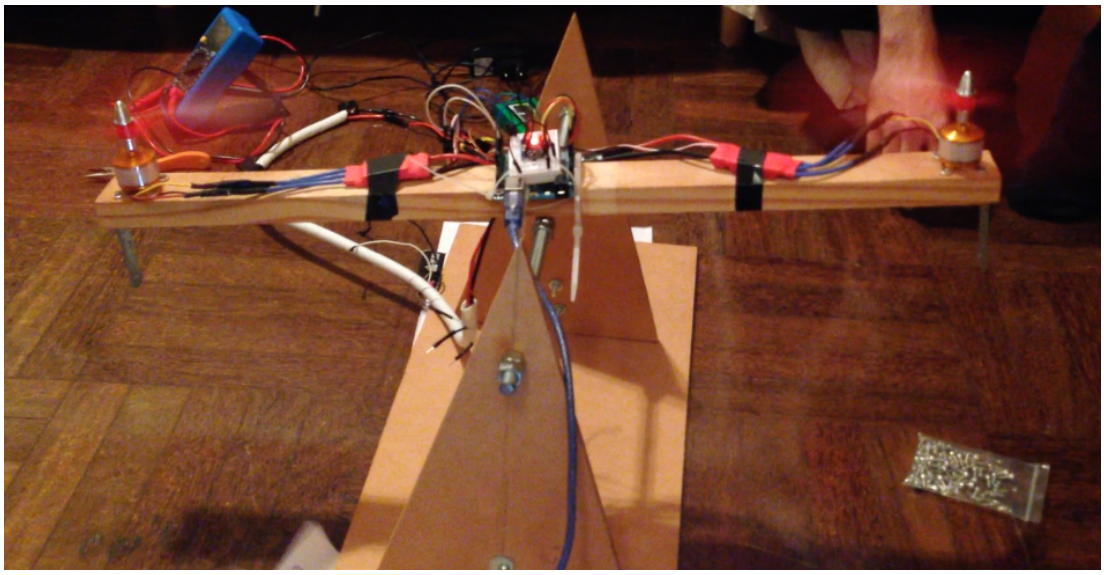


Figura 54 – Estrutura desenvolvida para simulação do movimento de rolagem do quadricóptero

Para execução desse teste, o sistema de controle foi alimentado apenas com as leituras da posição angular, filtrado por meio do filtro complementar descrito no seção anterior, e da velocidade angular de rolagem filtrada por meio de um filtro passa-baixas, conforme descrito pela expressão a seguir.

$$R_{OUT} = 0,75(R_{IN}) + 0,25(R_{OLD}) \tag{7.1}$$

Sendo R_{OUT} a velocidade angular de rolagem filtrada, R_{IN} a velocidade angular de rolagem lida pelo sensor e R_{OLD} a velocidade angular de rolagem no instante anterior.

Verifica-se que o sinal filtrado de fato atenua os sinais em alta frequência, porém acrescenta um atraso no sinal enviado para o sistema de controle, conforme observado na Fig.55.

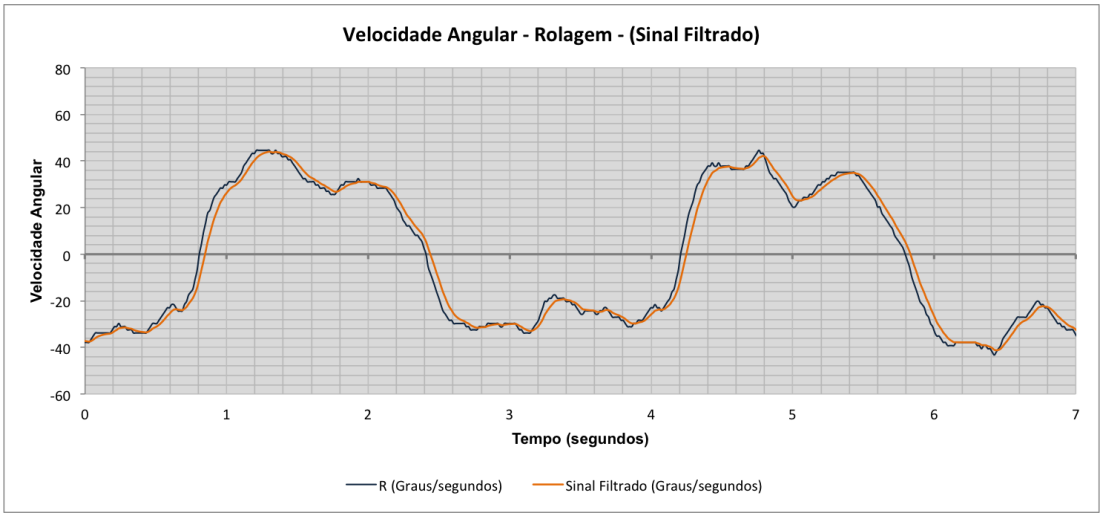


Figura 55 – Comparativo do sinal da velocidade angular de rolagem medido pelo sensor e do sinal filtrado

Durante o teste do movimento de rolagem, verificou-se que o sistema atinge estabilidade quando levado ao equilíbrio de forma lenta e sem a presença de nenhum distúrbio. Entretanto, ao introduzir um impulso em um dos lados da gangorra, nota-se que o sistema torna-se instável, muito provavelmente devido ao fato do ganho correspondente à velocidade angular estar desajustado, o que faz com que o erro, diferença entre o sinal medido e o sinal de referência, não convirja para zero em um intervalo suficientemente curto, quando a velocidade angular atinge valores elevados. Até a publicação deste relatório, a matriz de ganho [K] foi recalculada diversas vezes com o intuito de melhorar o desempenho do sistema de controle, sem obter um resultado satisfatório.

O teste do controle moderno também foi realizado no protótipo construído, conforme pode ser observado na Fig.56, com o intuito de verificar o desempenho do sistema de controle completo e não somente do controle do ângulo e velocidade de rolagem. Entretanto, o veículo apresentou muita instabilidade, qualquer que fosse a condição de teste.

Verificou-se que um dos motivos principais da instabilidade do quadricóptero foi o erro acumulado nas variáveis de estado estimadas: posição linear $[X,Y,Z]$ e velocidade linear $[U,V,W]$. Isso porque a estimação dessas variáveis foi realizada por meio da integração do sinal do acelerômetro, o que faz com que os sinais estimados acumulem erros proveniente da integração dos ruídos dos sinais medidos. Tal problema

poderia ser contornado utilizando-se um sensor de posição (GPS), o qual permitiria que a posição espacial do veículo pudesse ser medida ao invés de estimada, além de atribuir maior precisão na estimação da velocidade linear, uma vez que poderia ser utilizado um filtro complementar para fundir a leitura do acelerômetro e do sensor de posição.

Figura 56 – Estrutura desenvolvida para simulação do movimento de rolagem do quadricóptero

A Fig.57 ilustra a leitura da aceleração do protótipo na direção u_Y, realizada a partir do teste do movimento de rolagem descrito anteriormente, bem como a integração do seu sinal, resultando na velocidade linear V. Também é ilustrado na figura a integração do sinal da velocidade, resultando na posição linear Y. Nota-se que o sinal da posição Y cresce rapidamente conforme cresce o tempo de medição, o que não condiz com a aceleração medida, uma vez que essa oscila em torno da aceleração nula.

Também a título de ilustração do problema de integração dos ruídos no processo de estimação das variáveis de estado, a Fig.58 ilustra a leitura da velocidade angular de rolagem, bem como sua integração, resultando no ângulo de rolagem, para o mesmo teste mencionado no parágrafo anterior. Também é ilustrado na figura o ângulo de rolagem estimado pelo filtro complementar. Analisando a figura, observa-se que a velocidade angular medida pelo giroscópio (curva azul) oscila ao redor da velocidade nula. Logo, a posição angular estimada pela integração deste sinal (curva vermelha) deveria oscilar ao redor de zero, assim como descito pelo sinal da posição angular estimada por meio do filtro complementar (curva verde). As linhas pontilhadas vermelha e verde indicam, respectivamente, o valor médio dos sinais da posição angular estimada pela integração

do sinal lido pelo giroscópio e estimada por meio do filtro complementar. Nota-se que o erro, diferença entre as retas, tende a crescer conforme cresce o tempo de medição.

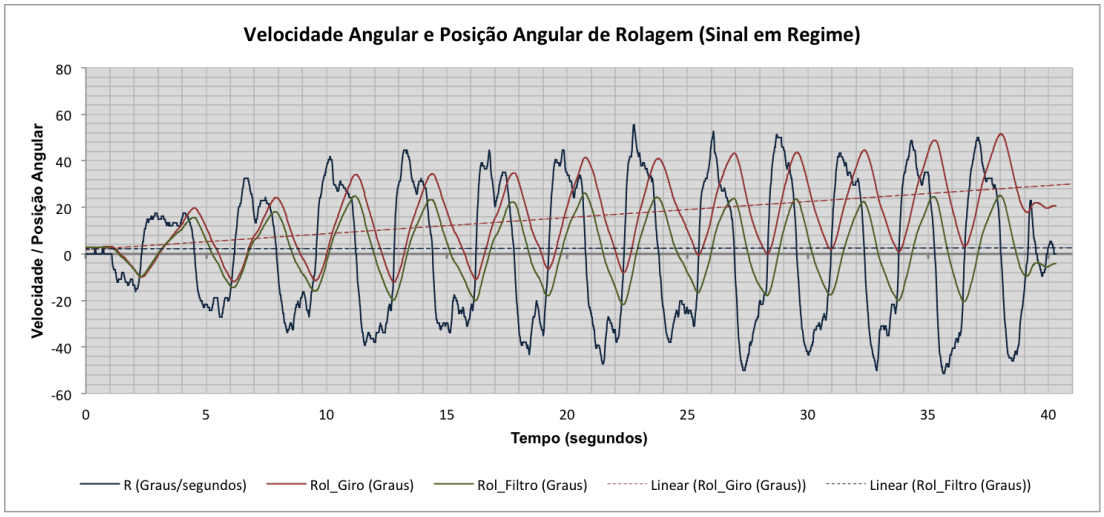


Figura 57 – Leitura do acelerômetro e respectiva integração durante teste do movimento de rolagem.

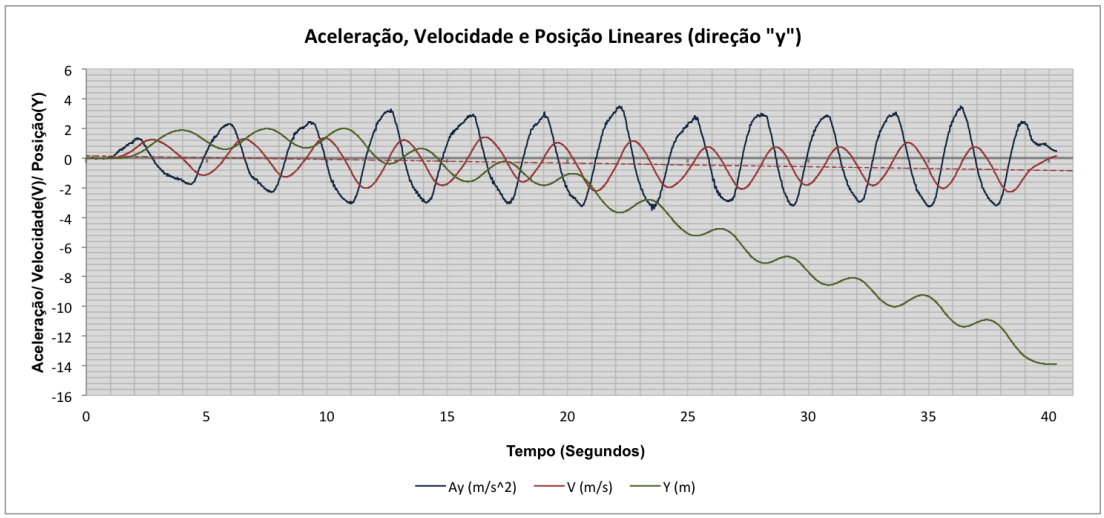


Figura 58 – Leitura do giroscópio e respectivas integrações durante teste do movimento de rolagem.

7.2 Controle Clássico

Diferentemente do controle moderno, o controle clássico apresentou bom desempenho para estabilização do protótipo, apesar de não ser possível controlar sua posição espacial. Em geral, o sistema de controle, após realizado os devidos ajustes mostrou-se capaz de estabilizar o veículo mesmo quando este era submetido a distúrbios aleatórios.

Assim como no procedimento de teste do sistema de controle moderno, o desempenho do controle clássico foi primeiramente testado somente para o movimento de rolagem, com auxílio da gangorra construída, conforme ilustrado na Fig. 59. Verificou-se que para esse teste inicial, os ganhos calculados por meio dos métodos Ziegler-Nichols e ITAE, além dos ganhos obtidos através da ferramenta *pidTuner*, vide cálculo descrito no capítulo 5, foram satisfatoriamente precisos para estabilizar o movimento de rolagem. Apenas o sistema de controle com os ganhos obtidos por meio do lugar das raízes resultou em um movimento estável, porém com oscilações de baixa amplitude em alta frequência, o que indicou que o ganho derivativo estava elevado. Os ganhos inicialmente utilizados no controle do movimento de rolagem, assim como descritos no capítulo 6, são reescritos a seguir. Deve-se observar que, considerando-se um método específico utilizado para a obtenção dos ganhos, os mesmos ganhos foram utilizados para todos os motores, apenas invertendo-se o sinal dos ganhos para motores posicionados diametralmente oposto, conforme o sentido positivo/negativo do movimento de arfagem e rolagem.

Tabela 16 – Ganhos do PID - PidTuner/Ziegler-Nichols/ITAE/Lugar das Raizes

FT	k_P	k_I	k_D
pidTuner	101,18	17,99	46,75
$Ziegler-Nichols$	162,00	410,13	16,00
ITAE	160,32	307,56	31,59
LugardasRaizes	120,00	270,00	49,70

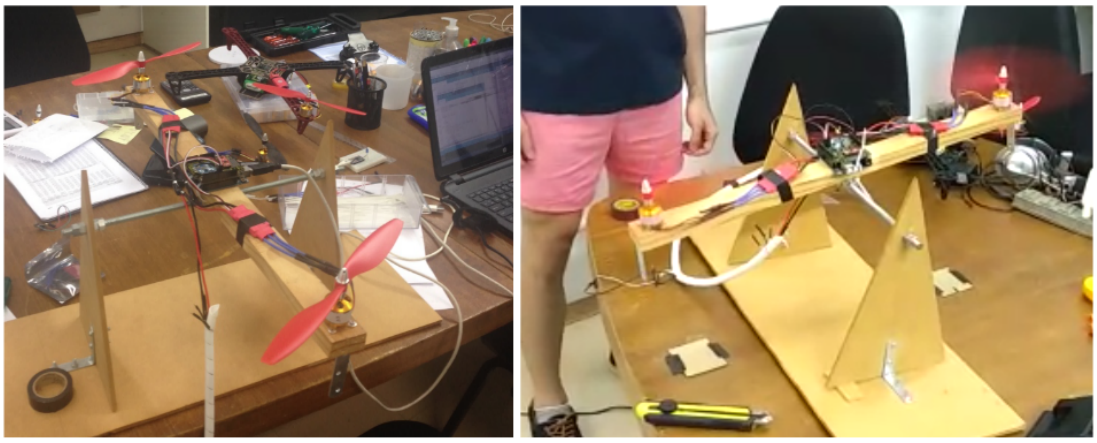


Figura 59 – Análise do desempenho do controle clássico por meio do controle do movimento de rolagem

Ao testar os controladores no protótipo, notou-se que para os ganhos calculados através da ferramenta *pidTuner*, o tempo de estabilização não era suficientemente rápido para manter o veículo pairado, indicando que os ganhos proporcional e integrativo não eram suficientemente elevados. No caso dos sistema de controle baseados nos ganhos obtidos pelos métodos Ziegler-Nichols e ITAE, verificou-se que o protótipo foi capaz de estabilizar no ar, sendo o método ITAE o de melhor desempenho. Por outro lado, através do controle projetado por meio do método do lugar das raízes, não foi possível controlar o protótipo, haja visto que este apresentou muita instabilidade.

