UNIVERSIDADE FEDERAL DE JUIZ DE FORA
FACULDADE DE ENGENHARIA
ENGENHARIA ELÉTRICA

SÉRGIO BARBOSA NEVES JÚNIOR

METODOLOGIA DE DESENVOLVIMENTO DE UM VEÍCULO AÉREO NÃO TRIPULADO: PROJETO, SIMULAÇÃO E TESTES

Juiz de Fora
2016

SÉRGIO BARBOSA NEVES JÚNIOR

METODOLOGIA DE DESENVOLVIMENTO DE UM VEÍCULO AÉREO NÃO TRIPULADO: PROJETO, SIMULAÇÃO E TESTES

Trabalho de conclusão de curso apresentado ao Curso de Graduação em Engenharia Elétrica da Faculdade de Engenharia Elétrica da Universidade Federal de Juiz de Fora, como requisito parcial para a obtenção do Grau de Engenheiro Eletricista com ênfase em Robótica e Automação Industrial

Orientadora: Prof[a]. Ph.D. Janaína Gonçalves de Oliveira

Juiz de Fora

2016

Sérgio Barbosa Neves Júnior

METODOLOGIA DE DESENVOLVIMENTO DE UM VEÍCULO AÉREO NÃO TRIPULADO: PROJETO, SIMULAÇÃO E TESTES

Trabalho de conclusão de curso apresentado ao Curso de Graduação em Engenharia Elétrica da Faculdade de Engenharia Elétrica da Universidade Federal de Juiz de Fora, como requisito parcial para a obtenção do Grau de Engenheiro Eletricista com ênfase em Robótica e Automação Industrial

Aprovado em _____ de _____ de 2016

BANCA EXAMINADORA:

_____

Orientadora: Profª. Ph.D. Janaína Gonçalves de Oliveira
Universidade Federal de Juiz de Fora, UFJF

_____

Prof. D.Sc. Danilo Pereira Pinto
Universidade Federal de Juiz de Fora, UFJF

_____

Prof. D.Sc. Leonardo Rocha Olivi
Universidade Federal de Juiz de Fora, UFJF

Dedicatória

Aos meus pais, Sérgio e Mariclara,
Aos meus avós, Eny e Justiniano,
À minha irmã, Clara,
À minha companheira, Aline.

AGRADECIMENTOS

À professora Janaína Gonçalves de Oliveira, por toda sua dedicação e orientação ao longo destes poucos, porém importantes meses de trabalho.

Aos meus pais Sérgio e Mariclara, por todo amor incondicional e por serem meus maiores guias durante toda minha vida. À minha irmã Clara, por toda a ajuda, e por ser meu maior modelo. A todos meus familiares, por todo o carinho e suporte.

À minha companheira Aline, pelo seu amor e por acreditar tanto em mim durante todos os momentos, e à toda sua família, por toda sua hospitalidade e carinho por todos estes anos.

A todos meus grandes amigos e professores da Faculdade de Engenharia da UFJF, que tiveram grande contribuição para a construção deste trabalho e minha formação.

Aos amigos e professores da Brunel University of London, os quais tiveram grande influência sobre minha formação acadêmica, em especial ao professor Roger Powell, cujo projeto serviu como inspiração a este trabalho.

Ao professor Francisco José Gomes e a todos os membos do PET Elétrica, por toda sua contribuição e auxílio na elaboração deste trabalho, especialmente aos membros da equipe do projeto de construção do *drone*.

RESUMO

METODOLOGIA DE DESENVOLVIMENTO DE UM VEÍCULO AÉREO NÃO TRIPULADO: PROJETO, SIMULAÇÃO E TESTES

Sérgio Barbosa Neves Júnior
Novembro/2016

Orientadora: Prof[a]. Ph.D. Janaína Gonçalves de Oliveira

Este trabalho propõe a construção de um Veículo Aéreo Não Tripulado quadrimotor de baixo custo, fácil replicação e capacidade de autonomia. O protótipo foi construído e programado baseando-se em plataformas open-source, entre elas a Ardupilot, empregada no controlador de voo, e a RepRap3D, utilizada na impressão 3D das peças projetadas para a estrutura do drone. Visando a melhoria da estabilidade de voo do protótipo, foram validadas duas técnicas de sintonia de controlador PID: Twiddle e ITAE, realizando simulações no software Mission Planner. Além disso, através do mesmo software, foi validado o método Campos Potenciais de planejamento de trajetória, construído neste trabalho em algoritmo Python, buscando a automatização do veículo.

Palavras chave: Robótica móvel, Método campos potenciais, PID, Ardupilot, RepRap3d, Mission Planner.

ABSTRACT

DEVELOPMENT METHODOLOGY OF NA UNMANNED AERIAL VEHICLE: PROJECT, SIMULATION AND TESTS

Sérgio Barbosa Neves Júnior

November/2016

Advisor: Janaína Gonçalves de Oliveira

This work proposes the construction of a quadrotor Unmanned Aerial Vehicle with low budget, easy replicate and autonomous capability. The prototype was constructed and programed based on open-source platforms, among them Ardupilot, employed at the flight controller, and RepRap3D, used in the 3D printing of the parts designed for the drone structure. Aiming for the improvement of the prototype's flight stability, two PID controller tuning techniques where validated: Twiddle and ITAE, performing simulations on the software Mission Planner. Furthermore, through the same software, was validated the Potential Fields path planning method, constructed in this work via Python algorithm, pursuing the vehicle's automation.

Keywords: Mobile robotics, Potential fields method, PID, Ardupilot, RepRap3d, Mission Planner.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

$\nabla U(q)$	Vetor gradiente de U na posição q
3D	Tridimensional
ABS	Acrilonitrila-butadieno-estireno
dP	Vetor de variação diferencial
e(t)	Função sinal de erro variante no tempo
ESCs	Electronic Speed Controlers
F(q)	Força artificial na posição q;
F_{att}	Força de atração
F_{rep}	Força de repulsão
IAE	Integral do Erro Absoluto
ISE	Integral Quadrática do Erro
ITAE	Integral do Erro Absoluto Ponderado no Tempo
K_{att}	Ganho da força de atração
K_p	Ganho proporcional
K_{rep}	Ganho da força de repulsão
L	Constante de atraso
P	Vetor de ganhos
PET Engenharia Elétrica	Programa de Educação Tutorial de Engenharia Elétrica
PID	Proporcional-integral-derivativo
q	Posição do robô
$q_{objetivo}$	Posição do objetivo
T	Constante de tempo
T_d	Tempo derivativo
T_i	Tempo integral
U(q)	Campo potencial artificial na posição q
U_{att}	Campo potencial de atração
u(t)	Função sinal de controle variante no tempo
UFJF	Universidade Federal de Juiz de Fora
U_{rep}	Campo potencial de repulsão
VANT	Veículo Aéreo Não Tripulado

ξ	Porcentagem de variação
$\rho(q)$	Distância euclidiana do robô ao obstáculo
ρ_o	Raio de detecção de obstáculo do robô
$\rho_{objetivo}$	Distância euclidiana do robô ao objetivo

SUMÁRIO

CAPÍTULO 1: INTRODUÇÃO

1.1 – MOTIVAÇÃO

A robótica tem seu maior sucesso no mundo da manufatura industrial. Entretanto, hoje ela se estende para diversas outras áreas de aplicação, compondo uma indústria a qual em 2015 recebeu em investimento ao redor do mundo de cerca de US$ 71 bilhões, valor este que tende a crescer anualmente em 17% nos próximos anos (VANIAN, 2016). Além disso, a robótica expande cada vez mais sua atuação na sociedade com a utilização de robôs móveis. Este foi o ponto chave da revolução industrial com manipuladores robóticos estacionários, levando a aumento exponencial na produtividade do setor (FERNANDEZ et al., 2012).

Ambientes perigosos e inóspitos, como Chernobyl, áreas de contaminação ou locais de guerra, tiveram variada aplicabilidade de robôs móveis teleoperados, tendo uso até em missões interplanetárias, como as explorações em Marte (HOWELL, 2016). Outros robôs comerciais atuam não apenas em locais onde humanos não podem ir, mas também em ambientes de vivência humana, dividindo o espaço com este. Neste caso, estes robôs são utilizados não somente por sua mobilidade, mas principalmente pela sua autonomia (SIEGWART et al., 2011). São vistos em aplicações das mais variadas, como corte e transporte de madeira, inspeção de dutos de ventilação, análise de rios e mares através de robôs submarinos, limpeza de chão em casas e supermercados, e guias de turistas em museus (SIEGWART et al., 2011).

Em anos recentes, houve a ampliação do uso de robôs móveis para ainda mais aplicações, através do desenvolvimento de robôs voadores, ou Veículos Aéreos Não Tripulados (VANTs), como são comumente chamados (CORKE, 2011). Seu uso tem sido cada vez mais popular em todo o mundo, tendo aplicações como vigilância civil (BBC, 2016), entrega de suprimentos médicos em áreas remotas (WEBSTER, 2016), capturas de imagens em uso pessoal (SANTINO, 2016), controle de pragas em lavouras (DIÁRIO DIGITAL, 2016), e até distribuição de pontos de acesso à internet em locais de difícil entrega (NEWTON, 2016).

Entre os diversos modelos de VANTs, destacam-se os quadricópteros, veículos multirotores cuja propulsão é feita através de quatro hélices. Estes possuem alta manobrabilidade, podendo voar em segurança até mesmo em interiores, e são considerados de alta simplicidade para voo, não tendo a mesma complexidade mecânica e aerodinâmica de

modelos de asa fixa ou helicópteros (CORKE, 2011). Tais modelos são hoje largamente disponíveis, tanto como produto comercial quanto como projeto em plataforma em código aberto.