Os ganhos do controlador ITAE foram, então, ajustados de forma a melhorar seu desempenho, por meio de várias etapas de tentativa e erro. Primeiramente, observou-se que ao aumentar o ganho proporcional para valores acima de 180, o movimento do veículo tornava-se instável apresentando inicialmente oscilações de baixa amplitude e alta frequência, as quais tornavam-se rapidamente oscilações de alta frequência e grande amplitude. Dessa forma, o ganho proporcional foi elevado para 170, porém ainda próximo do valor calculado através do método ITAE.

Em seguida, aumentando-se o ganho integrativo em relação ao ganho calculado pelo método ITAE, verificou-se que para valores superiores a 380 o sistema corrigia os ângulos de rolagem e arfagem além do necessário, isto é, o sistema de controle passava a apresentar um alto percentual de *overshoot*. Sendo assim, escolheu-se o valor 375 para o ganho integrativo.

Finalmente, variando-se o ganho derivativo, observou-se que o sistema tornava-se instável para ganhos superiores à 40, apresentando oscilações de alta frequência e baixa amplitude. Por essa razão, manteve-se o valor do ganho derivativo em 32. A tabela Tab. 17 ilustra os ganhos utilizados na versão final do controle, conforme descrito nos parágrafos acima.

Tabela 17 – Ganhos ITAE ajustados

FT	k_P	k_I	k_D
ITAEAjustado	170	375	32

Uma vez realizados os ajustes mencionados, observou-se que o tempo de acomodação dos ângulos de rolagem e arfagem diminuíram, bem como sua oscilação ao redor da posição de referência, $\phi = \theta = 0$.

A Fig.60 apresenta uma amostra do sinal do ângulo de rolagem, coletada durante um dos testes do sistema de controle ajustado. Observa-se que, em geral, o sinal oscila ao redor da referência, $\theta = 0$, conforme mencionado e que o sinal de atuação enviado para os motores é espelhado em relação à reta que passa pelo ponto de operação

(PWM = 1550). Esta última observação era de se esperar, uma vez que motores diametralmente opostos devem possuir atuações antagônicas e que durante a modelagem dos motores considerou-se a mesma dinâmica para todos, portanto mesmos ganhos.

Além disso, analisando o comportamento dos atuadores, nota-se que durante o intervalo de aproximadamente 10 a 22 segundos, o ângulo de rolagem permanece próximo a -3 graus, fazendo com que o sinal no motor 1 aumente enquanto o sinal no motor 2, diminui. Portanto, a resposta do atuador mostra-se bastante coerente com o esperado.

O gráfico da Fig.61, por sua vez, apresenta uma seção ampliada do gráfico ilustrado na Fig.60, refente à medida do ângulo de rolagem, onde é possível analisar o tempo de acomodação do ângulo em questão sob atuação do sistema de controle. Nota-se que o sistema demora aproximadamente 0,6 segundos para estabilizar o ângulo de rolagem, atendendo ao requisito de projeto, no qual o tempo de acomodação deveria ser inferior a 1,2 segundos.

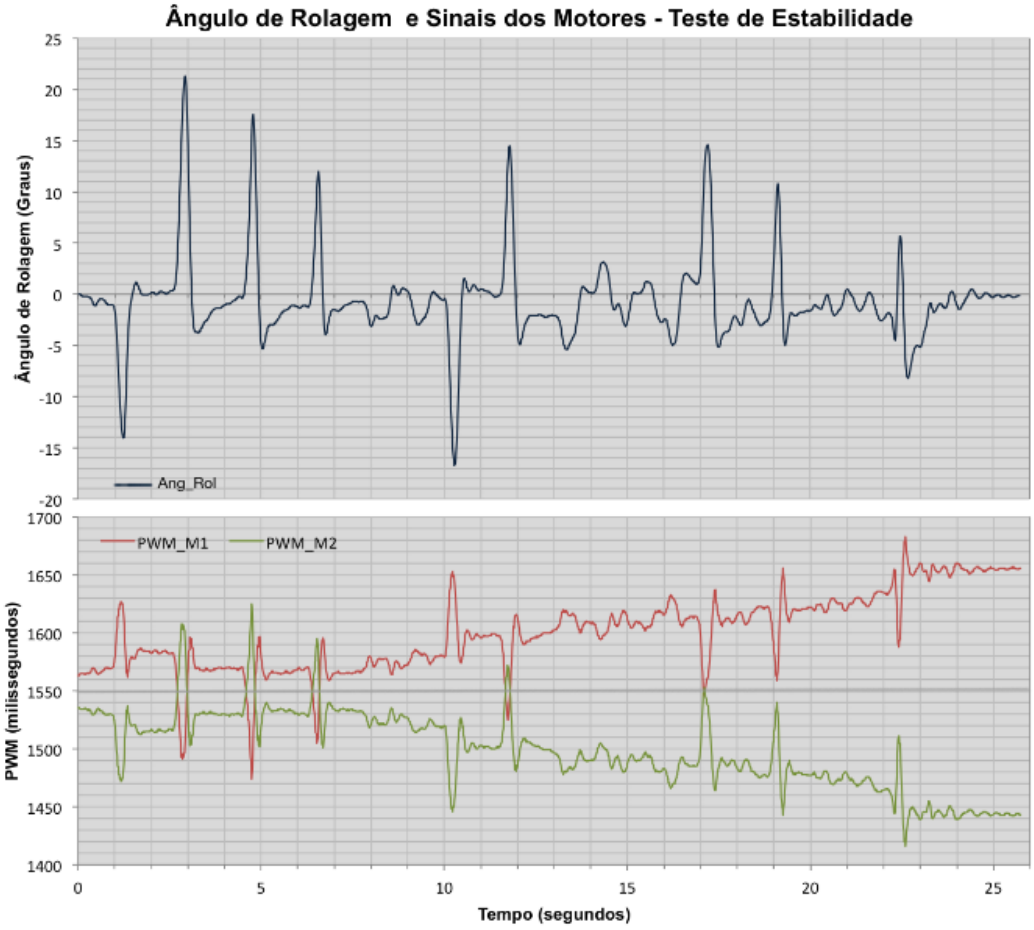


Figura 60 – Ângulo de rolagem e sinais dos motores 1 e 2, medidos durante teste de controle do veículo.

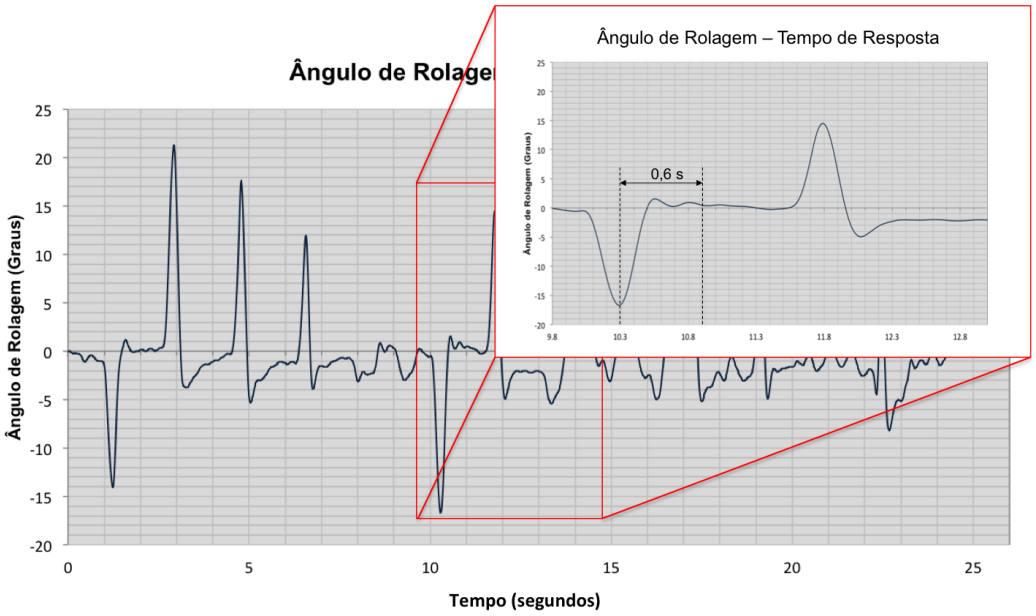


Figura 61 – Ângulo de rolagem, ampliado, medido durante teste de controle do veículo

8 CONCLUSÃO E TRABALHO FUTURO

Por meio do projeto do sistema de controle do quadricóptero, desenvolvido ao longo do trabalho de graduação do curso de engenharia mecânica na Escola Politécnica da Universidade de São Paulo e que é documentado no presente relatório, foi possível aprofundar na disciplina de controle da dinâmica de sistemas mecânicos, mais precisamente nas técnicas relacionadas às teorias de controle moderno e clássico. O escopo do trabalho presumia o projeto de dois sistemas de controle, que fossem equivalentes no objetivo de estabilização da dinâmica do veículo em questão, desenvolvidos, separadamente, com base nas duas teoria de controle mencionadas.

Além disso, era de comum objetivo do trabalho proposto, a construção de um protótipo funcional, no qual seriam testados os sistemas de controle.

Ao longo do projeto, pode-se notar as principais características de cada teoria de controle, bem como suas respectivas vantagens e desvantagens. Primeiramente, verificou-se que a sofisticação da teoria de controle moderno é muito superior a do controle clássico, haja visto que o projeto do controlador por meio da teoria de controle moderno leva em consideração todo acoplamento existente na dinâmica do veículo. Esta característica não se observa na teoria de controle clássica, pois os movimento são controlados de forma independente. Por essa razão, o potencial do controlador projetado através da teoria de controle moderno é muito maior do que o controlador equivalente, projetado pela teoria de controle clássica. Entretanto, por considerar a interdependência entre as variáveis de estados da dinâmica do quadricóptero, a teoria de controle moderna exige que os ganhos projetados sejam muito mais precisos que os ganhos projetados pela teoria de controle clássica. Dessa forma, a sintonização do controle clássico torna-se menos complicada. Por essa razão, o controlador clássico possui maior robustez que o controlador moderno, isto é, o controlador comporta-se melhor às incertezas na representação do modelo da planta e/ou às incertezas na representação dos distúrbios.

Por fim, os testes dos controladores projetados no protótipo construído confirmaram a maior facilidade de implementação do controle clássico, tendo em vista que apenas este controlador demonstrou um bom desempenho. Isso porque, fora as características mencionadas, o controle moderno utilizou variáveis estimadas por meio de integração numérica a partir das varáveis medidas, resultando em variáveis de estados estimadas com grande percentual de erro. No caso do controle clássico, utilizou-se apenas variáveis medidas diretamente, não sendo necessário a estimação por meio de integração.

Dados os motivos mencionados, apenas o controlador clássico foi capaz de estabilizar o protótipo e mantê-lo pairado. No entanto, esse resultado não significa que o desempenho do controle clássico é superior ao do controle moderno. Conforme mencionado, o resultado obtido apenas confirma a maior facilidade de implementação do controle clássico. Sendo assim, propõe-se como trabalho futuro a adaptação do sistema de controle projetado considerando-se a adição de um sensor de posicionamento (GPS). Dessa forma, as variáveis de estados relacionadas à posição espacial, até então estimadas por meio da integração dupla do sinal medido pelo acelerômetro, poderiam ser diretamente medidas, removendo o erro gerado em sua estimação. Além disso, por meio da fusão do sinal do sensor de posição com o sinal do acelerômetro, utilizando-se um filtro complementar, por exemplo, seria possível obter maior precisão na estimação das variáveis de estado referentes à velocidade linear do veículo.

REFERÊNCIAS

ADAM. *Search and Rescue Missions By Drones*. Favorite Drones. 2015. Disponível em: <http://favoritedrones.com/search-and-rescue-missions-by-drones/>. Acesso em: 12 outubro de 2016.

ALEX. *LiPo Batteries - How to choose the best battery for your drone*. Drone Trest. 13 de setembro de 2015. Disponível em: <http://www.dronetrest.com/t/lipo-batteries-how-to-choose-the-best-battery-for-your-drone/1277>. Acesso em: 23 janeiro de 2017.

ÅSTRÖM, K., MURRAY, R. *Feedback system*. Princeton University Press. Versão v2.11b. Princeton. New Jersey. 2012.

AVIASTAR. *Helicopters Engineering - Convertawings Model A*. 2011. Disponível em: <http://
www.aviastar.org/helicopters_eng/convertawings.php>. Acesso em: 12 outubro 2016.

BELL HELICOPTER, P. *Military Aircraft: The Bell-Boeing V-22*. Disponível em: <http://www
.bellhelicopter.com/en/aircraft/military/bellV-22.cfm>. Acesso em: 12 outubro 2016.

BERTRAND, S., GUENARD, N., HAMEL, T., PIET-LAHANIER, H., ECK, L. *A hierarchical controller for miniature VTOL UAVs: design and stability analysis using singular perturbation theory*. Control Eng. Pract. 19(10), 1099–1108.2011

BLESCH, C. *Navy funds rutgers to develop drone equally adept at flying and swimming*. Rutgers Today. Outubro. 2015. Disponível em: <http://news.rutgers.edu/research-news/navy-funds-rutgers-develop-drone-equally-adept-flying-and-swimming/20151022/#.WAGG5ZM>. Acesso em: 14 outubro 2016.

BOEING. *V-22*. 2016. Disponível em: <http://www.boeing.com/defense/v-22-osprey/#/gallery>. Acesso em: 12 outubro 2016.

BOUABDALLAH, S., NORTH, A., SIEGWART, R.*PID vs LQ control techniques applied to an indoor microquadrotor In Intelligent Robots and Systems*. Swiss Federal Institute of Technology. Proceedings of 2004 IEEE/RSJ International Conference, vo-

lume 3. Lousanne. Suiça. 2004.

BROWN. *Top Surveillance Drones Review: Best Models You Can Buy Right Now*. My Drone Lab. 2016. Disponível em: <http://mydronelab.com/best-pick/surveillance-drones-for-sale.htm
l>. Acesso em: 12 outubro 2016.

CABRAL, E., *PMR2400 - Controle e automação II*. Departamento de engenharia mecatrônica da Escola Politécnica da Universidade de São Paulo. Disponível em: <http://sites.poli.usp.br/d/
pmr2400/3-%20Lineariza%C3%A7%C3%A3o%20de%20sistemas.pdf>. Acesso em: 12 outubro 2016.

CHALONS-TOURISME. *Etienne Oehmichen*. 2016. Disponível em: <http://www.chalons-tourisme.com/en/discover /celebrities/etienne-oehmichen/>. Acesso em: 12 outubro 2016.

D'ANDREA, R. *Flying machine arena*. Raffaello D'Andrea - Dynamic Works. 2016. Disponível em: <http://raffaello.name/projects/flying-machine-arena/>. Acesso em: 12 outubro 2016.

DILLOW, C. *UPenns's amazing quad-rotor drones now work in teams to lift heavy payloads together*. Popular Science. Julho. 2010. Disponível em: <http://www.popsci.com/technolog
cle/2010-07/video-upenns-quadcopters-now-work-teams-lift-heavy-payloads>. Acesso em: 14 outubro 2016.

DOMINGUES, J. *Quadrotor prototype*. Universidade Técnica de Lisboa. Lisboa. Publicado em outubro de 2009.

FRANÇA, L., MATSUMURA, A., 2011, *Mecânica Geral*. Editora Edgar Blucher LTDA, Instituto Mauá de Tecnologia, 3a Edição, 2011.
HENRIQUES, B.*Estimation and Control of a Quadrotor Attitude*. Dissertação de mestrado, Departamento de Engenharia Mecânica, Instituto Superior Técnico, Universidade Técnica de Lisboa Disponível, Junho de 2011.

HIRZEL, T. *AnalogWrite*. Arduino Forum. Disponível em: <https://www.arduino.cc/en/Tutorial/PWM>. Acesso em: 27 fevereiro 2017.

SHUAIB, A., AHMED, M., *Robust PID Control System Design Using ITAE Performance Index (DC Motor Model)*. International Journal of Innovative Research in Sci-

ence, Engineering and Technology. Vol. 3, Versão 8, Publicado em agosto de 2014.

JAIN, P. *Magnetometers*. Engineers Garage. Publicado em julho de 2012. <https://www.engineersgarage.com/articles/magnetometer> Acesso em: 24 abril 2017.

LIANG, O. *LiPo battery beginner guide for drones and quadcopters*. Disponível em: <https://os
carliang.com/lipo-battery-guide/>. Acesso em: 17 março 2017.

LIMA, R. *PME2443 Microprocessadores e Controle Digital*.Publicado em 13 de novembro de 2014.

MAELE, P. *Getting the angular position from gyroscope data*. Pieter-Jan Website. Setembro de 2012. Disponível em: <http://www.pieter-jan.com/node/7>. Acesso em: 02 abril 2017.

MOKHTARI, A., BENALLEGUE, A., DAACHI, B. *Robust feedback linearization and gh8 controller for a quad rotor unmanned aerial vehicle*. Journal of Electrical Engineering. 2006.

NAUGHTON, R. *Aviation and Aeromodelling - Interdependent Evolutions and Histories*. Hargrave The Pioneers. CTIE Monash University. 2002. Disponível em: <http://www.ctie.monash.
edu.au/hargrave/breguet.html>. Acesso em: 12 outubro 2016.

NICE, E.*Design of a four rotor hovering vehicle*. Trabalho de graduação. Cornell University. Nova York. Estados Unidos. 2004

NISE, N.*Control Systems Engineering*. John Wiley & Sons, Inc. California State Polytechnic University. 7ª edição. Califórnia. Estados Unidos. 2015

OGATA, K.*Discrete-Time Control Systems*. Pearson, Second Edition, University of Minnesota, 2016.

OLIVEIRA, M.*Modeling, Identification and Control of a Quadrotor Aircraft*. Department of Control Engineering. Faculty of Electrical Engineering Czech. Technical University. Praga. República Checa. 2011.

PACHECO, N., RESENDE, D. *Controle de estabilidade de aeromodelo tipo quadcopter autômato por lei de controle PID*. Pontifícia Universidade Católica de Minas

Gerais. Departamento de Engenharia Mecatrônica. Belo Horizonte. 2014.

PÉREZ-PIÑAR, D.,*Attitude control in Mikrokopter quadrotor*. Bits of understanding. Wordpress. Agosto de 2011. Disponível em: <https://bitsofunderstanding.wordpress.com/2011/08/20/attitude-control-in-mikrokopter-quadrotor/>. Acesso em: 25 novembro de 2016.

PFEIFER, E.*Projeto e Controle de um UAV* . Tese de mestrado, Departamento de Engenharia Elétrica, Escola Politécnica da Universidade de São Paulo, 2013.

POUNDS, P., MAHONY, R., GRESHAM, J. CORKE, P., ROBERTS, J.,*Towards Dynamically-Favourable Quad-Rotor Aerial Robots*. Proceedings of the Australasian Conference on Robotics and Automation, Canberra, Australia, 2004.

RAYTHEON. *Fancy Flying: Students Steer Drones to Victory in Ratheon UK contest*. 2015. Disponível em: <http://www.raytheon.co.uk/news/feature/game_of_drones_winners.html>. Acesso em: 12 outubro 2016.

ROGERS, S. *Control and Estimation with MATLAB*®. Createspace. 3ª edição. 2016.

RONZO. *Gyroscope*. SparkFun. Fevereiro de 2013. Disponível em: <https://learn.sparkfun.com/tutorials/gyroscope/how-a-gyro-works>. Acesso em: 02 abril 2017.

RUSSON, M. *Darpa debuts turbo mini drone that autonomously avoids obstacles while zooming at speeds of 45mph*. International Business Times. Fevereiro. 2016. Disponível em: <http://www.ibtimes.co.uk/darpa-debuts-turbo-mini-drone-that-autonomously-avoids-obstacles-while-zooming-speeds-45mph-1543971>. Acesso em: 12 outubro 2016.

SHOSA, P. *A brief history of Quadcopters and Multirotors*. Eye On Drones. 2015. Disponível em: <http://www.eyeondrones.com/brief-history-quadcopters-multirotors/>. Acesso em: 12 outubro 2016.

SPARKFUN. *SparkFun 9DoF Sensor Stick*. 2017. Disponível em: <https://www.sparkfun.com/products/13944>. Acesso em: 24 abril 2017.

TU BERLIN. *Introduction to Discrete-Time Control Systems*. Technische Universitat Berlin. 2014. Disponível em: <http://www.control.tu-berlin.de/images/5/51/DCS_Intro_CCS.pdf>. Acesso em: 12 maio 2017.

VERTICAL MAGAZINE. *The Weird and Wacky*.2012. Disponível em: <http://www.verticalma

g.com/features/the-weird-and-wacky-html/>. Acesso em: 12 outubro 2016.

YOKOTA, Y.*Desenvolvimento de um quadricóptero*. Departamento de engenharia mecânica. Escola Politécnica da Universidade de São Paulo. São Paulo. 2014.

ZEMALACHE, K., MAAREF, H.*Intelligent control for a drone by self-tunable fuzzy inference system*. 6th International Multi-Conference on Systems, Signals and Devices (SSD 2009). Março 2009. Djerba, Tunisia. (elec. proc.). 2009. <10.1109/SSD.2009.4956805>. <hal-00965125>.

ANEXO A - VALIDAÇÃO DO MODELO LINEAR

De forma a verificar a validade do sistema linear obtido no capítulo 2, será realizado neste anexo a simulação numérica do sistema linear e não-linear para um mesmo sinal de entrada, com auxílio do "*software*" MATLAB®.

O sinal utilizado para simulação dos modelos foi definido como um sinal degrau, representado pelo aumentado de rotação dos motores, conforme ilustrado na Fig. 62. Analisando a figura, verifica-se que um degrau é enviado para os motores 1 e 3, no instante igual a 2 segundos, fazendo com que eles atinjam a velocidade de rotação de $387 rad/s$, retornando para o valor de $287 rad/s$ após 2 segundos. Verifica-se que o mesmo ocorre para os motores 2 e 4 no instante igual a 6 segundos.

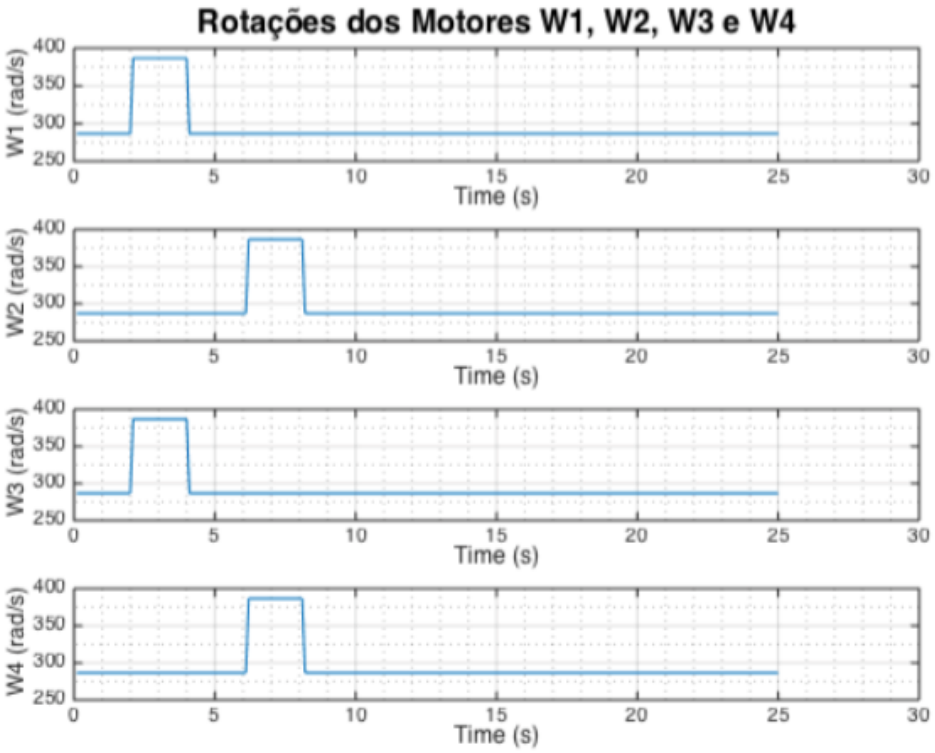


Figura 62 – Simulação numérica do sistema linear e não-linear: sinal degrau enviado aos motores

Analisando, por fim, as resposta ilustradas nas Fig. 63 e 64, verifica-se que a resposta do sistema linear é idêntica à resposta do sistema não-linear para o sinal de entrada degrau descrito na Fig. 62. Entretanto, as respostas podem divergir significativamente dependendo do sinal de entrada, caso a resposta do sistema distancie demasiadamente do ponto de operação.

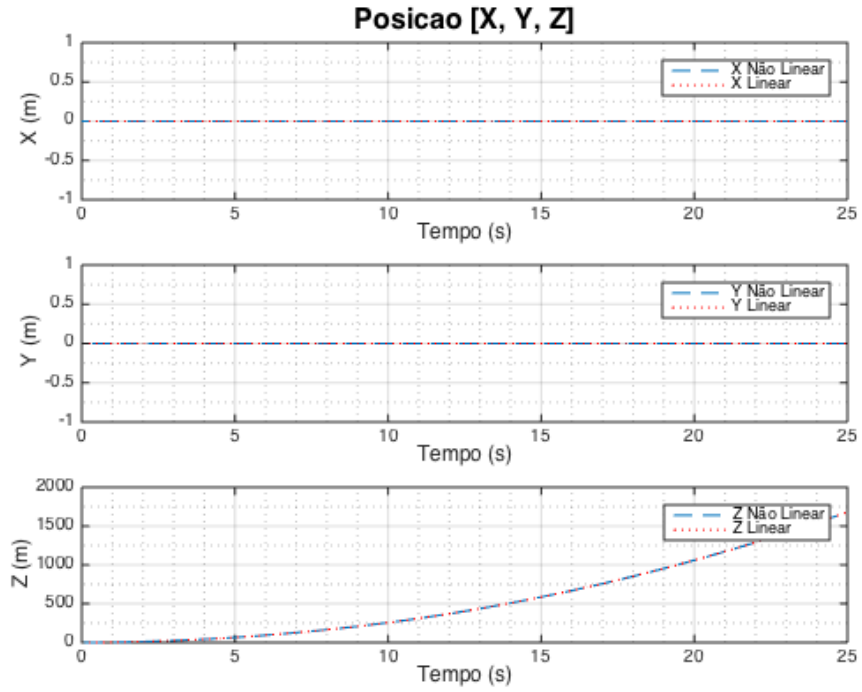


Figura 63 – Simulação numérica do sistema linear e não-linear: movimentos lineares

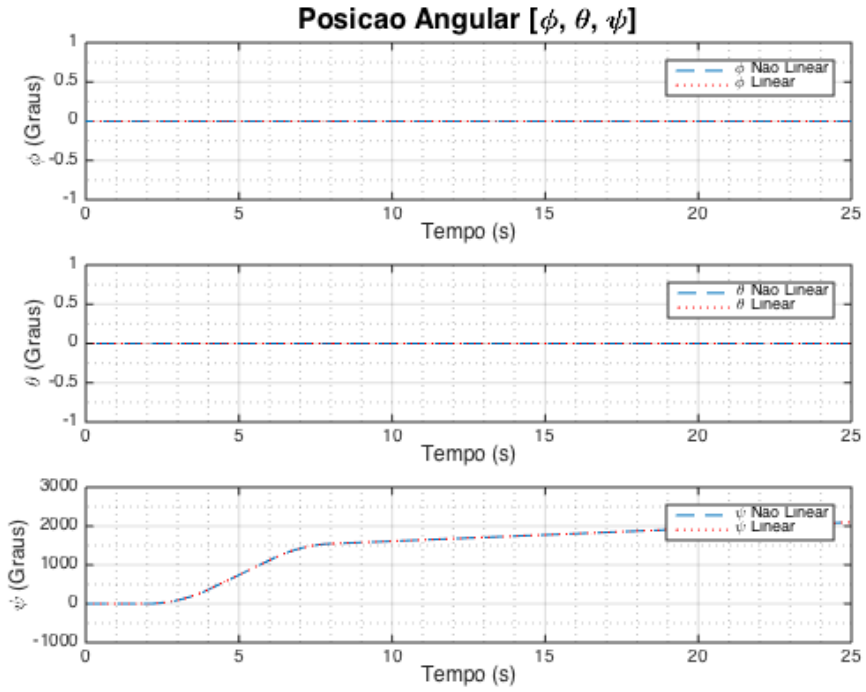


Figura 64 – Simulação numérica do sistema linear e não-linear: movimentos angulares