A relativamente baixa complexidade mecânica dos multirotores é de certa forma compensada por uma maior complexidade de eletrônica de acionamento dos seus motores para garantir sua estabilidade de voo. Pode-se dizer que o aumento de aplicabilidade destes veículos nos últimos anos se deve à larga evolução e popularização de componentes eletrônicos e linguagens de programação *open-source*.

Ligado diretamente a isto, houve o desenvolvimento de tecnologias de impressão tridimensional para prototipagem de peças, fazendo parte hoje do ambiente de criação de novas tecnologias em diversos ramos, tanto na indústria quanto em universidades (BAK, 2003). Isso abriu portas para ainda mais desenvolvimento na robótica móvel, pela sua facilidade e versatilidade de criação de peças. Neste cenário, vê-se a oportunidade da criação de uma plataforma de robô móvel autônomo de baixo custo, acessível, de fácil implementação.

1.2 – OBJETIVO

O objetivo deste trabalho é construir em plataformas *open-source* um VANT quadrimotor, utilizando de componentes acessíveis e de impressão 3D para tal, criando assim um robô voador de baixo orçamento e totalmente replicável, e validar, por meio de simulações, técnicas de sintonia de controlador proporcional-integral-derivativo (PID) para garantia da sua qualidade de voo, e método Campos Potenciais de planejamento de trajetória para automatização do veículo, criando uma plataforma de aprendizado

1.3 – ESTRUTURAÇÃO DO TRABALHO

Este trabalho está estruturado em 5 capítulos. No capítulo 2 será apresentada uma base teórica utilizada neste trabalho incluindo sistemas de controle, ações de controle, controlador PID, conceitos de autonomia, subsistemas de um robô móvel e método de planejamento de trajetória por campos potenciais. No capítulo 3 será exposta a metodologia utilizada para a construção do *drone*, processos de sintonia do controlador PID de voo e implementação de algoritmo de campos potenciais em simulador. No capítulo 4 serão apresentados os resultados obtidos para testes em voo e simulações. Finalizando, no capitulo 5 serão apresentadas as conclusões e algumas sugestões para trabalhos futuros.

CAPÍTULO 2: FUNDAMENTAÇÃO TÉCNICA

Neste capítulo será discutida a base teórica utilizada neste trabalho, a qual abrange fundamentos de sistemas de controle, entre eles o sistema PID e seus métodos de sintonia, além de apresentar conceitos fundamentais da robótica móvel, como autonomia, propulsão, sensoriamento e alimentação. Por fim, serão mostrados os conceitos do algoritmo de campos potenciais aplicado à robótica.

2.1 – SISTEMAS DE CONTROLE

Os sistemas de controles podem ser fundamentalmente subdivididos em dois tipos de funcionamento: controle a malha aberta e controle a malha fechada (OGATA, 1998).

Sistemas de controle a malha aberta são aqueles cujas ações de controles não são afetadas pela sua saída. Ou seja, num sistema deste tipo, não há medição ou observação do sinal de saída, e não é realizado nenhum tipo de comparação deste com o sinal de referência. Dessa forma, cada sinal de referência inserido na entrada do sistema irá corresponder a uma condição fixa de operação, o que leva o sistema a ter sua precisão dependente de calibrações. Além disto, não há qualquer compensação do sistema sobre possíveis distúrbios em sua operação, fazendo com que estes causem uma diferença entre o resultado obtido e aquele desejado (OGATA, 1998).

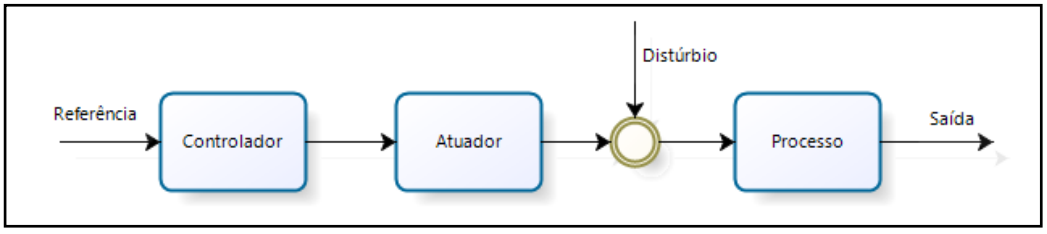

Figura 1 – Configuração de controle em malha aberta (Autor)

Devido a sua simplicidade e baixo custo de implantação, o controle em malha aberta normalmente é utilizado em processos mais simples e previsíveis, ou onde não é exigida grande

exatidão da saída (KEMPF, 2003). Como exemplo de sistemas deste tipo, pode ser citado o forno convencional caseiro, no qual é inserida uma temperatura desejada como entrada, porém o equipamento não realiza qualquer medição de sua temperatura interna para atingir a referência, contando apenas com sua calibração de fábrica. Assim, quaisquer distúrbios como aberturas no forno ou mudanças na temperatura do ambiente podem causar diferenças na temperatura obtida.

Um sistema de controle em malha fechada, também chamado de sistema de controle com retroação (OGATA, 1998), pode ser definido como um sistema em que há um processo de medição da grandeza de saída, comparando esta com o sinal de referência para manter entre ambos uma relação preestabelecida. A diferença entre ambos, chamada de sinal de erro, é utilizada no controlador com o objetivo de reduzi-la e aproximar o resultado à referência desejada (FRANCHI, 2011). Tal função possibilita que o controlador atue sobre o sistema de modo a compensar por possíveis erros e perturbações, mantendo a proximidade entre a referência e a saída.

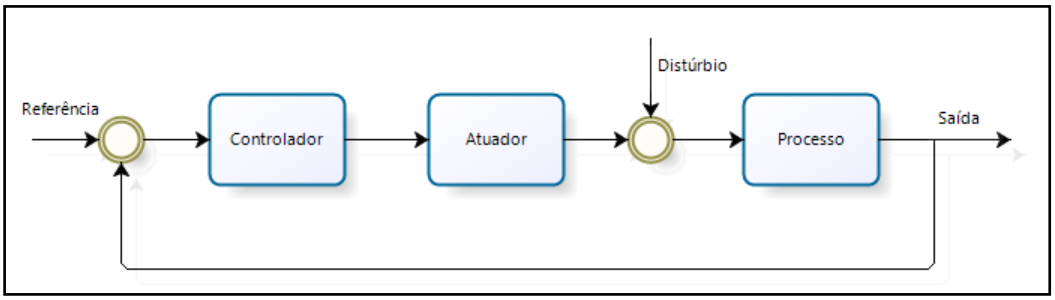

Figura 2 – Configuração de controle em malha fechada (Autor)

As aplicações de controles em malha fechada podem ser vistas em diversas áreas além da engenharia (OGATA, 1998). Um exemplo simples seria um sistema de controle de nível de um tanque de água à vazão. Neste, o nível atual é medido e enviado como sinal ao controlador, o qual utiliza este para controlar a vazão de entrada e saída de líquido do tanque, visando manter o nível do tanque o mais próximo possível daquele imposto pela referência.

2.1.1 – CONTROLE PID

Um controlador de malha fechada, ao comparar a saída medida com o valor de referência, pode produzir de diversas maneiras um sinal de controle que busca reduzir o valor de erro ao mínimo. A maneira utilizada por este controlador de produzir o sinal de controle é chamada ação de controle (OGATA, 1998).

Uma delas, a mais utilizada em diferentes aplicações de controle de processos (VILANOVA; ALFARO, 2011), é chamada de ação de controle proporcional-integral-derivativa, a qual consiste da combinação de três diferentes ações de controle: ação de controle proporcional, ação de controle integral e ação de controle derivativa. Um controlador deste tipo é dotado das vantagens de cada uma destas ações separadamente (OGATA, 1998). Sua forma matemática é descrita na equação 1.

$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(t)\,dt + K_p T_d \frac{de(t)}{dt} \quad \textbf{(1)}$$

Onde:

u(t) – é a função sinal de controle variante no tempo;

e(t) – é a função sinal de erro variante no tempo;

K_p – é o ganho proporcional;

T_i – é o tempo integral;

T_d – é o tempo derivativo;

O primeiro termo, referente a ação de controle proporcional, funciona basicamente como uma amplificadora da função erro com ganho ajustável através da constante de ganho proporcional K_p. Este proporciona que o controlador reaja a partir de variações no erro, porém isoladamente não garante no sistema sua estabilidade segundo Filatov nem a eliminação do erro em regime permanente (COELHO et al., 2003).

O segundo termo da equação, advindo da ação de controle integral, causa uma variação na saída do controlador segundo uma taxa proporcional ao sinal de erro (OGATA, 1998), ou seja, a derivada da saída do controlador varia proporcionalmente ao erro atuante, tendo essa proporção ajustada através da constante de tempo integral T_i. A combinação dos dois primeiros termos resulta em um controlador capaz de eliminar o erro em regime permanente (PAIM, 1997), porém a ação integral aumenta o risco de instabilidade do sistema.

A ação de controle derivativo, indicada no último termo da equação, faz com que o valor da saída do controlador se altere proporcionalmente à taxa de variação do sinal de erro (OGATA, 1998), tendo seu ajuste feito através do tempo derivativo T_d. Esta ação permite que o controlador possa se antecipar a alterações no sinal de erro observando sua derivada, permitindo correções mais efetivas, além de poder diminuir os efeitos desestabilizadores da ação integral (RODRIGUES, 2010), porém possui a desvantagem de amplificar ruídos no sinal de erro.