ANEXO B – CÓDIGO DE PROGRAMAÇÃO

SISTEMA DINÂMICO DO QUADRICÓPTERO

```
%_____________________Quadcopter Project_____________________
%              (Linear System Validation)
%
%           Code Developer: Felipe Mehsen Tufaile
%
%                     *Definitions*
%
%   Angular position (Euler Angles):
%
%         Phi - Roll Angle (Around "z" axis)
%         Theta - Pitch Angle (Around "x" axis)
%         Psi - Yaw Angle (Around "y" axis)
%
%
%   Linear velocity:
%
%         U - Linear velocity in "x" direction
%         V - Linear velocity in "y" direction
%         W - Linear velocity in "z" direction
%
%
%   Angular velocity:
%
%         P - Angular velocity around "x" axis
%         Q - Angular velocity around "y" axis
%         R - Angular velocity around "z" axis
%
%
%   Linear position:
%
%         X - linear position in "x" direction
%         Y - linear position in "y" direction
%         Z - linear position in "z" direction
%
%
%   State Vector and State Vector derivative:
%
%    1)   |   X   |      |     Xp     |
%    2)   |   Y   |      |     Yp     |
%    3)   |   Z   |      |     Zp     |
%    4)   |   U   |      |     Up     |
%    5)   |   V   |      |     Vp     |
%    6)   |   W   |      |     Wp     |
%    7)   |  Phi  |      |    Phip    |
%    8)   | Theta |      |   Thetap   |
%    9)   |  Psi  |      |    Psip    |
%   10)   |   P   |      |     Pp     |
%   11)   |   Q   |      |     Qp     |
%   12)   |   R   |      |     Rp     |
%

% *** NON-LINEAR SYSTEM SIMULATION ***
```

```
% ** Initial States **

% InitialState = [ X Y Z U V W Phi Theta Psi P Q R ]'
  InitialState = [ 0 0 -1.5 0 0 0  0    0    0  0 0 0 ]';

% ODE - Simulation Time
ti = 0;             %s
tff = 25;            %s

%ODE solver
[t, out] = ode45(@QuadEq,[ti tff],InitialState);

n = size(out);

% *** LINEAR SYSTEM SIMULATION ***

%ODE solver
[tlin, out_lin] = ode45(@QuadEqLIN,[ti tff],InitialState);

n_lin = size(out_lin);

a = 377;
c = 286.6880;
b = 370;

W1 = c;
W2 = c;
W3 = c;
W4 = c;

time = 0;
contador = 1;
while (time < 25)

if (time > 2 && time < 4)
    W1 = c+100;
    W3 = c+100;
end
if (time > 4)
   W1 = c;
   W3 = c;
end
if (time > 6 && time < 8)
    W2 = c+100;
    W4 = c+100;
end
if (time > 8)
    W2 = c;
    W4 = c;
end

time = time +0.1;
t_imp(contador,1) = time;
y_imp(contador,1) = W1;
y_imp(contador,2) = W2;
y_imp(contador,3) = W3;
y_imp(contador,4) = W4;
```

```
contador = contador + 1;
end

figure(1)
subplot(4,1,1);
plot(t_imp(:,1), y_imp(:,1))
title('RotaÁıes dos Motores W1, W2, W3 e W4', 'FontSize', 16);
xlabel('Time (s)', 'FontSize', 12);
ylabel('W1 (rad/s)', 'FontSize', 12);
grid on
grid minor
subplot(4,1,2);
plot(t_imp(:,1), y_imp(:,2))
xlabel('Time (s)', 'FontSize', 12);
ylabel('W2 (rad/s)', 'FontSize', 12);
grid on
grid minor
subplot(4,1,3);
plot(t_imp(:,1), y_imp(:,3))
xlabel('Time (s)', 'FontSize', 12);
ylabel('W3 (rad/s)', 'FontSize', 12);
grid on
grid minor
subplot(4,1,4);
plot(t_imp(:,1), y_imp(:,4))
xlabel('Time (s)', 'FontSize', 12);
ylabel('W4 (rad/s)', 'FontSize', 12);
grid on
grid minor

figure(2)
subplot(3,1,1);
plot (t(1:n),out(1:n,1),'--',tlin(1:n_lin),out_lin(1:n_lin,1),'r:');
title('Posicao [X, Y, Z]', 'FontSize', 16);
legend ('X N„o Linear','X Linear');
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('X (m)', 'FontSize', 12);
grid on
grid minor
subplot(3,1,2);
plot (t(1:n),out(1:n,2),'--',tlin(1:n_lin),out_lin(1:n_lin,2),'r:');
legend ('Y N„o Linear','Y Linear');
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('Y (m)', 'FontSize', 12);
grid on
grid minor
subplot(3,1,3);
plot (t(1:n),out(1:n,3),'--',tlin(1:n_lin),out_lin(1:n_lin,3),'r:');
legend ('Z N„o Linear','Z Linear');
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('Z (m)', 'FontSize', 12);
grid on
grid minor
%
figure(3)
subplot(3,1,1);
plot (t(1:n),out(1:n,7)*180/pi,'--',tlin(1:n_lin),out_lin(1:n_lin,7)*180/pi,'r:');
title('Posicao Angular [\phi, \theta, \psi]', 'FontSize', 16);
```

```
legend ('\phi N„o Linear','\phi Linear');
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('\phi (Graus)', 'FontSize', 12);
grid on
grid minor
subplot(3,1,2);
plot (t(1:n),out(1:n,8)*180/pi,'--',tlin(1:n_lin),out_lin(1:n_lin,8)*180/pi,'r:');
legend ('\theta N„o Linear','\theta Linear');
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('\theta (Graus)', 'FontSize', 12);
grid on
grid minor
subplot(3,1,3);
plot (t(1:n),out(1:n,9)*180/pi,'--',tlin(1:n_lin),out_lin(1:n_lin,9)*180/pi,'r:');
legend ('\psi N„o Linear','\psi Linear');
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('\psi (Graus)', 'FontSize', 12);
grid on
grid minor
```

SISTEMA - NÃO LINEAR

```
function dx_dt = QuadEq(t,x)

% *** DEFINICAO DAS CONSTANTES ***

m = 1.2; %Quadcopter mass in kilogram
KT = 0.0000146;
KM = 0.00000038;
dcg = 0.225; %distance to center of gravity
g = 9.8; % Gravity aceleration;

X = x(1);
Y = x(2);
Z = x(3);
U = x(4);
V = x(5);
W = x(6);
Phi = x(7);
Theta = x(8);
Psi = x(9);
P = x(10);
Q = x(11);
R = x(12);

%------
a = 377;
c = 286.6880;
b = 370;

W1 = c;
W2 = c;
W3 = c;
W4 = c;

if (t > 2 && t < 4)
    W1 = c+100;
    W3 = c+100;
```

```
end
if (t > 4)
   W1 = c;
   W3 = c;
end
if (t > 6 && t < 8)
    W2 = c+100;
    W4 = c+100;
end
if (t > 8)
    W2 = c;
    W4 = c;
end
%-----

S(1,1) = cos(Theta)*cos(Psi);
S(1,2) = sin(Phi)*sin(Theta)*cos(Psi) - cos(Phi)*sin(Psi);
S(1,3) = cos(Phi)*sin(Theta)*cos(Psi) + sin(Phi)*sin(Psi);
S(2,1) = cos(Theta)*sin(Psi);
S(2,2) = cos(Phi)*cos(Psi)+sin(Phi)*sin(Theta)*sin(Psi);
S(2,3) = sin(Theta)*cos(Phi)*sin(Psi) - sin(Phi)*cos(Psi);
S(3,1) = -sin(Theta);
S(3,2) = sin(Phi)*cos(Theta);
S(3,3) = cos(Theta)*cos(Phi);

Fx = 0;
Fy = 0;
Fz = -KT*(W1^2 + W2^2 + W3^2 + W4^2);

Mx = KT*(W4^2 - W2^2)*dcg;
My = KT*(W1^2 - W3^2)*dcg;
Mz = KM*(W1^2 - W2^2 + W3^2 - W4^2);

I11 = 0.0081;
I22 = 0.0081;
I33 = 0.0162;

dx_dt(1,1) = S(1,1)*U + S(1,2)*V + S(1,3)*W; %Xp
dx_dt(2,1) = S(2,1)*U + S(2,2)*V + S(2,3)*W; %Yp
dx_dt(3,1) = S(3,1)*U + S(3,2)*V + S(3,3)*W; %Zp

dx_dt(4,1) = Fx/m - g*sin(Theta) - (Q*W - R*V); %Up
dx_dt(5,1) = Fy/m + g*cos(Theta)*sin(Phi) - (R*U - P*W); %Vp
dx_dt(6,1) = Fz/m + g*cos(Theta)*cos(Phi) - (P*V - Q*U); %Wp

dx_dt(7,1) = P + Q*tan(Theta)*sin(Phi) + R*tan(Theta)*cos(Phi);
%Phip
dx_dt(8,1) = Q*cos(Phi) - R*sin(Phi); %Thetap
dx_dt(9,1) = Q*sin(Phi)/cos(Theta) + R*cos(Phi)/cos(Theta); %Psip

dx_dt(10,1) = Mx/I11 - (I33-I22)*Q*R/I11; %Pp
dx_dt(11,1) = My/I22 - (I11-I33)*R*P/I22; %Qp
dx_dt(12,1) = Mz/I33 - (I22-I11)*P*Q/I33; %Rp

end
```

SISTEMA - LINEAR

```
function dx_dt = QuadEqLIN(t,x)

% *** DEFINICAO DAS CONSTANTES ***

m = 1.2; %Quadcopter mass in kilogram
KT = 0.0000146;
KM = 0.00000038;
dcg = 0.225; %distance to center of gravity
g = 9.8; % Gravity aceleration;

X = x(1);
Y = x(2);
Z = x(3);
U = x(4);
V = x(5);
W = x(6);
Phi = x(7);
Theta = x(8);
Psi = x(9);
P = x(10);
Q = x(11);
R = x(12);

%------
a = 377;
c = 286.6880;
b = 370;

W1 = c;
W2 = c;
W3 = c;
W4 = c;

if (t > 2 && t < 4)
    W1 = c+100;
    W3 = c+100;
end
if (t > 4)
   W1 = c;
   W3 = c;
end
if (t > 6 && t < 8)
    W2 = c+100;
    W4 = c+100;
end
if (t > 8)
    W2 = c;
    W4 = c;
end
%-----

Fz = -KT*(W1^2 + W2^2 + W3^2 + W4^2);

Mx = KT*(W4^2 - W2^2)*dcg;
My = KT*(W1^2 - W3^2)*dcg;
Mz = KM*(W1^2 - W2^2 + W3^2 - W4^2);
```

```
I11 = 0.0081;
I22 = 0.0081;
I33 = 0.0162;

dx_dt(1,1) = U; %Xp
dx_dt(2,1) = V; %Yp
dx_dt(3,1) = W; %Zp

dx_dt(4,1) = - g*Theta; %Up
dx_dt(5,1) =   g*Phi; %Vp
dx_dt(6,1) =   Fz/m+g; %Wp

dx_dt(7,1) = P; %Phip
dx_dt(8,1) = Q; %Thetap
dx_dt(9,1) = R; %Psip

dx_dt(10,1) = Mx/I11; %Pp
dx_dt(11,1) = My/I22; %Qp
dx_dt(12,1) = Mz/I33; %Rp

end
```

CÓDIGO DO PROJETO DE CONTROLE

```
%_____________________Quadcopter Project_____________________
%                (Hovering Stability)
%
%          Code Developer: Felipe Mehsen Tufaile
%
%                    *Definitions*
%
%   Angular position (Euler Angles):
%
%        Phi - Roll Angle (Around "z" axis)
%        Theta - Pitch Angle (Around "x" axis)
%        Psi - Yaw Angle (Around "y" axis)
%
%
%   Angular velocity:
%
%        P - Angular velocity around "x" axis
%        Q - Angular velocity around "y" axis
%        R - Angular velocity around "z" axis
%
%
%   State Vector and State Vector derivative:
%
%    1)   |  Phi  |      |    Phip    |
%    2)   | Theta |      |   Thetap   |
%    3)   |  Psi  |      |    Psip    |
%    4)   |   P   |      |     Pp     |
%    5)   |   Q   |      |     Qp     |
%    6)   |   R   |      |     Rp     |
%


% *** LINEAR SYSTEM SIMULATION ***
```

```
%_____________________STATE SPACE MODEL_____________________
%                 (Continuous Time)

% Constants Declaration

massa = 1.2;   %1.026     % Quadcopter mass in kilogram
KTM = 0.026;  % Motor gain (Torque)
dcg = 0.225;      % Distance to center of gravity
g = 9.81;         % Gravity aceleration;

I11 = 0.011;
I22 = 0.011;
I33 = 0.020;

rsz = -0.007834;

k1 = 0.0115;
k2 = 0.0115;
k3 = 0.0115;
k4 = 0.0115;

Td = 0.2; %Time interval definition

%    X   Y   Z   U   V   W  Ph  Th  Ps  P   Q   R
A = [ 0   0   0   1   0   0   0   0   0   0   0   0;   %  1) X
      0   0   0   0   1   0   0   0   0   0   0   0;   %  2) Y
      0   0   0   0   0   1   0   0   0   0   0   0;   %  3) Z
      0   0   0   0   0   0   0   g   0   0   0   0;   %  4) U
      0   0   0   0   0   0  -g   0   0   0   0   0;   %  5) V
      0   0   0   0   0   0   0   0   0   0   0   0;   %  6) W
      0   0   0   0   0   0   0   0   0   1   0   0;   %  7) Phip
      0   0   0   0   0   0   0   0   0   0   1   0;   %  8) Thetap
      0   0   0   0   0   0   0   0   0   0   0   1;   %  9) Psip
      0   0   0   0   0   0   0   0   0   0   0   0;   % 10) Pp
      0   0   0   0   0   0   0   0   0   0   0   0;   % 11) Qp
      0   0   0   0   0   0   0   0   0   0   0   0;]; % 12) Rp

%         T1            T2             T3          T4
B = [      0             0              0           0        ;   %  1) X
           0             0              0           0        ;   %  2) Y
           0             0              0           0        ;   %  3) Z
           0             0              0           0        ;   %  4) U
           0             0              0           0        ;   %  5) V
           k1/massa      k2/massa       k3/massa    k4/massa ;   %  6) W
           0             0              0           0        ;   %  7) Phip
           0             0              0           0        ;   %  8) Thetap
           0             0              0           0        ;   %  9) Psip
           0        -k2*dcg/I11         0       k4*dcg/I11   ;   % 10) Pp
       k1*dcg/I22        0         -k3*dcg/I22      0        ;   % 11) Qp
       k1*KTM/I33   -k2*KTM/I33     k3*KTM/I33   -k4*KTM/I33 ;]; % 12) Rp

%     X   Y   Z   U   V   W       Ph          Th        Ps  P   Q   R
C = [ 1   0   0   0   0   0       0           0         0   0   0   0;     %  1)X
      0   1   0   0   0   0       0           0         0   0   0   0;     %  2)Y
      0   0   1   0   0   0       0           0         0   0   0   0;     %  3)Z
      0   0   0   0   0   0       0          -g         0   0   0   0;     %  4)ax
```

```
    0   0   0   0   0   0      g          0       0   0   0   0;     %  5)ay
    0   0   0   0   0   0      0          0      -1   0   0   0;     %  6)Psi
    0   0   0   0   0   0      0          0       0   1   0   0;     %  7)Pp
    0   0   0   0   0   0      0          0       0   0   1   0;     %  8)Qp
    0   0   0   0   0   0      0          0       0   0   0   1;];   %  9)Rp


%          T1                        T2                  T3                  T4
D = [      0                         0                   0                   0;              %  1)
           0                         0                   0                   0;              %  2)
           0                         0                   0                   0;              %  3)
      k1*dcg*rsz/I22                 0                -k3*dcg*rsz/I22        0;              %
4)
           0                   k2*(dcg/I11)*rsz          0                -k4*(dcg/I11)*rsz;     %
5)
           0                         0                   0                   0;              %  6)
           0                         0                   0                   0;              %  7)
           0                         0                   0                   0;              %  8)
           0                         0                   0                   0;];            %  9)


% System defition in the State Space approach (Continuos-time)
SYS_CT = ss(A,B,C,D);

%Open-Loop System Poles (Continuos Time)
Poles_Continuous_Open = eig(A);

%_____________________STUDY OF CONTROLABILITY_____________________

%Controlability Matrix
Controlability = ctrb(A,B);

%Rank of Controlability Matrix (has to be equal to 12)
RANKC = rank(Controlability);

%STUDY OF OBSEVABILITY

%Observability Matrix
Observability = obsv(A,C);

%Rank of Observability Matrix (has to be equal to 12)
RANKO = rank(Observability);

%_____________________STATE SPACE MODEL_____________________
%                (Discrete Time)


% System definition in the State Space approach (Discrete-time)
SYS_DT = c2d(SYS_CT,Td); %Transforming from continuous time to discrete time
Poles_Discrete_Open = exp(Poles_Continuous_Open*Td);
[G,H,Cd,Dd,Td] = ssdata(SYS_DT); %Get matrixes from SYS_DT


%_____________________POLES PLACEMENT_____________________
%

%Poles determination (continuous time approach)
```

```
%Stabilization time for each state variable
stable_time_phi = 1.501;
stable_time_theta = 1.502;
stable_time_psi = 1.503;
stable_time_X = 1.601;
stable_time_Y = 1.602;
stable_time_Z = 1.603;

%Overshoot for each state variable
overshoot_phi = 0.010;
overshoot_theta = 0.010;
overshoot_psi = 0.010;
overshoot_X = 0.020;
overshoot_Y = 0.020;
overshoot_Z = 0.020;

%Damping coeficient for each state variable
damping_phi = (((log(overshoot_phi))^2)^0.5)/(((log(overshoot_phi))^2+pi^2)^0.5);
damping_theta = (((log(overshoot_theta))^2)^0.5)/(((log(overshoot_theta))^2+pi^2)^0.5);
damping_psi = (((log(overshoot_psi))^2)^0.5)/(((log(overshoot_psi))^2+pi^2)^0.5);
damping_X = (((log(overshoot_X))^2)^0.5)/(((log(overshoot_X))^2+pi^2)^0.5);
damping_Y = (((log(overshoot_Y))^2)^0.5)/(((log(overshoot_Y))^2+pi^2)^0.5);
damping_Z = (((log(overshoot_Z))^2)^0.5)/(((log(overshoot_Z))^2+pi^2)^0.5);

%Natural frequency for each state variable
wn_phi = 4/(damping_phi*stable_time_phi);
wn_theta = 4/(damping_theta*stable_time_theta);
wn_psi = 4/(damping_psi*stable_time_psi);
wn_X = 4/(damping_X*stable_time_X);
wn_Y = 4/(damping_Y*stable_time_Y);
wn_Z = 4/(damping_Z*stable_time_Z);

%State variable pole
continuous_pole_phi = -damping_phi*wn_phi + wn_phi*((1-damping_phi^2)^0.5)*j;
continuous_pole_theta = -damping_theta*wn_theta + wn_theta*((1-damping_theta^2)^0.5)*j;
continuous_pole_psi = -damping_psi*wn_psi + wn_psi*((1-damping_psi^2)^0.5)*j;
continuous_pole_P = -damping_phi*wn_phi - wn_phi*((1-damping_phi^2)^0.5)*j;
continuous_pole_Q = -damping_theta*wn_theta - wn_theta*((1-damping_theta^2)^0.5)*j;
continuous_pole_R = -damping_psi*wn_psi - wn_psi*((1-damping_psi^2)^0.5)*j;
continuous_pole_X = -damping_X*wn_X + wn_X*((1-damping_X^2)^0.5)*j;
continuous_pole_Y = -damping_Y*wn_Y + wn_Y*((1-damping_Y^2)^0.5)*j;
continuous_pole_Z = -damping_Z*wn_Z + wn_Z*((1-damping_Z^2)^0.5)*j;
continuous_pole_U = -damping_X*wn_X - wn_X*((1-damping_X^2)^0.5)*j;
continuous_pole_V = -damping_Y*wn_Y - wn_Y*((1-damping_Y^2)^0.5)*j;
continuous_pole_W = -damping_Z*wn_Z - wn_Z*((1-damping_Z^2)^0.5)*j;

%New poles vector (continuous)
New_poles_continuous = [continuous_pole_X; continuous_pole_Y; continuous_pole_Z;
   continuous_pole_U; continuous_pole_V; continuous_pole_W;continuous_pole_phi;
   continuous_pole_theta; continuous_pole_psi; continuous_pole_P; continuous_pole_Q;
   continuous_pole_R;];

%New poles vector (discrete)
New_poles_discrete = exp(New_poles_continuous*Td);

K_DT = place(G,H,New_poles_discrete); %Discrete-Time Gain Matrix [4x12]
K_CT = place(A,B,New_poles_continuous); %Countinuous-Time Gain Matrix [4x12]
```

```
%_____________________CLOSED LOOP SYSTEM_____________________
%

iterations = 500; %Number of iterations (iterations*Td = total time of simulation)

%Initial State
clear X0

X0 = [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0;]; %Initial state vector

clear i

%Output Vector
Y_DT(:,1) = X0; %Setting the "beginning of time" as state X0

i = 1; %Counter Variable

OLD1 = X0;

R = [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0;]; %Desired State Reference

R_PWM = [1450; 1450; 1450; 1450;]; %PWM operational point

PWM(:,1) = R_PWM; %Setting PWM initial state as "PWM operational point"

Omega(:,1) = [445;445;445;445;]; %Setting initial state of input vector as "PWM operational
point"

Trajetoria(:,1) = [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0;];

while i < iterations


   %Defining input signal
   if i == 1
      OLD1 = X0;
   elseif i > 1
      OLD1 = NEXT1;
   end

   if (i == 20)
      R = R + [-0.25; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0;]; %u
   end

   if (i == 100)
      R = R + [0.25; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0;]; %u
   end

     if (i == 180)
      R = R + [0; 0.25; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0;]; %u
   end

   if (i == 260)
      R = R + [0; -0.25; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0;]; %u
   end

     if (i == 340)
```

```
        R = R + [0; 0; 0.1; 0; 0; 0; 0; 0; 0; 0; 0; 0;]; %u
    end

    if (i == 420)
        R = R + [0; 0; -0.1; 0; 0; 0; 0; 0; 0; 0; 0; 0;]; %u
    end

    Trajetoria(:,i+1) = R;

    %Input Signal
    U1 = round(K_DT*(R-OLD1),0);

    %Discrete Time Integraion Loop
    NEXT1 = G*OLD1 + H*U1; %State Space Equation for Closed Loop

    %Output Vector
    Y_DT(:,i+1) = NEXT1;
    PWM(:,i+1) = U1 + R_PWM;
    Omega(:,i+1) = 0.7217*PWM(:,i+1) - 603.76*ones(4,1);

    i = 1 + i;
end

%Difining simulation interval
time(1) = 0;
i = 2;
while(i < iterations + 1)
    time(i) = (i-1)*Td;
    i = i + 1;
end


%_____________________________STATE OBSERVER_____________________________
%

factor = 5;
Real_Observer = real(New_poles_discrete)/factor;
Imag_Observer = imag(New_poles_discrete)/factor;
Poles_Observer = Real_Observer + Imag_Observer*j;
K_ob = place(G,H,Poles_Observer); %Countinuous-Time Gain Matrix [4x12]


%_____________________________FIGURE PLOTS_____________________________
%

figure(1)
subplot(2,1,1);
stairs (time, Omega(1,:));
title('Rotacao - Motor 1', 'FontSize', 16);
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('rad/s', 'FontSize', 12);
grid on
grid minor
subplot(2,1,2);
stairs (time, Omega(2,:));
title('Rotacao - Motor 2', 'FontSize', 16);
xlabel('Tempo (s)', 'FontSize', 12);
```

```
ylabel('rad/s', 'FontSize', 12);
grid on
grid minor

figure(2)
subplot(2,1,1);
stairs (time, Omega(3,:));
title('Rotacao - Motor 3', 'FontSize', 16);
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('rad/s', 'FontSize', 12);
grid on
grid minor
subplot(2,1,2);
stairs (time, Omega(4,:));
title('Rotacao - Motor 4', 'FontSize', 16);
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('rad/s', 'FontSize', 12);
grid on
grid minor

figure(3)
subplot(2,1,1);
stairs (time, PWM(1,:));
title('Sinal PWM - Motor 1', 'FontSize', 16);
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('PWM', 'FontSize', 12);
grid on
grid minor
subplot(2,1,2);
stairs (time, PWM(2,:));
title('Sinal PWM - Motor 2', 'FontSize', 16);
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('PWM', 'FontSize', 12);
grid on
grid minor


figure(4)
subplot(2,1,1);
stairs (time, PWM(3,:));
title('Sinal PWM - Motor 3', 'FontSize', 16);
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('PWM', 'FontSize', 12);
grid on
grid minor
subplot(2,1,2);
stairs (time, PWM(4,:));
title('Sinal PWM - Motor 4', 'FontSize', 16);
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('PWM', 'FontSize', 12);
grid on
grid minor


figure(5)
subplot(2,1,1);
stairs (time, Y_DT(1,:));
hold on
```

```
stairs (time, Y_DT(2,:));
hold on
stairs (time, Y_DT(3,:));
title('Posicao Linear', 'FontSize', 16);
legend ('X', 'Y', 'Z');
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('(m)', 'FontSize', 12);
grid on
grid minor
subplot(2,1,2);
stairs (time, Y_DT(4,:));
hold on
stairs (time, Y_DT(5,:));
hold on
stairs (time, Y_DT(6,:));
title('Velocidade Linear', 'FontSize', 16);
legend ('U', 'V', 'W');
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('(m/s)', 'FontSize', 12);
grid on
grid minor

figure(6)
subplot(2,1,1);
stairs (time, Y_DT(7,:)*180/pi);
hold on
stairs (time, Y_DT(8,:)*180/pi);
hold on
stairs (time, Y_DT(9,:)*180/pi);
title('Posicao Angular', 'FontSize', 16);
legend ('Phi', 'Theta', 'Psi');
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('graus', 'FontSize', 12);
grid on
grid minor
subplot(2,1,2);
stairs (time, Y_DT(10,:));
hold on
stairs (time, Y_DT(11,:));
hold on
stairs (time, Y_DT(12,:));
title('Velocidade Angular', 'FontSize', 16);
legend ('P', 'Q', 'R');
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('rad/s', 'FontSize', 12);
grid on
grid minor


xhline = [-8; -7; -6; -5; -4; -3; -2; -1; 0; 1; 2;];
yhline = [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0;];

xvline = [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0;];
yvline = [5; 4; 3; 2; 1; 0; -1; -2; -3; -4; -5;];


figure(7)
plot (real(New_poles_continuous), imag(New_poles_continuous),'rx', xhline, yhline,'k', xvline,
```

```
yvline, 'k');
title('Polos do sistema em malha fechada - Tempo continuo', 'FontSize', 16);
xlabel('Eixo real', 'FontSize', 12);
ylabel('Eixo imaginario', 'FontSize', 12);
grid on
grid minor


th = 0:pi/50:2*pi;
xcirc = cos(th);
ycirc = sin(th);

xhline2 = [-1.5; -1; 0; 1; 1.5;];
yhline2 = [0; 0; 0; 0; 0;];

xvline2 = [0; 0; 0; 0; 0;];
yvline2 = [-1.5; -1; 0; 1; 1.5;];


figure(8)
plot (real(New_poles_discrete), imag(New_poles_discrete),'rx', xcirc, ycirc,'--b',xhline2,
yhline2,'k', xvline2, yvline2, 'k');
title('Polos do sistema em malha fechada - Tempo discreto', 'FontSize', 16);
xlabel('Eixo real', 'FontSize', 12);
ylabel('Eixo imaginario', 'FontSize', 12);
grid on
grid minor

figure(9)
subplot(3,1,1);
stairs (time, Y_DT(1,:));
hold on
stairs (time, Trajetoria(1,:));
xlim([0 40])
ylim([-0.35 0.35])
title('Posicao Linear X', 'FontSize', 16);
legend ('Posicao X', 'Trajetoria X');
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('(m)', 'FontSize', 12);
grid on
grid minor
subplot(3,1,2);
stairs (time, Y_DT(2,:));
hold on
stairs (time, Trajetoria(2,:));
xlim([30 70])
ylim([-0.1 0.6])
title('Posicao Linear Y', 'FontSize', 16);
legend ('Posicao Y', 'Trajetoria Y');
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('(m)', 'FontSize', 12);
grid on
grid minor
subplot(3,1,3);
stairs (time, Y_DT(3,:));
hold on
stairs (time, Trajetoria(3,:));
xlim([60 100])
```

```
ylim([-0.2 0.2])
title('Posicao Linear Z', 'FontSize', 16);
legend ('Posicao Z', 'Trajetoria Z');
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('(m)', 'FontSize', 12);
grid on
grid minor

figure(10)
subplot(2,1,1);
stairs (time, Omega(1,:));
hold on
stairs (time, Omega(2,:));
title('Rotacao - Motor 1 e Motor 2', 'FontSize', 16);
legend ('Motor 1', 'Motor 2');
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('rad/s', 'FontSize', 12);
grid on
grid minor
subplot(2,1,2);
stairs (time, Omega(3,:));
hold on
stairs (time, Omega(4,:));
title('Rotacao - Motor 3 e Motor 4', 'FontSize', 16);
legend ('Motor 3', 'Motor 4');
xlabel('Tempo (s)', 'FontSize', 12);
ylabel('rad/s', 'FontSize', 12);
grid on
grid minor

%_____________________TRANSFER FUNCTIONS_____________________
%

% Transfer Functions

syms s
Gs = C*(inv(s*eye(12)-A))*B+D;

[num_1,den_1] = ss2tf(A,B,C,D,1);
[num_2,den_2] = ss2tf(A,B,C,D,2);
[num_3,den_3] = ss2tf(A,B,C,D,3);
[num_4,den_4] = ss2tf(A,B,C,D,4);

%--- MOTOR 1 -----

FT_ay_4 = minreal(tf(num_4(5,:),den_4));
FT_P_4 = minreal(tf(num_4(7,:),den_4));

FT_z_1 = minreal(tf(num_1(3,:),den_1));
FT_ax_1 = minreal(tf(num_1(4,:),den_1));
FT_psi_1 = minreal(tf(num_1(6,:),den_1));
FT_Q_1 = minreal(tf(num_1(8,:),den_1));
FT_R_1 = minreal(tf(num_1(9,:),den_1));
FT_theta_1 = minreal(FT_ay_4/g + FT_P_4*tf([1 0],1)*rsz/g);

PID_z_1 = tf([1147.5508 2483.6034 441.6258], [1 0]);
H_z_1 = minreal((PID_z_1*FT_z_1)/(1 + PID_z_1*FT_z_1));
```

```
PID_ax_1 = tf([0 0 -1456957], [1 0]);
H_ax_1 = minreal((PID_ax_1*FT_ax_1)/(1 + PID_ax_1*FT_ax_1));

PID_psi_1 = tf([-735.6095 -1592.0535 -283.0935], [1 0]);
H_psi_1 = minreal((PID_psi_1*FT_psi_1)/(1 + PID_psi_1*FT_psi_1));

PID_Q_1 = tf([0 42.418 88.6904], [1 0]);
H_Q_1 = minreal((PID_Q_1*FT_Q_1)/(1 + PID_Q_1*FT_Q_1));

PID_R_1 = tf([0 667.416 1395.4786], [1 0]);
H_R_1 = minreal((PID_R_1*FT_R_1)/(1 + PID_R_1*FT_R_1));

PID_theta_1 = tf([46.7521 101.1838 17.9922], [1 0]);
H_theta_1 = minreal((PID_theta_1*FT_theta_1)/(1 + PID_theta_1*FT_theta_1));

%--- MOTOR 2 -----

FT_z_2 = minreal(tf(num_2(3,:),den_2));
FT_ay_2 = minreal(tf(num_2(5,:),den_2));
FT_psi_2 = minreal(tf(num_2(6,:),den_2));
FT_P_2 = minreal(tf(num_2(7,:),den_2));
FT_R_2 = minreal(tf(num_2(9,:),den_2));
FT_phi_2 = minreal(FT_ay_2/g + FT_P_2*tf([1 0],1)*rsz/g);

PID_z_2 = tf([1147.5508 2483.6034 441.6258], [1 0]);
H_z_2 = minreal((PID_z_2*FT_z_2)/(1 + PID_z_2*FT_z_2));

PID_ay_2 = tf([-6.0073 -13.4389 -7.5161], [1 0]);
H_ay_2 = minreal((PID_ay_2*FT_ay_2)/(1 + PID_ay_2*FT_ay_2));

PID_psi_2 = tf([735.6095 1592.0535 283.0935], [1 0]);
H_psi_2 = minreal((PID_psi_2*FT_psi_2)/(1 + PID_psi_2*FT_psi_2));

PID_P_2 = tf([0 -41.6438 -85.4824], [1 0]);
H_P_2 = minreal((PID_P_2*FT_P_2)/(1 + PID_P_2*FT_P_2));

PID_R_2 = tf([0 -667.416 -1395.4786], [1 0]);
H_R_2 = minreal((PID_R_2*FT_R_2)/(1 + PID_R_2*FT_R_2));

PID_phi_2 = tf([-46.7521 -101.1838 -17.9922], [1 0]);
H_phi_2 = minreal((PID_phi_2*FT_phi_2)/(1 + PID_phi_2*FT_phi_2));

%--- MOTOR 3 -----

FT_z_3 = minreal(tf(num_3(3,:),den_3));
FT_ax_3 = minreal(tf(num_3(4,:),den_3));
FT_psi_3 = minreal(tf(num_3(6,:),den_3));
FT_Q_3 = minreal(tf(num_3(8,:),den_3));
FT_R_3 = minreal(tf(num_3(9,:),den_3));
FT_theta_3 = -minreal(FT_ay_4/g + FT_P_4*tf([1 0],1)*rsz/g);

PID_z_3 = tf([1147.5508 2483.6034 441.6258], [1 0]);
H_z_3 = minreal((PID_z_3*FT_z_3)/(1 + PID_z_3*FT_z_3));

PID_ax_3 = tf([0 0 1456957], [1 0]);
H_ax_3 = minreal((PID_ax_3*FT_ax_3)/(1 + PID_ax_3*FT_ax_3));

PID_psi_3 = tf([-735.6095 -1592.0535 -283.0935], [1 0]);
```

```
H_psi_3 = minreal((PID_psi_3*FT_psi_3)/(1 + PID_psi_3*FT_psi_3));

PID_Q_3 = tf([0 -42.418 -88.6904], [1 0]);
H_Q_3 = minreal((PID_Q_3*FT_Q_3)/(1 + PID_Q_3*FT_Q_3));

PID_R_3 = tf([0 667.416 1395.4786], [1 0]);
H_R_3 = minreal((PID_R_3*FT_R_3)/(1 + PID_R_3*FT_R_3));

PID_theta_3 = tf([-46.7521 -101.1838 -17.9922], [1 0]);
H_theta_3 = minreal((PID_theta_3*FT_theta_3)/(1 + PID_theta_3*FT_theta_3));

%--- MOTOR 4 -----

FT_z_4 = minreal(tf(num_4(3,:),den_4));
FT_ay_4 = minreal(tf(num_4(5,:),den_4));
FT_psi_4 = minreal(tf(num_4(6,:),den_4));
FT_P_4 = minreal(tf(num_4(7,:),den_4));
FT_R_4 = minreal(tf(num_4(9,:),den_4));
FT_phi_4 = minreal(FT_ay_4/g + FT_P_4*tf([1 0],1)*rsz/g);

PID_z_4 = tf([1147.5508 2483.6034 441.6258], [1 0]);
H_z_4 = minreal((PID_z_4*FT_z_4)/(1 + PID_z_4*FT_z_4));

PID_ay_4 = tf([6.0073 13.4389 7.5161], [1 0]);
H_ay_4 = minreal((PID_ay_4*FT_ay_4)/(1 + PID_ay_4*FT_ay_4));

PID_psi_4 = tf([735.6095 1592.0535 283.0935], [1 0]);
H_psi_4 = minreal((PID_psi_4*FT_psi_4)/(1 + PID_psi_4*FT_psi_4));

PID_P_4 = tf([0 41.6438 85.4824], [1 0]);
H_P_4 = minreal((PID_P_4*FT_P_4)/(1 + PID_P_4*FT_P_4));

PID_R_4 = tf([0 -667.416 -1395.4786], [1 0]);
H_R_4 = minreal((PID_R_4*FT_R_4)/(1 + PID_R_4*FT_R_4));

PID_phi_4 = tf([46.7521 101.1838 17.9922], [1 0]);
H_phi_4 = minreal((PID_phi_4*FT_phi_4)/(1 + PID_phi_4*FT_phi_4));


%PID plots

j = 2;
dt_signal_z(1) = 0;
while(j < 41)
   dt_signal_z(j) = dt_signal_z(j-1) + Td;

   if (j-1 > 10)
   u_signal_z(j) = 0.05;
   else u_signal_z(j) = 0;
   end

   j = j + 1;
end

% Motor 1

lsim(H_z_1, u_signal_z, dt_signal_z)
```

```
[y_phi_1, x_phi_1] = step (H_ax_1);
[y_psi_1, x_psi_1] = step (H_psi_1);
[y_Q_1, x_Q_1] = step (H_Q_1);
[y_R_1, x_R_1] = step (H_R_1);

figure(11)
subplot(4,1,1);
plot(x_phi_1, y_phi_1);
title('FTs: Posicao e Velocidades Angulares - Motor 1', 'FontSize', 16);
legend ('Phi');
ylabel('Amplitude');
xlabel('Tempo (s)');
grid on
grid minor
subplot(4,1,2);
plot(x_psi_1, y_psi_1);
legend ('Psi');
ylabel('Amplitude');
xlabel('Tempo (s)');
grid on
grid minor
subplot(4,1,3);
plot(x_Q_1, y_Q_1);
legend ('Q');
ylabel('Amplitude');
xlabel('Tempo (s)');
grid on
grid minor
subplot(4,1,4);
plot(x_R_1, y_R_1);
legend ('R');
ylabel('Amplitude');
xlabel('Tempo (s)');
grid on
grid minor

% Motor 2

[y_theta_2, x_theta_2] = step (H_ay_2);
[y_psi_2, x_psi_2] = step (H_psi_2);
[y_P_2, x_P_2] = step (H_P_2);
[y_R_2, x_R_2] = step (H_R_2);

figure(12)
subplot(4,1,1);
plot(x_theta_2, y_theta_2);
title('FTs: Posicao e Velocidades Angulares - Motor 2', 'FontSize', 16);
legend ('Theta');
ylabel('Amplitude');
xlabel('Tempo (s)');
grid on
grid minor
subplot(4,1,2);
plot(x_psi_2, y_psi_2);
legend ('Psi');
ylabel('Amplitude');
xlabel('Tempo (s)');
grid on
```

```
grid minor
subplot(4,1,3);
plot(x_P_2, y_P_2);
legend ('P');
ylabel('Amplitude');
xlabel('Tempo (s)');
grid on
grid minor
subplot(4,1,4);
plot(x_R_2, y_R_2);
legend ('R');
ylabel('Amplitude');
xlabel('Tempo (s)');
grid on
grid minor

% Motor 3

[y_phi_3, x_phi_3] = step (H_ax_3);
[y_psi_3, x_psi_3] = step (H_psi_3);
[y_Q_3, x_Q_3] = step (H_Q_3);
[y_R_3, x_R_3] = step (H_R_3);

figure(13)
subplot(4,1,1);
plot(x_phi_3, y_phi_3);
title('FTs: Posicao e Velocidades Angulares - Motor 3', 'FontSize', 16);
legend ('Phi');
ylabel('Amplitude');
xlabel('Tempo (s)');
grid on
grid minor
subplot(4,1,2);
plot(x_psi_3, y_psi_3);
legend ('Psi');
ylabel('Amplitude');
xlabel('Tempo (s)');
grid on
grid minor
subplot(4,1,3);
plot(x_Q_3, y_Q_3);
legend ('Q');
ylabel('Amplitude');
xlabel('Tempo (s)');
grid on
grid minor
subplot(4,1,4);
plot(x_R_3, y_R_3);
legend ('R');
ylabel('Amplitude');
xlabel('Tempo (s)');
grid on
grid minor

% Motor 4

[y_theta_4, x_theta_4] = step (H_ay_4);
[y_psi_4, x_psi_4] = step (H_psi_4);
```