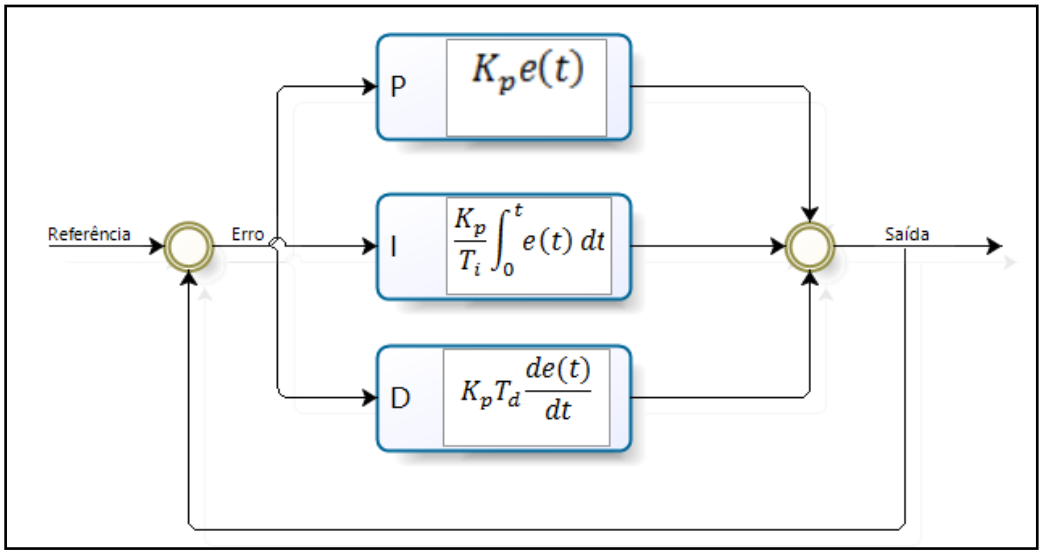

Figura 3 – Diagrama de blocos de controlador PID (Autor com base em OGATA, 1998)

Para se projetar um controlador PID, é preciso obter os parâmetros do controlador (Kp, Ti e Td) necessários para que se adquira a performance do sistema de acordo com as especificações desejadas. Este processo é conhecido como sintonia do controlador. Estes podem ser obtidos de diversas formas a partir do modelo matemático do processo a ser implantado. Porém, em alguns casos, devido à alta complexidade do processo, seu modelo não pode ser obtido facilmente, inviabilizando a abordagem analítica da projeção do controlador PID (OGATA, 1998), sendo necessário adotar um método empírico para obtenção dos parâmetros.

2.1.1.1 – SINTONIA PID - ITAE

Desenvolvido na década de 60 por pesquisadores da Louisiana State University, o método Integral of Time-weighted Absolute Error (ITAE), também conhecido como método da integral do erro absoluto ponderado no tempo, pode ser considerado um dos melhores métodos empíricos para sintonia de controladores (SARAIVA, 2011). Com regras de obtenção próximas ao método Ziegler e Nichols, os parâmetros são obtidos a partir da resposta experimental do processo em malha aberta a uma entrada em degrau (OGATA, 1998).

Este método visa minimizar os critérios de desempenho baseados nas integrais de erro: Integral do Erro Absoluto (IAE), Integral Quadrática do Erro (ISE) e Integral do Erro Absoluto Ponderado no Tempo (ITAE) (SARAIVA, 2011). Para a obtenção dos parâmetros do controlador, é necessária a obtenção das constantes de atraso L e de tempo T, obtidas a partir da curva de reação do sistema em malha aberta. As constantes são determinadas através da reta tangente do ponto de inflexão da curva, medindo-se o ponto em que a reta intercepta o eixo das abscissas, conforme é mostrado na Figura 4.

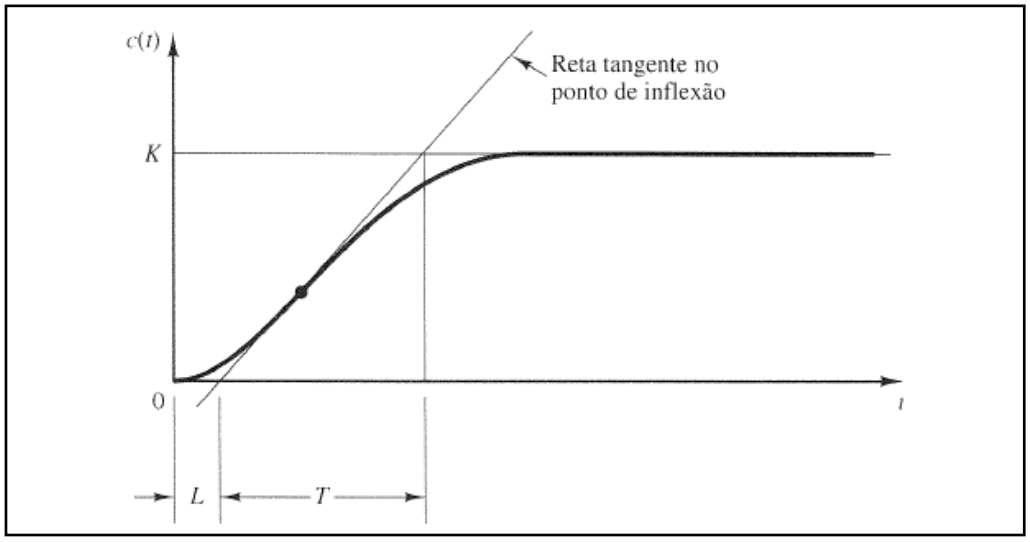

Figura 4 – Método da curva de reação para obtenção de L e T (OGATA, 1998)

A partir das constantes obtidas na curva de reação do processo, os parâmetros do controlador são obtidos através das equações 2, 3 e 4.

$$KK_p = 1{,}35\left(\frac{L}{T}\right)^{-1.0} + 0{,}27 \qquad (2)$$

$$\frac{T_i}{T} = \frac{2{,}5\left(\frac{L}{T}\right)\left[1{,}0+\frac{\left(\frac{L}{T}\right)}{5.0}\right]}{1{,}0+0{,}6\left(\frac{L}{T}\right)} \quad (3)$$

$$\frac{T_d}{T} = \frac{0{,}37\left(\frac{L}{T}\right)}{1{,}0+0{,}2\left(\frac{L}{T}\right)} \quad (4)$$

Onde:

K_p – é o ganho proporcional;

T_i – é o tempo integral;

T_d – é o tempo derivativo;

L – é a constante de atraso;

T – é a constante de tempo.

2.1.1.2 – SINTONIA PID - TWIDDLE

Outro método de sintonia de controladores PID, muito popular em diversos tipos de sistemas, é chamado *Twiddle* (NORVIG; THRUN, 2012). Este possui grande versatilidade, podendo ser aplicado em qualquer processo, tendo sido aplicado inclusive em trabalhos com quadricópteros (GOLDSTEIN, 2012). O método foi criado baseado em estratégia de sintonia fina e nos critérios de desempenho de integrais de erro para gerar uma técnica que minimize o erro médio medido na resposta da planta. Para executar o método, é importante conseguir reproduzir a performance da planta a ser controlada em uma simulação, pois para atingir a minimização do erro podem ser necessárias muitas iterações.

O método tem como parâmetros um vetor P de ganhos a serem inseridos no controlador PID (P = [Kp, Ti, Td]), um vetor dP de variação diferencial para os ganhos de P, e uma porcentagem de variação ξ para os parâmetros do vetor dP. Inicialmente, os valores de P são inseridos no controlador PID, e a simulação executa um teste com estes valores, medindo o erro total E obtido. A seguir, o primeiro parâmetro de P (Kp), é acrescido de dP e o teste é refeito. Caso o valor de E seja menor do que na iteração anterior, o que indica que a variação feita foi na direção certa, o valor de Kp continua sendo acrescido de dP. Caso contrário, tenta-se variar Kp na direção contrária, decrescendo este por dP. Obtendo piora de resultado em ambas as

direções, tenta-se novamente reduzindo o valor de dP por ξ, e assim sucessivamente até que a variação de resultado seja desprezível, passando então para o próximo parâmetro (Ti).

Após executar o processo por todos os parâmetros, são obtidos os valores de Kp, Ti e Td para os quais são obtidos os menores valores de erro atingidos. O método não garante atingir precisamente o mínimo global do sistema, porém se mostra altamente eficiente para a grande maioria dos casos (NORVIG; THRUN, 2012). A execução do método é ilustrada pelo fluxograma da figura 5.