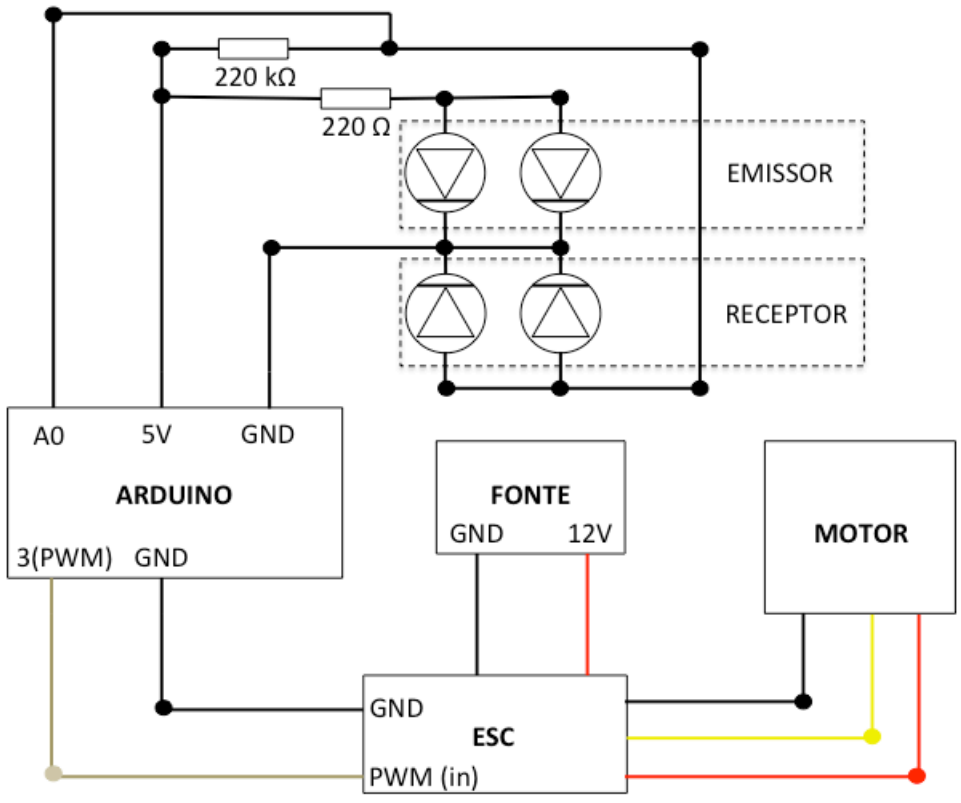
```
[y_P_4, x_P_4] = step (H_P_4);
[y_R_4, x_R_4] = step (H_R_4);

figure(14)
subplot(4,1,1);
plot(x_theta_4, y_theta_4);
title('FTs: Posicao e Velocidades Angulares - Motor 4', 'FontSize', 16);
legend ('Theta');
ylabel('Amplitude');
xlabel('Tempo (s)');
grid on
grid minor
subplot(4,1,2);
plot(x_psi_4, y_psi_4);
legend ('Psi');
ylabel('Amplitude');
xlabel('Tempo (s)');
grid on
grid minor
subplot(4,1,3);
plot(x_P_4, y_P_4);
legend ('P');
ylabel('Amplitude');
xlabel('Tempo (s)');
grid on
grid minor
subplot(4,1,4);
plot(x_R_4, y_R_4);
legend ('R');
ylabel('Amplitude');
xlabel('Tempo (s)');
grid on
grid minor

%_____________________CLASSICAL CONTROL THEORY_____________________
%

%PidTuner

PID_1 = tf([46.75 101.18 17.99], [1 0]);
H_1 = minreal((PID_1*FT_theta_1)/(1 + PID_1*FT_theta_1));


%Ziegler-Nichols

PID_2 = tf([16 162 410.13], [1 0]);
H_2 = minreal((PID_2*FT_theta_1)/(1 + PID_2*FT_theta_1));

%ITAE

PID_3 = tf([31.59 160.32 307.56], [1 0]);
H_3 = minreal((PID_3*FT_theta_1)/(1 + PID_3*FT_theta_1));

%Lugar das Ralzes

[NUM, DEN] = tfdata(FT_theta_1);
DEN = cell2mat(DEN);
NUM = cell2mat(NUM);
```

```
PID_4 = tf([48 120 270], [1 0]);
H_4 = minreal((PID_4*FT_theta_1)/(1 + PID_4*FT_theta_1));


%iteracao 1
K0P = 120;

%iteracao 2
AUX = minreal(tf(1,[1 0])*tf(NUM,DEN+K0P*NUM));

figure(15)
rlocus(AUX);
Zeta = 0.7;
Wn = 1.8;
sgrid(Zeta,Wn);

K0I = 270;

%iteracao 3
NUM_AUX = tf([1 0 0],1)*tf(NUM,1);
DEN_AUX = tf([1 0],1)*tf(DEN,1)+K0P*tf([1 0],1)*tf(NUM,1)+K0I*tf(NUM,1);
AUX = minreal(tf(NUM_AUX,DEN_AUX));

figure(16)
rlocus(AUX);
Zeta = 0.7;
Wn = 1.8;
sgrid(Zeta,Wn);

figure(17)
step(H_1);
hold on
step(H_2,':');
hold on
step(H_3);
hold on
step(H_4);
axis([0 4 0 1.8])
title('Resposta Degrau - ComparaÁ„o entre Controladores', 'FontSize', 16);
legend ('pidTuner', 'Ziegler-Nichols', 'ITAE', 'Root-Locus');
ylabel('Amplitude');
xlabel('Tempo (s)');
grid on
grid minor

figure (18)
bodeplot(H_1,H_2,H_3,H_4);
title('Analise no dominio da frequencia - Diagrama de Bode', 'FontSize', 16);
legend ('pidTuner', 'Ziegler-Nichols', 'ITAE', 'Root-Locus');
grid on
grid minor


NYQUIST1 = (1+ PID_1*FT_theta_1);
[NUM2, DEN2] = tfdata(NYQUIST1);
DEN2 = cell2mat(DEN2);
NUM2 = cell2mat(NUM2);
```

```
NYQUIST2 = tf(NUM2,1);

figure (19)
nyquist(H_1);
title('Diagrama de Nyquist - Controle por pidTuner', 'FontSize', 16);
legend ('pidTuner');
grid on
grid minor

figure (20)
nyquist(H_2, 'm');
title('Diagrama de Nyquist - Controle por Ziegler-Nichols', 'FontSize', 16);
legend ('Ziegler-Nichols');
grid on
grid minor

figure (21)
nyquist(H_3, 'c');
title('Diagrama de Nyquist - Controle por ITAE', 'FontSize', 16);
legend ('ITAE');
grid on
grid minor

figure (22)
nyquist(H_4, 'r');
title('Diagrama de Nyquist - Controle por Root-Locus', 'FontSize', 16);
legend ('Root-Locus');
grid on
grid minor
```

ANEXO C - CIRCUITO ELÉTRICO DO SISTEMA DE TESTE DE ROTAÇÃO

A seguir, é ilustrado o circuito elétrico projetado para realizar a medição da rotação do motor. A hélice fixada ao motor era identificada por meio da interrupção do sinal enviado do LED emissor para o LED receptor. Além do circuito elétrico ilustrado abaixo, também foram utilizados três rotinas de programação responsáveis por enviar sinais de comando para o motor (configuração da velocidade), leitura dos sinais lidos pelo receptor e calibração do ESC para uma determinada faixa de operação.



ANEXO D - CÓDIGO DO PROGRAMA DE TESTE DA ROTAÇÃO DO MOTOR

Ambiente de programação: Arduino

Código para controle da rotação e leitura do sinal enviado pelo receptor

```
#include <SoftwareSerial.h>

int pino_teste = 5;
int pinpin = 3;
int value;


//analog outputs:
int sensorPin = A0;
int sensorValue = 0; // variable to store the value coming from the sensor
int sensorTeste = A1;
int testeValue = 0;


void setup()
{
  Serial.begin(230400);
  delay(4000);
  analogWrite(pinpin, 1);
}


void loop()
{
//while (Serial.available() == 0);
//int val = Serial.parseInt(); //read int or parseFloat for ..float...
//Serial.println(val);
//analogWrite(pinpin, val);

  sensorValue = (analogRead (sensorPin));
  Serial.print(micros()); // print the time in milliseconds since the program
started
  Serial.print(';');
  Serial.println(sensorValue);// print to serial
```

```
}
```

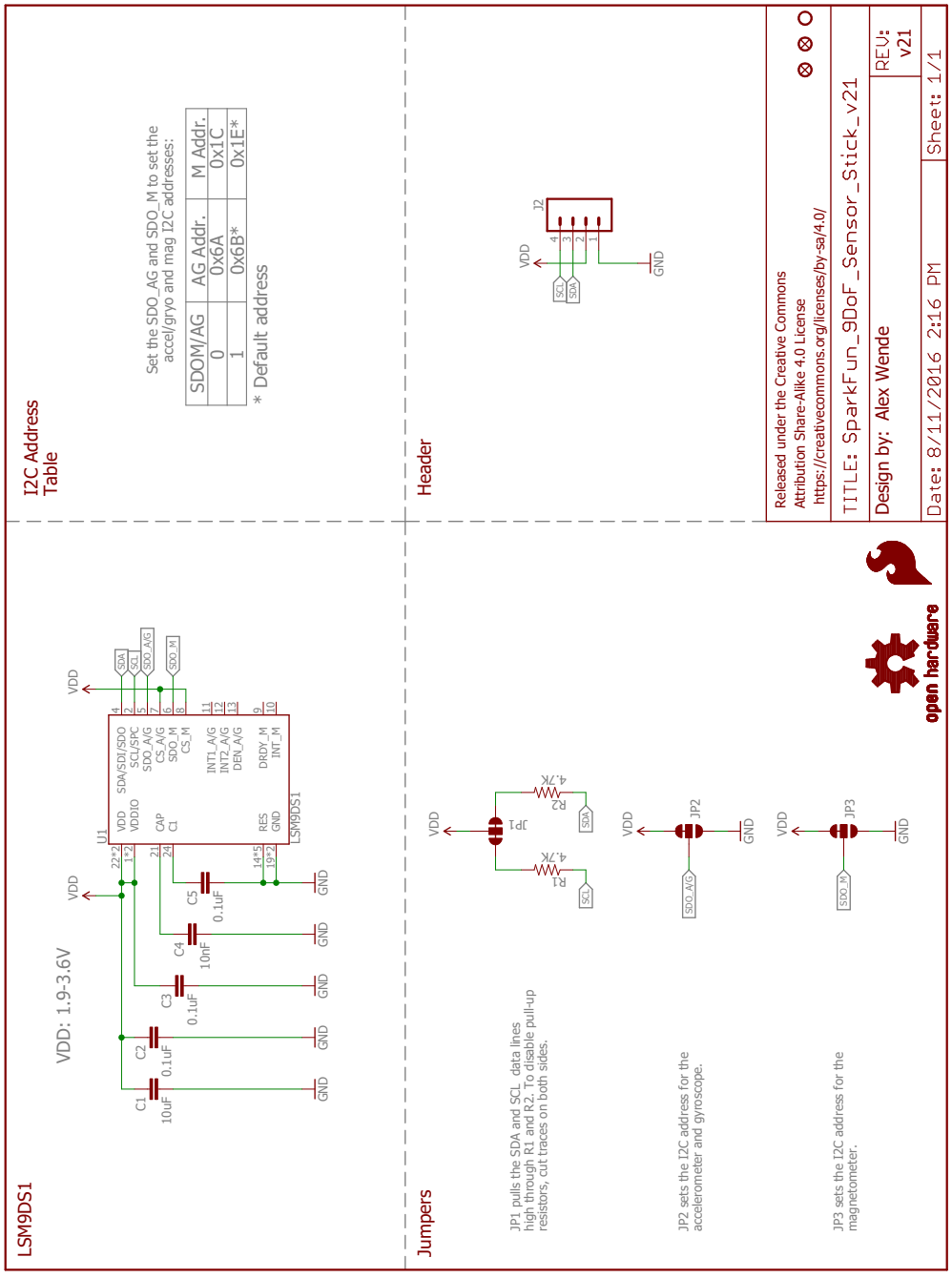
Código para calibrar o ESC

```
#include <SoftwareSerial.h>

int pino_teste = 5;
int pinpin = 3;
int value;


//analog outputs:
int sensorPin = A0; // select the input pin for the potentiometer
int sensorValue = 0; // variable to store the value coming from the sensor
int sensorTeste = A1;
int testeValue = 0;


void setup()
{
  Serial.begin(230400);
}


void loop()
{
while (Serial.available() == 0);
int val = Serial.parseInt(); //read int or parseFloat for ..float...
Serial.println(val);
analogWrite(pinpin, val);

  //sensorValue = (analogRead (sensorPin));
  //Serial.print(micros()); // print the time in milliseconds since the program
started
  //Serial.print(';');
  //Serial.println(sensorValue);// print to serial
}
```

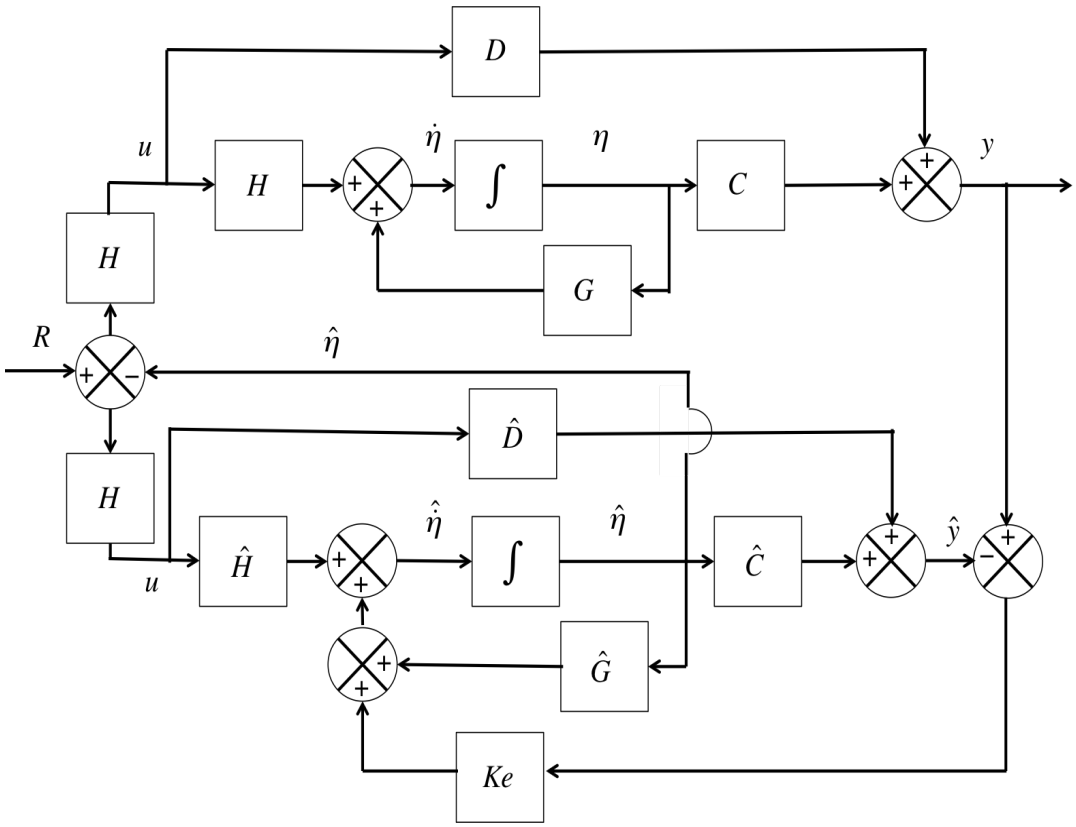
Código para leitura da largura de pulso (PWM)

```
#include <SoftwareSerial.h>

byte pinpin = 5;
int pwm_value;

void setup()
{
  pinMode(pinpin, INPUT);
  Serial.begin(230400);
  delay(4000);
  analogWrite(pinpin, 220);


}


void loop()
{
  pwm_value = pulseIn(pinpin, HIGH);
  Serial.println(pwm_value);

}
```

ANEXO E – CONFIGURAÇÃO DO SENSOR SPARKFUN 9DOF



ANEXO F – DIAGRAMA DE BLOCOS DO OBSERVADOR



ANEXO G – CÓDIGO ARDUINO: CONTROLE MODERNO

```
/* ============================================
I2Cdev device library code is placed under the MIT license
Copyright (c) 2012 Jeff Rowberg

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
OTHER DEALINGS IN THE SOFTWARE.
===============================================
*/

// I2Cdev and MPU6050 must be installed as libraries, or else the .cpp/.h files
// for both classes must be in the include path of your project
#include "I2Cdev.h"
#include <Servo.h>
#include <SPI.h>

#include "MPU6050_6Axis_MotionApps20.h"

// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE
implementation
// is used in I2Cdev.h
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  #include "Wire.h"
#endif

// class default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// AD0 low = 0x68 (default for SparkFun breakout and InvenSense evaluation
board)
```

```
// AD0 high = 0x69
MPU6050 mpu;
//MPU6050 mpu(0x69); // <-- use for AD0 high

#define OUTPUT_READABLE_YAWPITCHROLL
#define OUTPUT_READABLE_WORLDACCEL

//Some definitions in order to use in the steup() and loop()

#define PPM0 1540

#define MOTOR1      5  //Pin that sends the command to the actuator 1
//#define MOTOR2      6  //Pin that sends the command to the actuator 2
//#define MOTOR3      9  //Pin that sends the command to the actuator 3
#define MOTOR4      10  //Pin that sends the command to the actuator 4

#define INTERVALO   1

#define wM   writeMicroseconds

Servo ESC[4];

#define LED_PIN 13 // (Arduino is 13, Teensy is 11, Teensy++ is 6)
bool blinkState = false;

// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 = success, !0
= error)
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer
uint16_t tempo;

// orientation/motion vars
Quaternion q;           // [w, x, y, z]        quaternion container
VectorInt16 aa;         // [x, y, z]           accel sensor measurements
VectorInt16 aaReal;     // [x, y, z]           gravity-free accel sensor measurements
VectorInt16 aaWorld;    // [x, y, z]           world-frame accel sensor
measurements
VectorFloat gravity;    // [x, y, z]           gravity vector
float euler[3];         // [psi, theta, phi]    Euler angle container
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity
vector

// packet structure for InvenSense teapot demo
```

```
uint8_t teapotPacket[14] = { '$', 0x02, 0,0, 0,0, 0,0, 0,0, 0x00, 0x00, '\r', '\n' };

//
============================================================
===
// ===            INTERRUPT DETECTION ROUTINE            ===
//
============================================================
===

volatile bool mpuInterrupt = false;     // indicates whether MPU interrupt pin
has gone high
void dmpDataReady() {
   mpuInterrupt = true;
}

//
============================================================
===
// ===                  INITIAL SETUP                  ===
//
============================================================
===

void setup() {

tempo = millis();

 ESC[0].attach(MOTOR1);
 //ESC[1].attach(MOTOR2);
 //ESC[2].attach(MOTOR3);
 ESC[3].attach(MOTOR4);

 ESC[0].writeMicroseconds(2000);
 //ESC[1].writeMicroseconds(2000);
 //ESC[2].writeMicroseconds(2000);
 ESC[3].writeMicroseconds(2000);
 delay(2000);

 ESC[0].writeMicroseconds(1000);
 //ESC[1].writeMicroseconds(1000);
 //ESC[2].writeMicroseconds(1000);
 ESC[3].writeMicroseconds(1000);
 delay(2000);

   // join I2C bus (I2Cdev library doesn't do this automatically)
   #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
```

```
    Wire.begin();
    TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
  #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
    Fastwire::setup(400, true);
  #endif

  // initialize serial communication
  // (115200 chosen because it is required for Teapot Demo output, but it's
  // really up to you depending on your project)
  Serial.begin(115200);
  while (!Serial); // wait for Leonardo enumeration, others continue
immediately

  // NOTE: 8MHz or slower host processors, like the Teensy @ 3.3v or Ardunio
  // Pro Mini running at 3.3v, cannot handle this baud rate reliably due to
  // the baud timing being too misaligned with processor ticks. You must use
  // 38400 or slower in these cases, or use some kind of external separate
  // crystal solution for the UART timer.

  // initialize device
  Serial.println(F("Initializing I2C devices..."));
  mpu.initialize();

  // verify connection
  Serial.println(F("Testing device connections..."));
  Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") :
F("MPU6050 connection failed"));

  // wait for ready
  //Serial.println(F("\nSend any character to begin DMP programming and
demo: "));
  //while (Serial.available() && Serial.read()); // empty buffer
  //while (!Serial.available());                 // wait for data
  //while (Serial.available() && Serial.read()); // empty buffer again

  // load and configure the DMP
  Serial.println(F("Initializing DMP..."));
  devStatus = mpu.dmpInitialize();

  // supply your own gyro offsets here, scaled for min sensitivity
  mpu.setXGyroOffset(220);
  mpu.setYGyroOffset(76);
  mpu.setZGyroOffset(-85);
  mpu.setZAccelOffset(1788); // 1688 factory default for my test chip

  // make sure it worked (returns 0 if so)
  if (devStatus == 0) {
```

```
        // turn on the DMP, now that it's ready
        Serial.println(F("Enabling DMP..."));
        mpu.setDMPEnabled(true);

        // enable Arduino interrupt detection
        Serial.println(F("Enabling interrupt detection (Arduino external interrupt
0)..."));
        attachInterrupt(0, dmpDataReady, RISING);
        mpuIntStatus = mpu.getIntStatus();

        // set our DMP Ready flag so the main loop() function knows it's okay to
use it
        Serial.println(F("DMP ready! Waiting for first interrupt..."));
        dmpReady = true;

        // get expected DMP packet size for later comparison
        packetSize = mpu.dmpGetFIFOPacketSize();
    } else {
        // ERROR!
        // 1 = initial memory load failed
        // 2 = DMP configuration updates failed
        // (if it's going to break, usually the code will be 1)
        Serial.print(F("DMP Initialization failed (code "));
        Serial.print(devStatus);
        Serial.println(F(")"));
    }

    // configure LED for output
    pinMode(LED_PIN, OUTPUT);
}

//
============================================================
===
// ===               MAIN PROGRAM LOOP               ===
//
============================================================
===

void loop() {


    // reset interrupt flag and get INT_STATUS byte
    mpuInterrupt = false;
    mpuIntStatus = mpu.getIntStatus();

    // get current FIFO count
```

```
    fifoCount = mpu.getFIFOCount();

    // check for overflow (this should never happen unless our code is too
inefficient)
    if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
        // reset so we can continue cleanly
        mpu.resetFIFO();
        Serial.println(F("FIFO overflow!"));

    // otherwise, check for DMP data ready interrupt (this should happen
frequently)
    } else if (mpuIntStatus & 0x02) {
        // wait for correct available data length, should be a VERY short wait
        while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();

        // read a packet from FIFO
        mpu.getFIFOBytes(fifoBuffer, packetSize);

        // track FIFO count here in case there is > 1 packet available
        // (this lets us immediately read more without waiting for an interrupt)
        fifoCount -= packetSize;

        #ifdef OUTPUT_READABLE_YAWPITCHROLL
            // display Euler angles in degrees
            mpu.dmpGetQuaternion(&q, fifoBuffer);
            mpu.dmpGetGravity(&gravity, &q);
            mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
        #endif

        #ifdef OUTPUT_READABLE_WORLDACCEL
            // display initial world-frame acceleration, adjusted to remove gravity
            // and rotated based on known orientation from quaternion
            mpu.dmpGetQuaternion(&q, fifoBuffer);
            mpu.dmpGetAccel(&aa, fifoBuffer);
            mpu.dmpGetGravity(&gravity, &q);
            mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
            mpu.dmpGetLinearAccelInWorld(&aaWorld, &aaReal, &q);
        #endif

        // blink LED to indicate activity
        blinkState = !blinkState;
        digitalWrite(LED_PIN, blinkState);

if(millis()>30000){
if(millis() - tempo > INTERVALO){

     //Gain Matrix
```

```
//K=[8.5 -0.34 166.36 9.60 -0.19 99.49 0.60 48.42 98.97 0.07 13.53 63.09;
//   0.10 8.42 160.33 0.08 9.59 98.59 -48.38 0.29 -107.76 -13.50 0.05 -66.79;
//  -8.29 -0.36 166.36 -9.44 -0.21 99.49 0.64 -47.97 98.97 0.08 -13.48 63.09;
//   0.08 -8.37 160.33 0.07 -9.45 98.59 48.00 0.24 -107.76 13.51 0.05 -66.79;]

// R = [X,Y,Z,U,V,W,phi,theta,psi,p,q,r]
// R = [0,0,0.2,0,0,0,0,0,0,0,0,0];

//Variables uptade

x_var = 0.5*aaWorld.x*(millis()-tempo)^2;
y_var = 0.5*aaWorld.y*(millis()-tempo)^2;
z_var = 0.5*aaWorld.z*(millis()-tempo)^2;
u_var = aaWorld.x*(millis()-tempo);
v_var = aaWorld.y*(millis()-tempo);
w_var = aaWorld.z*(millis()-tempo);

tempo = millis();

// Control Processe for Actuator 1
PPM01 = 8.5*(0-x_var);
PPM01 = PPM01 - 0.34*(0-y_var);
PPM01 = PPM01 + 166.36*(0-z_var);
PPM01 = PPM01 + 9.60*(0-u_var);
PPM01 = PPM01 - 0.19*(0-v_var);
PPM01 = PPM01 + 99.49*(0-w_var);
PPM01 = PPM01 + 0.60*(0-ypr[1]);
PPM01 = PPM01 + 48.42*(0-ypr[2]);
PPM01 = PPM01 + 98.97*(0-ypr[0]);
PPM01 = PPM01 + 0.07*(0-p_var);
PPM01 = PPM01 + 13.53*(0-q_var);
PPM01 = PPM01 + 63.09*(0-r_var);

// Control Processe for Actuator 2
PPM02 = 0.1*(0-x_var);
PPM02 = PPM02 - 8.42*(0-y_var);
PPM02 = PPM02 + 160.33*(0-z_var);
PPM02 = PPM02 + 0.08*(0-u_var);
PPM02 = PPM02 - 9.59*(0-v_var);
PPM02 = PPM02 + 98.59*(0-w_var);
PPM02 = PPM02 - 48.38*(0-ypr[1]);
PPM02 = PPM02 + 0.29*(0-ypr[2]);
PPM02 = PPM02 - 107.76*(0-ypr[0]);
PPM02 = PPM02 - 13.5*(0-p_var);
PPM02 = PPM02 + 0.05*(0-q_var);
PPM02 = PPM02 -66.79*(0-r_var);
```

```
 // Control Processe for Actuator 3
 PPM03 = -8.29*(0-x_var);
 PPM03 = PPM03 - 0.36*(0-y_var);
 PPM03 = PPM03 + 166.36*(0-z_var);
 PPM03 = PPM03 -9.44*(0-u_var);
 PPM03 = PPM03 – 0.21*(0-v_var);
 PPM03 = PPM03 + 99.49*(0-w_var);
 PPM03 = PPM03 + 0.64*(0-ypr[1]);
 PPM03 = PPM03 – 47.97*(0-ypr[2]);
 PPM03 = PPM03 + 98.97*(0-ypr[0]);
 PPM03 = PPM03 + 0.08*(0-p_var);
 PPM03 = PPM03 - 13.48*(0-q_var);
 PPM03 = PPM03 + 63.09*(0-r_var);

 // Control Processe for Actuator 4
 PPM04 = 0.08*(0-x_var);
 PPM04 = PPM04 - 8.37*(0-y_var);
 PPM04 = PPM04 + 160.33*(0-z_var);
 PPM04 = PPM04 + 0.07*(0-u_var);
 PPM04 = PPM04 - 9.45*(0-v_var);
 PPM04 = PPM04 + 98.59*(0-w_var);
 PPM04 = PPM04 + 48.00*(0-ypr[1]);
 PPM04 = PPM04 + 0.24*(0-ypr[2]);
 PPM04 = PPM04 - 107.76*(0-ypr[0]);
 PPM04 = PPM04 + 13.51*(0-p_var);
 PPM04 = PPM04 + 0.05*(0-q_var);
 PPM04 = PPM04 -66.79*(0-r_var);

 // Control Output

 ESC[0].wM(PPM01);
 //ESC[1].wM(PPM02);
 //ESC[2].wM(PPM03);
 ESC[3].wM(PPM04); }

//else{

//ESC[0].wM(0);
//ESC[1].wM(0);
//ESC[2].wM(0);
//ESC[3].wM(0);
}}}}
```