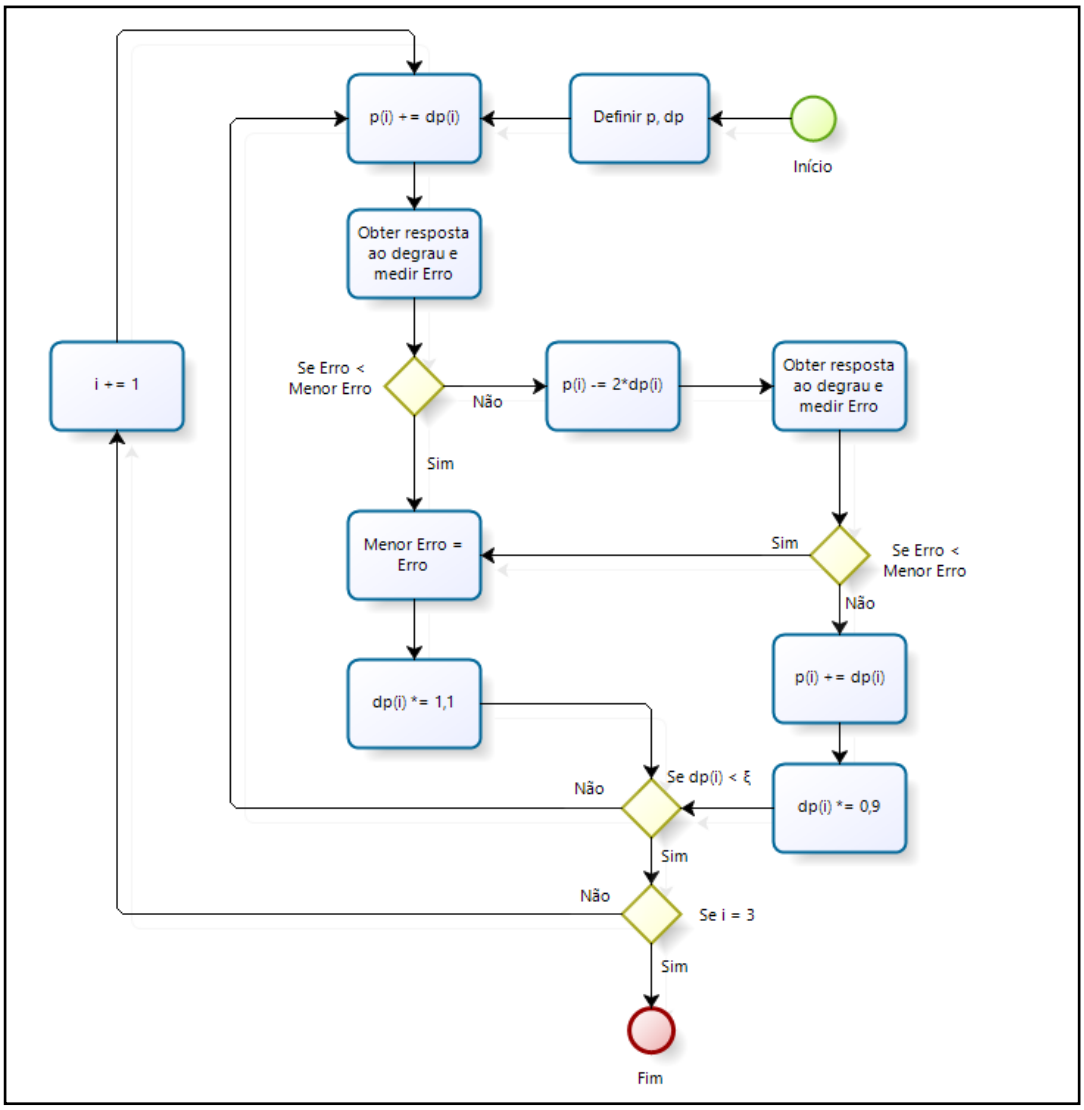

Figura 5 – Fluxograma da sintonia Twiddle (autor).

2.2 – ROBÓTICA MÓVEL

Ao estudar as formas de comportamento de um robô e seu funcionamento, um aspecto essencial é analisar quem toma as decisões (SALICHS et al., 2010), ou seja, qual o grau de autonomia deste robô. Existem diferentes conceitos da palavra autonomia dentro de cada área de estudo, e até mesmo dentro da robótica podem haver vários aspectos envolvidos, mas de maneira geral pode-se definir como a capacidade do robô de atuar de forma independente, sem a necessidade de supervisão ou intervenção de um ser humano (HASELAGER, 2005).

Figura 6 – Diferentes graus de autonomia em robótica móvel (CARDOZO, 2014).

Uma medição precisa do grau de autonomia de um robô pode não ser uma tarefa simples, inclusive pela própria subjetividade do termo. Porém, esta pode ser analisada assumindo que a relação homem-máquina se baseia em um esquema mestre-escravo, conforme descrito por Huang et al. (2004). O trabalho realiza a medição baseada em três fatores: complexidade da missão a ser realizada pelo robô, dificuldade do ambiente onde este é inserido, e interação homem-máquina necessária para ser realizada a missão, sendo o último talvez o fator mais importante. Um robô o qual necessita de teleoperação durante toda ou maior parte de sua missão, portanto, teria baixa autonomia, enquanto outro que seja capaz de adquirir informações e se locomover sobre o ambiente, e realizar suas tarefas sem intervenção humana, poderia ser considerado plenamente autônomo.

2.2.1 – SUBSISTEMAS DE UM ROBÔ MÓVEL

A estruturação de robôs móveis em geral pode ser dividida em quatro subsistemas fundamentais integrados: tração/propulsão, sensoriamento, processamento e alimentação.

Um robô móvel necessita de mecanismos que possibilitem sua locomoção desacoplado de elementos do seu ambiente. Devido à larga variedade de possibilidades de meios de um robô se mover, a seleção da abordagem de locomoção deste é um aspecto importante no projeto de um robô móvel (SIEGWART et al., 2011). Os graus de liberdade de movimentação de um robô podem ser medidos através dos conceitos de holonomia e estabilidade estática, os quais representam as restrições de movimento que um robô possui e sua capacidade de manter-se estável enquanto parado, respectivamente. Dessa forma, robôs aéreos se diferem dos terrestres por possuírem 6 graus de liberdade (CORKE, 2011), tendo translação e rotação nos 3 eixos, sendo, portanto, os mais adequados para missões em terrenos variados (*all-terrain*).

O sistema de sensoriamento do robô abrange uma das mais importantes tarefas num sistema autônomo: adquirir conhecimento sobre si mesmo e o ambiente ao seu redor (SIEGWART et al., 2011). Isto é feito tomando medidas através de sensores e então transformando-as em grandezas físicas.

Os sensores podem ser classificados em dois tipos funcionais principais, nomeados proprioceptivos e estereoceptivos (PIRES; CHAIMOWICZ, 2012). Sensores proprioceptivos são os responsáveis por medir as grandezas internas ao robô, como velocidade de rotação dos motores, orientação, aceleração, níveis das baterias, temperatura dos componentes, corrente dos circuitos internos, entre outros. Os sensores estereoceptivos, por sua vez, medem as grandezas do ambiente em que o robô se encontra, como distâncias a objetos ou objetivos, luminosidade e temperatura do local, ondas sonoras, entre outras grandezas.

O projeto de um robô depende também do seu subsistema de processamento, o qual será responsável por receber os dados de leitura dos sensores, interpretá-los e gerar seus comandos e acionamentos, podendo ser considerado, portanto, o "cérebro" do robô. Para realizar a tarefa, podem ser empregados microcontroladores (como PIC, Atmel, Texas Instruments), microprocessadores, com capacidade mais robusta (como ARM e Intel), ou uma combinação entre ambos, utilizando microcontroladores para tarefas de temporização precisa, como geração de sinais e algoritmos de controle, trabalhando em baixo nível, e

microprocessadores para aplicações-fim, executando tarefas em alto nível, como processamento de imagem e geração de trajetórias.

Por fim, temos como último subsistema essencial para o robô sua alimentação. Esta pode ser provida através de cabos ou baterias, sendo o último mais comum na robótica móvel para que se tenha maior liberdade de movimentação. As baterias são os principais limitadores do tempo de operação do robô, muitas vezes sendo responsáveis por reduzir o grau de autonomia do robô por depender de ação humana para seu recarregamento, salvo por exceções em que o robô é capaz de se mover automaticamente a uma estação de carga (OLIVI, 2015). Entretanto, na maioria dos casos o problema não pode ser resolvido simplesmente inserindo baterias de capacidade maior, pois isto resulta em baterias de maior volume e peso, o que pode ser crucial para um robô móvel, especialmente se tratando de um aéreo.

2.3 – PLANEJAMENTO DE TRAJETÓRIA

Conforme dito anteriormente, a diretriz de tomada de decisões de um robô é um dos aspectos fundamentais de seu funcionamento. Tais diretrizes dizem respeito à cognição da máquina, ou seja, suas instruções de decisões e execuções que o sistema utiliza para alcançar seus objetivos de ordem mais alta. Em se tratando de um robô móvel, um dos principais aspectos de sua cognição, o qual está diretamente ligado à sua robustez de mobilidade é chamada de *competência de navegação* (SIEGWART. et al., 2011). Esta é definida pela capacidade do robô de interpretar as informações dos seus sensores, transformá-los em conhecimento a respeito de sua localização e parte do seu ambiente, e atuar na sua movimentação buscando alcançar seus objetivos da maneira mais eficiente e confiável possível. A técnica utilizada na programação do robô para realizar na prática esta função pode ser definida como seu *planejamento de trajetória*.

2.3.1 – MÉTODO CAMPOS POTENCIAIS

Para que o robô seja capaz de planejar sua rota ao longo do meio, é necessário que este realize uma representação geométrica discretizada do seu ambiente, decompondo o mesmo, e a forma como essa decomposição é feita que difere cada técnica de planejamento de trajetória. Uma das principais técnicas utilizadas, a qual está em rápido crescimento de popularidade em aplicações de planejamento de trajetória e desvio de obstáculos, é chamada de técnica ou algoritmo de campos potencias (KOREN et al., 1991).

O método funciona impondo uma função matemática sobre o espaço, fazendo-o baseado no comportamento de cargas elétricas sob efeito de campos potenciais elétricos, de maneira simples e direta. Nele, o robô é tido como uma carga sob influência de um campo potencial de carga oposta, tendo como centro o objetivo final. Dessa forma, o robô se move seguindo o campo, sendo influenciado por uma força de atração para este. Os obstáculos são representados por áreas de cargas iguais ao robô, agindo como forças de repulsão à movimentação deste. O algoritmo aplica a superposição de todas as forças de influência ao robô, o qual na maioria das vezes é representado por uma carga pontual. Dessa forma, os campos potenciais artificiais guiam suavemente a movimentação do robô até seu objetivo, simultaneamente desviando dos seus obstáculos (SIEGWART et al., 2011).

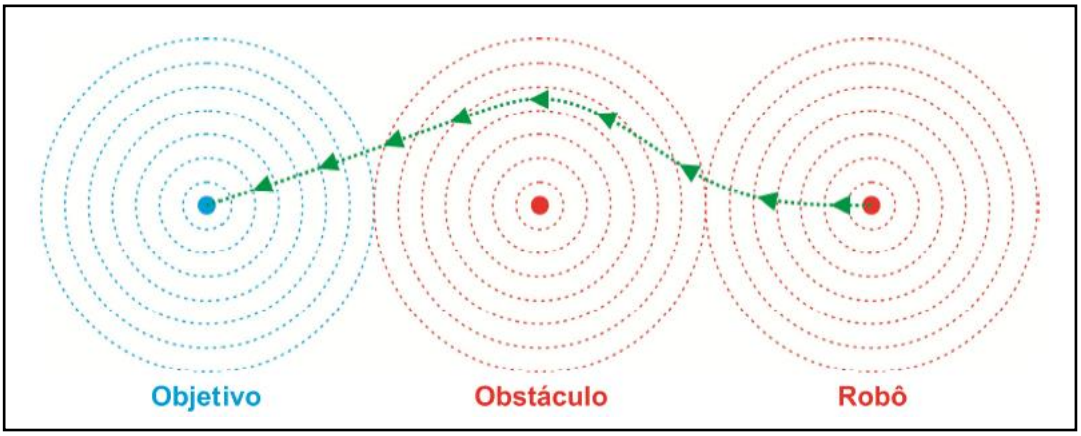

Figura 7 – Campos potenciais gerando trajetória do robô (OLIVI, 2014)

Assumindo o robô como uma carga pontual, a orientação do mesmo pode ser ignorada e seu campo potencial resultante tem apenas duas dimensões. Assumindo uma função campo

potencial derivável U(q), podemos encontrar a força artificial F(q) correspondente, agindo na posição q = (x, y).

$$F(q) = -\nabla U(q) \tag{5}$$

Onde:

F(q) é a força artificial na posição q;

∇U(q) é o vetor gradiente de U na posição q.

O campo potencial atuando sobre o robô é então calculado como a soma do campo de atração do objetivo final e os campos de repulsão dos obstáculos.

$$U(q) = U_{att}(q) + U_{rep}(q) \tag{6}$$

De maneira similar, a força total pode ser decomposta em atrativas e repulsivas.

$$F(q) = F_{att}(q) - F_{rep}(q) \tag{7}$$

O campo potencial de atração pode então ser definido como uma parábola.

$$U_{att}(q) = \frac{1}{2} k_{att} . \rho^2_{objetivo}(q) \tag{8}$$

Onde:

k_{att} é o ganho da força de atração;

$\rho_{objetivo}(q)$ é a distância euclidiana do robô ao objetivo.

Aplicando a equação 5 em 8, temos a equação da força de atração F_{att}, a qual converge linearmente para zero conforme o robô alcança a posição do objetivo.

$$F_{att}(q) = -k_{att} . (q - q_{objetivo}) \tag{9}$$

Onde:

q é a posição do robô;

$q_{objetivo}$ é a posição do objetivo.

O campo de repulsão deve corresponder à ideia de que deve gerar uma força contrária a todos os obstáculos conhecidos, tendo maior intensidade conforme a proximidade do robô

aumenta, porém não deve influenciar no movimento do robô quando este estiver muito longe do obstáculo. Dessa forma, o algoritmo utiliza de um raio de detecção de obstáculos ρ_o, fazendo com que as forças de repulsão sejam calculadas somente para obstáculos encontrados dentro deste.

$$U_{rep}(q) = \begin{cases} \frac{1}{2} k_{rep} \left(\frac{1}{\rho(q)} - \frac{1}{\rho_o} \right)^2, & \rho(q) \leq \rho_o \\ 0, & \rho(q) \geq \rho_o \end{cases} \quad (10)$$

Onde:

k_{rep} é o ganho da força de repulsão;

$\rho(q)$ é a distância mínima entre o robô e o obstáculo;

ρ_o é o raio de detecção de obstáculo do robô.

Analogamente ao que foi feito para encontrar a força de atração, podemos encontrar a força de repulsão aplicando a equação 5 em 10.

$$F_{rep}(q) = \begin{cases} k_{rep} \left(\frac{1}{\rho(q)} - \frac{1}{\rho_o} \right) \frac{1}{\rho^2(q)} \frac{(q - q_{obstáculo})}{\rho(q)}, & \rho(q) \leq \rho_o \\ 0, & \rho(q) \geq \rho_o \end{cases} \quad (11)$$

Onde:

$q_{obstáculo}$ é a posição do obstáculo.

Finalmente, inserindo as equações 9 e 11 na equação 7, temos a força resultante atuando no robô pontual, a qual, sendo gerada a partir das forças de atração e repulsão, distanciam o robô dos obstáculos do ambiente e o movem em direção ao seu objetivo. A figura 8 ilustra um cenário típico do algoritmo, contendo um ponto de mínimo no ambiente (o objetivo), com diversos picos ao seu redor (os obstáculos).

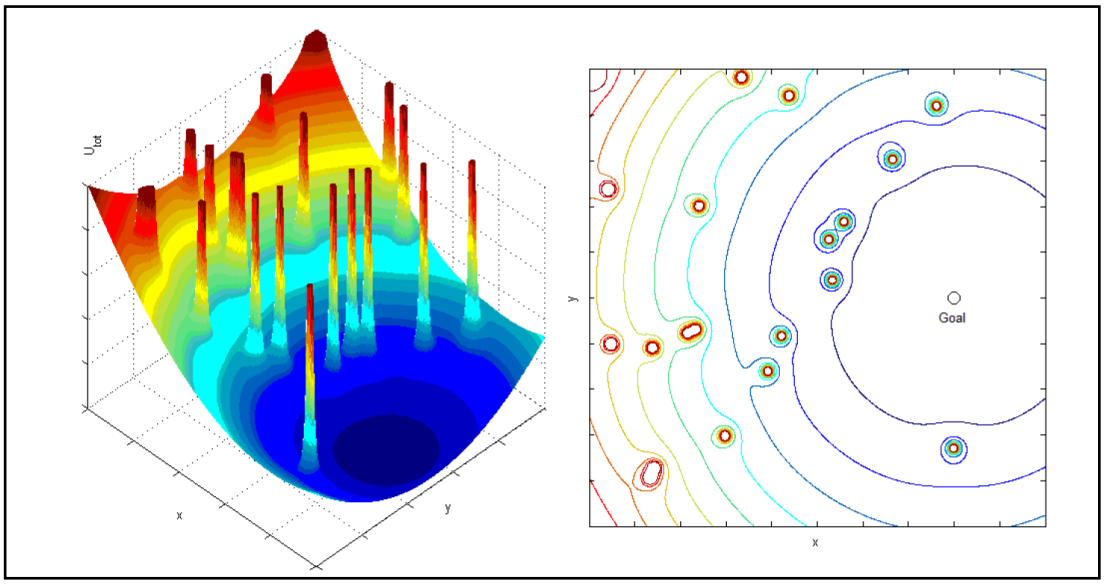

Figura 8 – Cenário típico do algoritmo de Campos Potencias (OLIVI, 2014)

Sob condições ideais, o algoritmo leva o robô ao seu objetivo, desviando suavemente dos obstáculos, de maneira similar a uma bola rolando por obstáculos ladeira abaixo (SIEGWART et al., 2011). Porém, a maior falha do algoritmo está em cenários que possam existir pontos de mínimos locais, os quais podem ocorrer caso o robô se depare com um obstáculo côncavo em sua trajetória, causando que este não alcance seu objetivo final.

CAPÍTULO 3: METODOLOGIA

Este capítulo tem como objetivo apresentar todas as etapas de construção do *drone*, incluindo a seleção dos componentes e esboços da estrutura, além de mostrar os processos de obtenção de sintonia de parâmetros PID para o controle de arfagem e rolagem do voo do *drone* em simulação. Dois métodos diferentes foram utilizados, visando obter o mais adequado para aplicação no modelo real. Por fim, o algoritmo criado para execução do método dos campos potenciais para planejamento de trajetória do robô será apresentado.

A etapa de construção deste *drone* fez parte de um projeto realizado em equipe dentro do Programa de Educação Tutorial de Engenharia Elétrica da Universidade Federal de Juiz de Fora (PET Engenharia Elétrica – UFJF), sendo o autor deste trabalho responsável pela liderança da equipe durante esta etapa. As etapas de sintonia PID e algoritmo de planejamento de trajetória, realizadas em simulação, foram realizadas somente pelo autor. Em paralelo, a equipe deu continuidade do projeto de construção, entretanto esta continuidade não será apresentada por este trabalho.

3.1 – CONSTRUÇÃO DO DRONE

Toda a estrutura do drone foi construída de forma manual, e maior parte das peças da armação foram impressas através de impressora 3D, sendo todos os desenhos de autoria do próprio projeto, criados pensando em cada uma das necessidades deste. Cada peça foi desenhada no *software SolidWorks* e impressa em termoplástico Acrilonitrila-butadieno-estireno (ABS), um material de alta resistência e tenacidade, essenciais para resistir aos impactos e vibrações da aeronave, além de ter baixo custo e ser capaz de resistir a distorção térmica (BOM, 2008).