ANEXO H – CÓDIGO ARDUINO: CONTROLE CLÁSSICO

```
/* ============================================
I2Cdev device library code is placed under the MIT license
Copyright (c) 2012 Jeff Rowberg

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
OTHER DEALINGS IN THE SOFTWARE.
===============================================
*/

// I2Cdev and MPU6050 must be installed as libraries, or else the .cpp/.h files
// for both classes must be in the include path of your project
#include "I2Cdev.h"
#include <PID.h>
#include <Servo.h>
#include <SPI.h>

#include "MPU6050_6Axis_MotionApps20.h"

// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE
implementation
// is used in I2Cdev.h
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  #include "Wire.h"
#endif

// class default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
```

```
// AD0 low = 0x68 (default for SparkFun breakout and InvenSense evaluation
board)
// AD0 high = 0x69
MPU6050 mpu;
//MPU6050 mpu(0x69); // <-- use for AD0 high


#define OUTPUT_READABLE_YAWPITCHROLL
#define OUTPUT_READABLE_ACCELGYRO

//Some definitions in order to use in the steup() and loop()

#define PPM0 1550

#define MOTOR1      5  //Pin that sends the command to the actuator 1
#define MOTOR2      6  //Pin that sends the command to the actuator 2
#define MOTOR3      9  //Pin that sends the command to the actuator 3
#define MOTOR4      10  //Pin that sends the command to the actuator 4

#define INTERVALO   1

#define wM   writeMicroseconds

Servo ESC[4];
//170, 450, 80
//myPid PID1(175, 250, 30);         //Actuator 1 to angle "Phi"
myPid PID2(160, 200, 30);       //Actuator 2 to angle "Theta"
myPid PID3(-160, -200, -30);         //Actuator 1 to angle "Phi"
//myPid PID4(-175, -250, -30);       //Actuator 2 to angle "Theta"

#define LED_PIN 13 // (Arduino is 13, Teensy is 11, Teensy++ is 6)
bool blinkState = false;

// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 = success, !0
= error)
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

uint16_t tempo;
uint16_t tempo_gyro;
uint16_t tempo_gyro_2;
uint16_t tempo_gyro_1;
```

```
// orientation/motion vars
Quaternion q;           // [w, x, y, z]        quaternion container
VectorInt16 aa;         // [x, y, z]           accel sensor measurements
VectorInt16 aaReal;     // [x, y, z]           gravity-free accel sensor measurements
VectorInt16 aaWorld;    // [x, y, z]           world-frame accel sensor
measurements
VectorFloat gravity;    // [x, y, z]           gravity vector
float euler[3];         // [psi, theta, phi]    Euler angle container
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity
vector
int16_t gyros[3];       // [p, q, r]   yaw/pitch/roll angular velocity

float g_roll;


// packet structure for InvenSense teapot demo
uint8_t teapotPacket[14] = { '$', 0x02, 0,0, 0,0, 0,0, 0,0, 0x00, 0x00, '\r', '\n' };

//
============================================================
===
// ===            INTERRUPT DETECTION ROUTINE            ===
//
============================================================
===

volatile bool mpuInterrupt = false;     // indicates whether MPU interrupt pin
has gone high
void dmpDataReady() {
   mpuInterrupt = true;
}

//
============================================================
===
// ===                  INITIAL SETUP                  ===
//
============================================================
===

void setup() {
tempo_gyro_1 = millis();
tempo_gyro_2 = millis();
tempo_gyro = millis();
g_roll = 0;
```

```
tempo = millis();

  //ESC[0].attach(MOTOR1);
  ESC[1].attach(MOTOR2);
  ESC[2].attach(MOTOR3);
  //ESC[3].attach(MOTOR4);

  ESC[0].writeMicroseconds(2000);
  ESC[1].writeMicroseconds(2000);
  ESC[2].writeMicroseconds(2000);
  ESC[3].writeMicroseconds(2000);
  delay(2000);

  ESC[0].writeMicroseconds(1000);
  ESC[1].writeMicroseconds(1000);
  ESC[2].writeMicroseconds(1000);
  ESC[3].writeMicroseconds(1000);
  delay(2000);

    // join I2C bus (I2Cdev library doesn't do this automatically)
    #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
       Wire.begin();
       TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
    #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
       Fastwire::setup(400, true);
    #endif

// initialize serial communication
    // (115200 chosen because it is required for Teapot Demo output, but it's
    // really up to you depending on your project)
    Serial.begin(115200);
    while (!Serial); // wait for Leonardo enumeration, others continue
immediately

    // NOTE: 8MHz or slower host processors, like the Teensy @ 3.3v or Ardunio
    // Pro Mini running at 3.3v, cannot handle this baud rate reliably due to
    // the baud timing being too misaligned with processor ticks. You must use
    // 38400 or slower in these cases, or use some kind of external separate
    // crystal solution for the UART timer.

    // initialize device
    Serial.println(F("Initializing I2C devices..."));
    mpu.initialize();

    // verify connection
    Serial.println(F("Testing device connections..."));
```

```
  Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") :
F("MPU6050 connection failed"));

  // wait for ready
  //Serial.println(F("\nSend any character to begin DMP programming and
demo: "));
  //while (Serial.available() && Serial.read()); // empty buffer
  //while (!Serial.available());                 // wait for data
  //while (Serial.available() && Serial.read()); // empty buffer again

  // load and configure the DMP
  Serial.println(F("Initializing DMP..."));
  devStatus = mpu.dmpInitialize();

  // supply your own gyro offsets here, scaled for min sensitivity
  mpu.setXGyroOffset(220);
  mpu.setYGyroOffset(76);
  mpu.setZGyroOffset(-85);
  mpu.setZAccelOffset(1788); // 1688 factory default for my test chip

  // make sure it worked (returns 0 if so)
  if (devStatus == 0) {
    // turn on the DMP, now that it's ready
    Serial.println(F("Enabling DMP..."));
    mpu.setDMPEnabled(true);

    // enable Arduino interrupt detection
    Serial.println(F("Enabling interrupt detection (Arduino external interrupt
0)..."));
    attachInterrupt(0, dmpDataReady, RISING);
    mpuIntStatus = mpu.getIntStatus();

    // set our DMP Ready flag so the main loop() function knows it's okay to
use it
    Serial.println(F("DMP ready! Waiting for first interrupt..."));
    dmpReady = true;

    // get expected DMP packet size for later comparison
    packetSize = mpu.dmpGetFIFOPacketSize();
  } else {
    // ERROR!
    // 1 = initial memory load failed
    // 2 = DMP configuration updates failed
    // (if it's going to break, usually the code will be 1)
    Serial.print(F("DMP Initialization failed (code "));
    Serial.print(devStatus);
    Serial.println(F(")"));
```

```
    }

    // configure LED for output
    pinMode(LED_PIN, OUTPUT);
}



// ================================================================
// ===                MAIN PROGRAM LOOP                ===
// ================================================================

void loop() {

  float gvx;
  float gvy;
  float gvz;
  int16_t dtempo;
  float g_pitch;
  float g_yaw;
  int16_t dtempo_2;
  int16_t dtempo_1;

    // if programming failed, don't try to do anything
    if (!dmpReady) return;

    // reset interrupt flag and get INT_STATUS byte
    mpuInterrupt = false;
    mpuIntStatus = mpu.getIntStatus();

    // get current FIFO count
    fifoCount = mpu.getFIFOCount();

    // check for overflow (this should never happen unless our code is too
inefficient)
    if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
        // reset so we can continue cleanly
        mpu.resetFIFO();
        Serial.println(F("FIFO overflow!"));

    // otherwise, check for DMP data ready interrupt (this should happen
frequently)
    } else if (mpuIntStatus & 0x02) {
```

```
    // wait for correct available data length, should be a VERY short wait
    while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();

    // read a packet from FIFO
    mpu.getFIFOBytes(fifoBuffer, packetSize);

    // track FIFO count here in case there is > 1 packet available
    // (this lets us immediately read more without waiting for an interrupt)
    fifoCount -= packetSize;



#ifdef OUTPUT_READABLE_ACCELGYRO
    // display tab-separated gyro values

        mpu.dmpGetGyro(gyros, fifoBuffer);
        gvx = (float)gyros[0] / 131.0f;
        gvy = (float)gyros[1] / 131.0f;
        gvz = (float)gyros[2] / 131.0f;
        g_pitch = g_pitch + gvx*dtempo;
        g_roll = g_roll + gvy*dtempo;
        g_yaw = g_yaw + gvz*dtempo;
        dtempo = (millis()-tempo_gyro);
        tempo_gyro = millis();

#endif


    #ifdef OUTPUT_READABLE_YAWPITCHROLL
        // display Euler angles in degrees
        mpu.dmpGetQuaternion(&q, fifoBuffer);
        mpu.dmpGetGravity(&gravity, &q);
        mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
    #endif

    #ifdef OUTPUT_READABLE_WORLDACCEL
        // display initial world-frame acceleration, adjusted to remove gravity
        // and rotated based on known orientation from quaternion
        mpu.dmpGetQuaternion(&q, fifoBuffer);
        mpu.dmpGetAccel(&aa, fifoBuffer);
        mpu.dmpGetGravity(&gravity, &q);
        mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
        mpu.dmpGetLinearAccelInWorld(&aaWorld, &aaReal, &q);
    #endif

    // blink LED to indicate activity
    blinkState = !blinkState;
```

```
      digitalWrite(LED_PIN, blinkState);

if(millis()>30000){
if(millis() - tempo > INTERVALO){

    tempo = millis();

    // Control Processe for Actuator 1
    //PID1.addNewSample(ypr[2]+0.02);
    //int controlePPM01 = (PID1.process() + PPM0);

    // Control Processe for Actuator 2
    PID2.addNewSample(ypr[1]-0.024127);
    int controlePPM02 = (PID2.process() + PPM0);
    if(controlePPM02 > 1990){controlePPM02 = PPM0;}
    if(controlePPM02 < 1100){controlePPM02 = PPM0;}

    // Control Processe for Actuator 3
    PID3.addNewSample(ypr[1]-0.024127);
    int controlePPM03 = (PID3.process() + PPM0);
    if(controlePPM03 > 1990){controlePPM03 = 1990;}
    if(controlePPM03 < 1100){controlePPM03 = 1110;}

    // Control Processe for Actuator 4
    //PID4.addNewSample(ypr[2]+0.02);
    //int controlePPM04 = (PID4.process() + PPM0);

    // Control Output

    //ESC[0].wM(controlePPM01);
    ESC[1].wM(controlePPM02);
    ESC[2].wM(controlePPM03);
    //ESC[3].wM(controlePPM04);

  }

  //else{

  //ESC[0].wM(0);
  //ESC[1].wM(0);
  //ESC[2].wM(0);
  //ESC[3].wM(0);

 //}
  }
  }
  }
```

ANEXO I – CRONOGRAMA DE ATIVIDADES

Atividade proposta	2º Sem. 2016				1º Sem. 2017						2º Sem. 2017				
	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N
Revisão Bibliográfica	x	x	x	x	x										
Redação do relatório parcial de PME2598	x	x													
Obtenção do modelo dinâmico não linear do sistema	x														
Obtenção do modelo dinâmico linear do sistema		x													
Implementação do modelo em ambiente de programação			x												
Obtenção do modelo em espaço de estado			x												
Redação do relatório final de PME2598			x	x											
Seleção de atuadores e Sensores					x	x	x	x	x	x					
Projeto do protótipo			x	x	x	x	x	x	x	x	x	x	x	x	
Construção do protótipo					x	x	x	x	x	x	x	x			
Relatório parcial de PME2599					x	x	x								
Medição dos parâmetros físicos do protótipo							x	x	x						
Implementação dos parâmetros no sistema de controle								x	x						
Síntese do controlador (Controle Moderno)							x	x	x						
Redação do relatório final e do artigo técnico de PME2599								x	x	x					
Programação do algoritmo de controle utilizando Arduino											x	x	x		
Síntese do controlador (Controle Clássico)											x	x	x		
Redação do relatório parcial de PME2600											x	x			
Implementação do sistema de controle no protótipo											x	x	x	x	x
Análise e ajuste do algoritmo de controle											x	x	x	x	x
Redação do relatório final e do artigo técnico de PME2600													x	x	x
Preparação e apresentação do pôster de PME2600														x	x