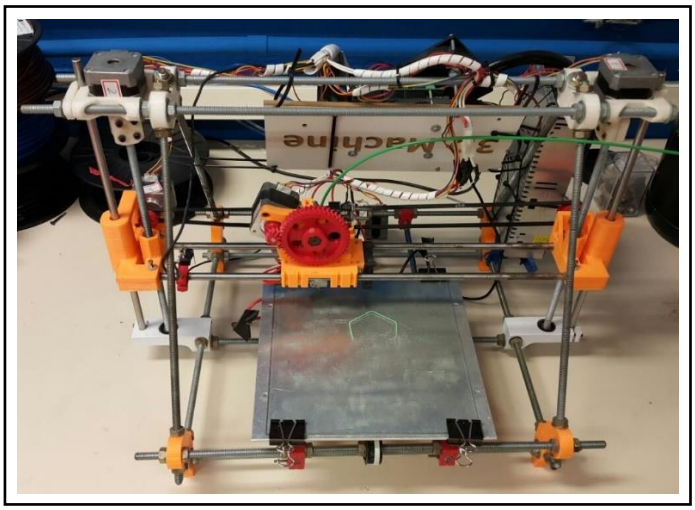

Figura 9 – Impressora 3D utilizada na criação das peças da estrutura (Autor)

3.1.1 – CONSTRUÇÃO DA ESTRUTURA

A parte central da estrutura, a qual abriga os componentes eletrônicos do drone (como o microcontrolador, bateria, receptor de rádio e outros periféricos), foi composta de três peças principais. Estas podem ser vistas na figura 10, e são: duas placas utilizadas para apoiar os componentes, e a terceira com finalidade de sustentar os braços da estrutura. Os braços são compostos por tubos de plástico reforçados com fibra de carbono, fazendo parte das poucas peças da armação não criadas pela impressão 3D, devido à necessidade de maior rigidez e resistência destes.

Além destas, outras três peças foram criadas para esta estrutura, as quais são mostradas na figura 11: peças e apoio central para os braços, suportes para os motores e trens de pouso. O esquemático em *SolidWorks* da estrutura completa pode ser visto na figura 12, enquanto a figura 13 mostra toda a estrutura após impressão e montagem.

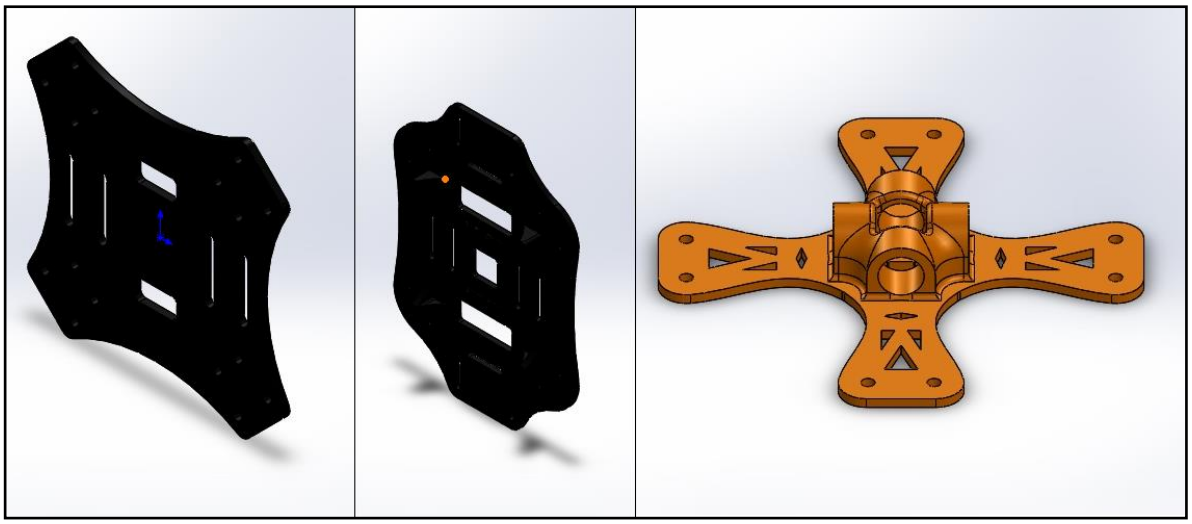

Figura 10 – Peças de base superior, inferior e suporte interno – esquemático em SolidWorks (Autor)

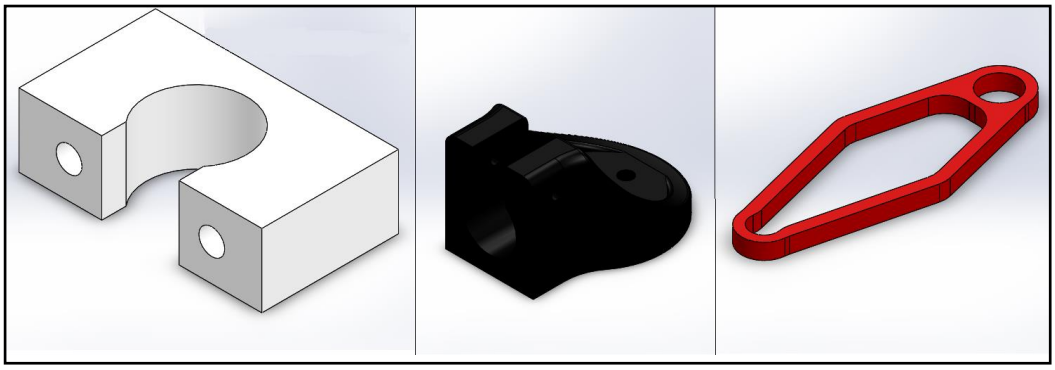

Figura 11 – Peças de apoio central, suporte de motores e trem de pouso – esquemático em SolidWorks (Autor)

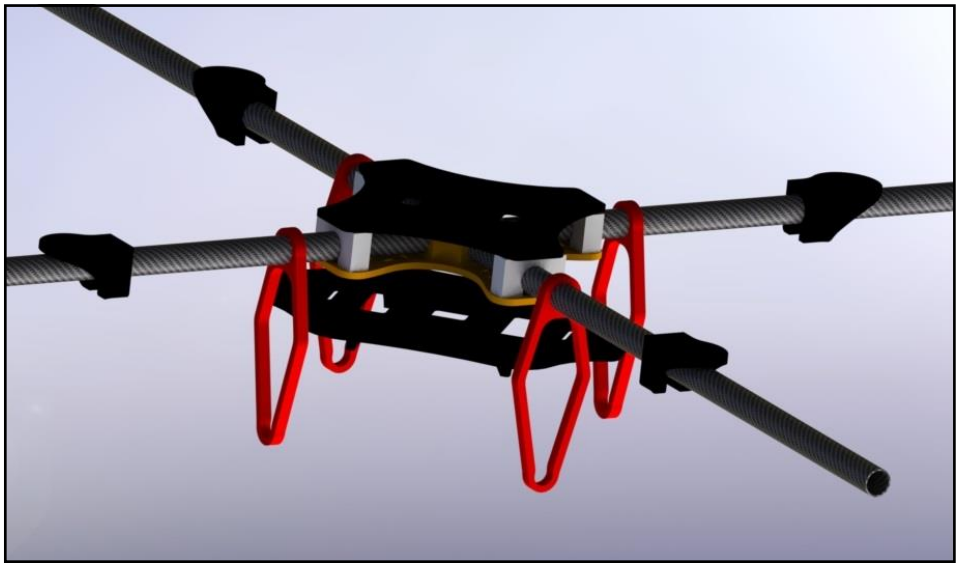

Figura 12 – Montagem completa do drone – esquemático em SolidWorks (Autor)

Figura 13 – Estrutura impressa e montada (Autor)

3.1.2 – ESCOLHA DOS COMPONENTES

Entre todos os componentes do veículo, pode-se dizer que um dos mais importantes é o controlador, e para tal função foi escolhido o dispositivo *ArduCopter* APM 2.6, compondo assim o subsistema de processamento do robô. O microcontrolador se baseia na plataforma *open source Arduino*, sendo construído especificamente para funcionar como controlador de voo de VANTs (Veículo Aéreo Não-Tripulado) multirotores. O dispositivo possui embarcado todos os sensores e recursos necessários para realizar o controle de voo e navegação do drone, como giroscópio, acelerômetro, barômetro, dispositivo GPS, bússola, além de saídas analógicas

para acionamento dos motores e porta serial para envio de dados de telemetria, fazendo com que o dispositivo contemple também o subsistema de sensoriamento.

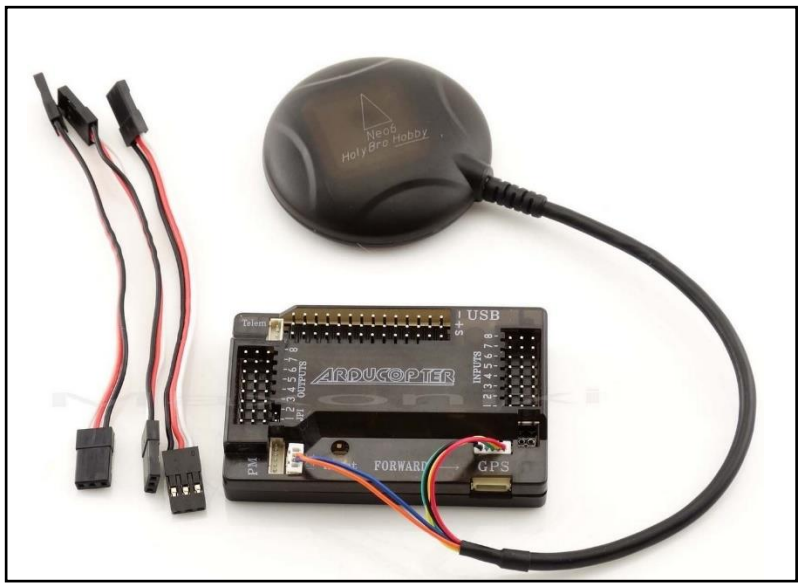

Figura 14 – Microcontrolador Arducopter APM 2.6 (INGO, 2016).

Para o subsistema de propulsão da aeronave, seria necessário um conjunto capaz de sustentar no mínimo o peso de 1,5Kg estimado do drone com a estrutura e todos os componentes. Por isso, foram escolhidos para tal os motores *Emax* 2822, os quais são máquinas de comutação eletrônica, sem escovas e de ímãs permanentes no rotor. Combinados com hélices 10x4.7'', estes são capazes de gerar até 800g de empuxo cada, o que resultaria em 2,4Kg de empuxo máximo do conjunto.

Figura 15 – Motor *brushless Emax* 2822 e hélices APC 10x4.7 (Autor)

Como os motores escolhidos são trifásicos de corrente alternada, estes necessitam de dispositivos para realizar seu acionamento, capazes de converter a corrente contínua da bateria para suas saídas de forma a serem modulados de acordo com os sinais analógicos de tensão advindos do microcontrolador. Para isto foram utilizados ESCs (*Electronic Speed Controlers*) *HobbyKing* 25-30A, capazes de suportar a corrente máxima de 16A de cada motor.

Ao projetar o subsistema de alimentação do drone, era preciso ter um tempo de autonomia de voo por volta de 10 minutos, sem sacrificar demasiadamente a performance da aeronave devido a peso excessivo. Para isto, foi escolhida a bateria *Turningy MultiStar* de 4000 mAh de três células, a qual fornece tensão de 11.1 V, compatível com a alimentação dos motores. A montagem completa do drone, incluindo a inserção de todos dispositivos, pode ser vista na figura 17.

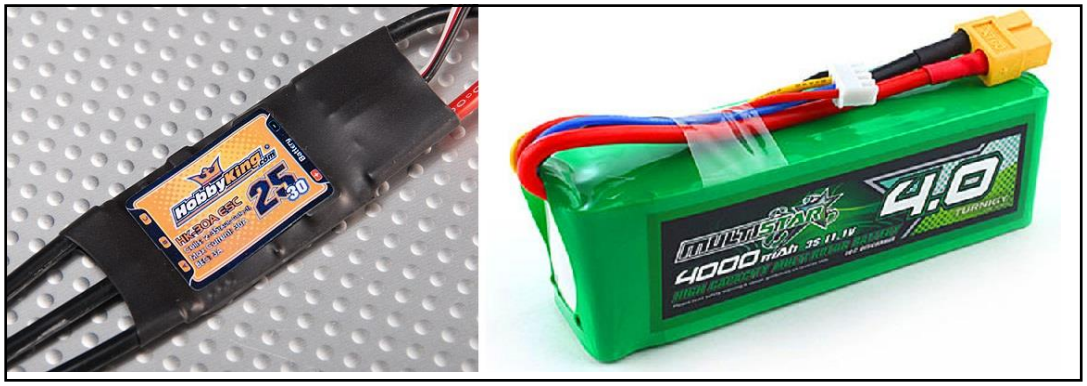

Figura 16 – ESC *HobbyKing* 25-30A e Bateria *Turningy MultiStar* 4000mAh (Autor)

Figura 17 – Estrutura completa com componentes instalados (Autor)

3.2 – SIMULAÇÃO – SINTONIA PID

Nesta seção, dois tipos de métodos empíricos de sintonia de PID foram aplicados. Um método de análise de curva de reação, e outro de minimização de erro, para otimização do controle de voo em simulação, buscando encontrar o método mais adequado para aplicação no drone construído. Métodos que envolvessem processos de modelagem da função de transferência da planta não foram utilizados pela complexidade de realizar tal procedimento tanto em simulação quanto no modelo real.

O primeiro método aplicado foi o ITAE, o qual consiste de um método de análise de curva de reação. Conforme descrito anteriormente, neste método os parâmetros de controle PID são obtidos através de parâmetros medidos na curva de resposta ao degrau do sistema em malha aberta, a qual é mostrada na figura 18.

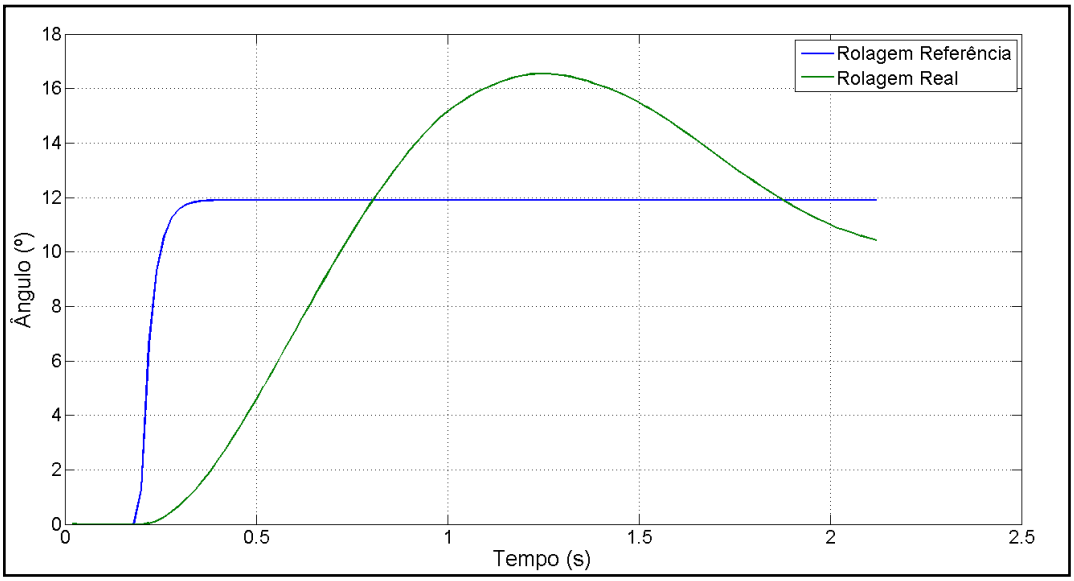

Figura 18 – Resposta ao degrau em malha aberta – simulação (Autor)

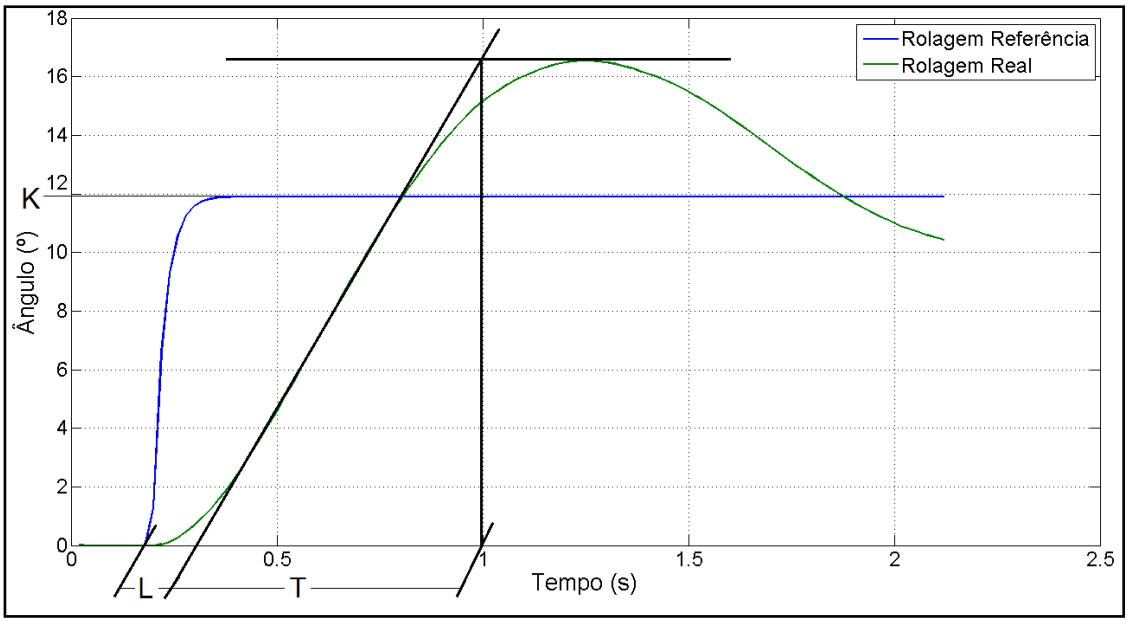

Figura 19 – Obtenção dos parâmetros K, L e T – simulação (Autor)

O degrau de entrada inserido foi de apenas 20% do valor máximo, a fim de evitar descontrole do veículo no teste. Através da resposta observada na figura 18, os parâmetros L e T são obtidos conforme mostrado na figura 19, onde K = 11,91, L = 0,14 e T = 0,68. Pode-se obter os parâmetros PID aplicando estes valores nas equações 2, 3 e 4 (OGATA, 1998).

$$11{,}91K_p = 1{,}35\left(\frac{0{,}14}{0{,}68}\right)^{-1.0} + 0{,}27 \tag{11}$$

$$\frac{T_i}{0{,}68} = \frac{2{,}5\left(\frac{0{,}14}{0{,}68}\right)\left[1{,}0+\frac{\left(\frac{0{,}14}{0{,}68}\right)}{5.0}\right]}{1{,}0+0{,}6\left(\frac{0{,}14}{0{,}68}\right)} \tag{12}$$

$$\frac{T_d}{0{,}68} = \frac{0{,}37\left(\frac{0{,}14}{0{,}68}\right)}{1{,}0+0{,}2\left(\frac{0{,}14}{0{,}68}\right)} \tag{13}$$

$$K_p = 0{,}5089 \tag{14}$$

$$T_i = 0{,}2516 \tag{15}$$

$$T_d = 0{,}0538 \tag{16}$$

O segundo método aplicado, *Twiddle*, requer diversas iterações para que se alcance os valores ditos ideais, conforme mostrado na seção 2. Dado que o processo de execução de cada resposta e inserção de novos parâmetros para teste não era feito online, e sim de forma manual, algumas simplificações foram feitas visando reduzir a quantidade de execuções a serem feitas, como por exemplo o uso de parâmetros PID iniciais como os originais do software (Kp = 0.0135, Ti = 0,090, Td = 0,0036), ao invés dos inicias em zero conforme o recomendado pelo método. O método aplicado é mostrado no fluxograma da figura 20.

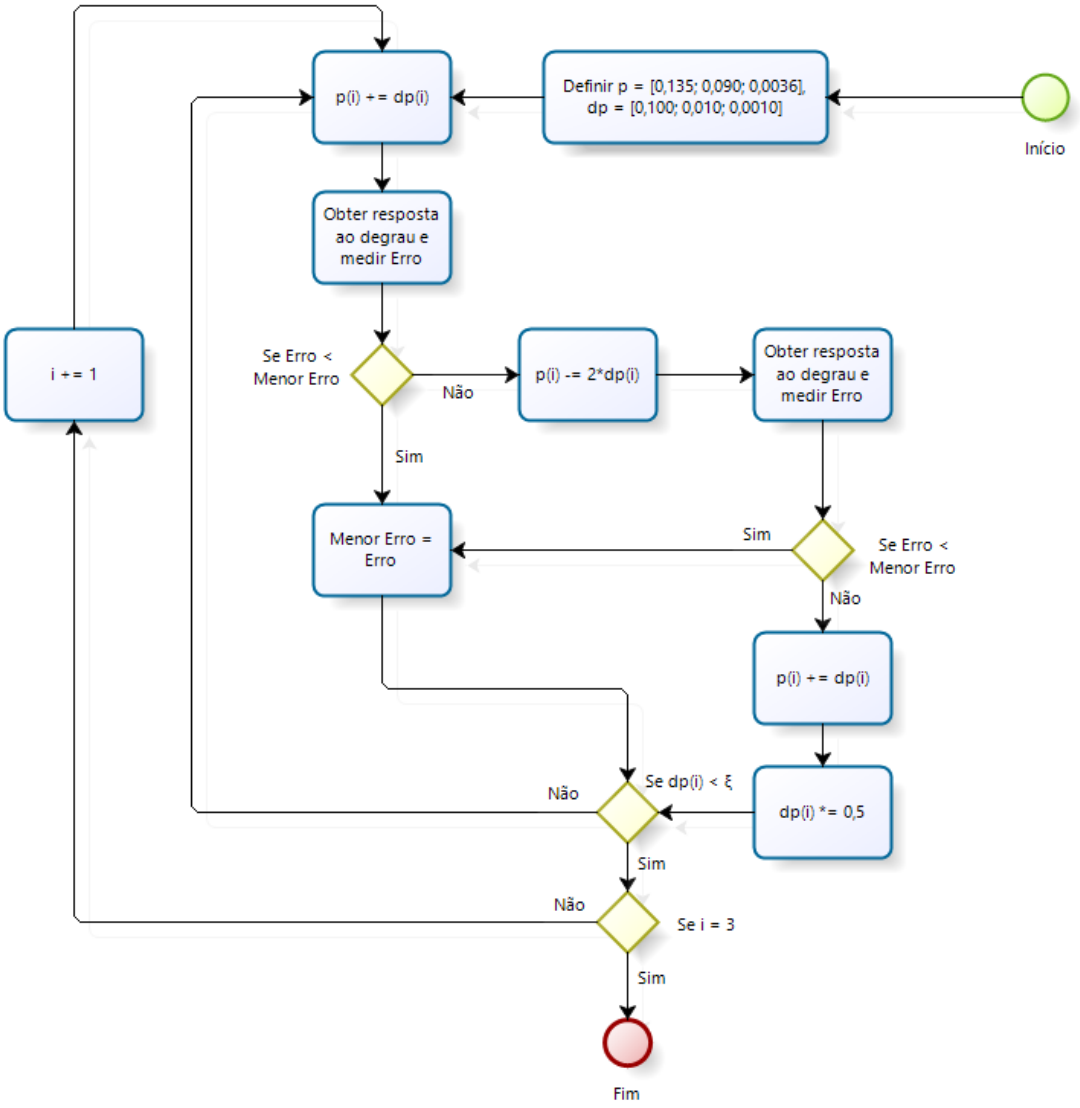

Figura 20 – Fluxograma do método aplicado (Método *Twiddle*) - (Autor)

.Apesar das mudanças, foi mantido o objetivo de buscar por um ótimo de erro de performance, sendo este medido pelo método de Integral Absoluta do Erro (IAE).

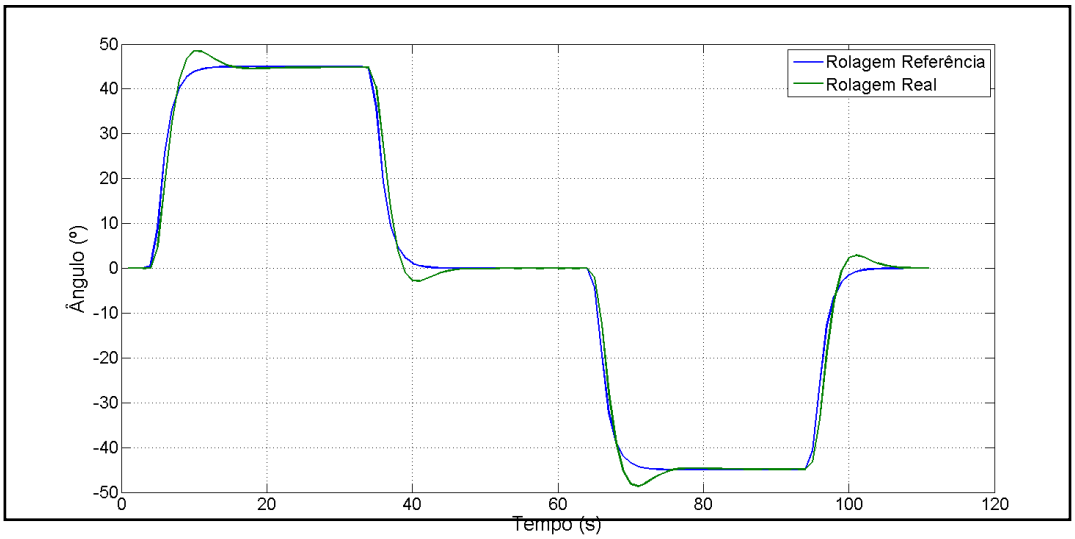

Figura 21 – Resposta ao degrau antes de otimização (Método *Twiddle*) - (Autor)

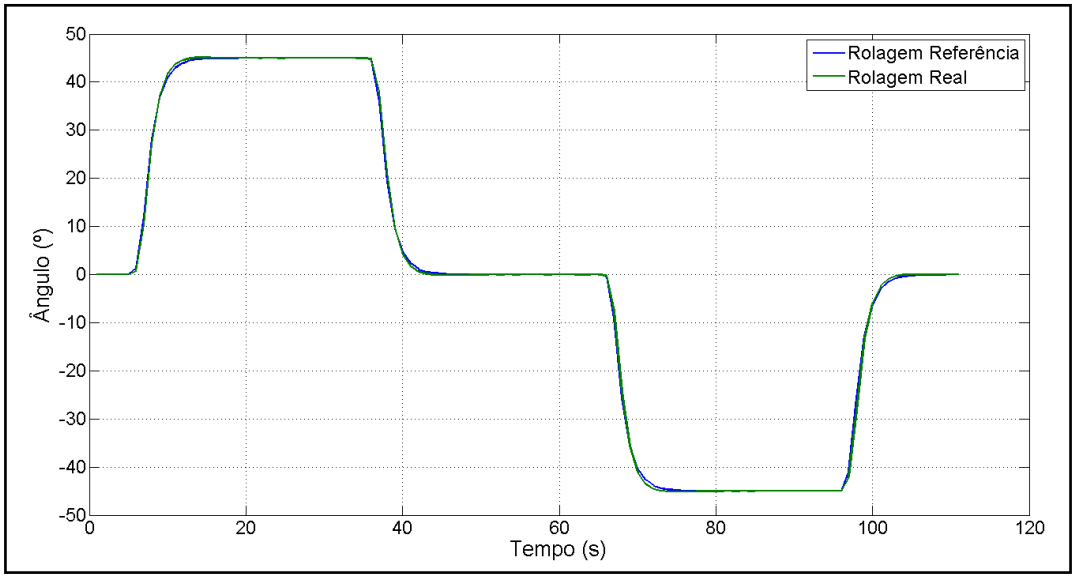

Figura 22 – Resposta ao degrau após otimização (Método *Twiddle*) - (Autor)

As figuras 21 e 22 mostram o resultado de resposta ao degrau com seus respectivos valores de PID e erros resultantes. De maneira geral pode-se notar que o sistema foi muito mais sensível a mudanças no valor de Kp, enquanto as variações de Ti e Td causaram muito pouca ou nenhuma melhoria na resposta. A tabela 1 mostra todas as iterações realizadas e o valor IAE obtido em cada uma. Nela vemos que a iteração nº 7 foi a melhor sucedida, com valores Kp = 0,435, Ti = 0,085 e Td = 0,0036.

Tabela 1 – Parâmetros PID e IAE correspondente

Nº	Kp	Ti	Td	IAE
1	0,135	0,090	0,0036	146,60
2	0,235	0,090	0,0036	78,25
3	0,335	0,090	0,0036	45,55
4	0,435	0,090	0,0036	31,79
5	0,435	0,100	0,0036	31,56
6	0,435	0,080	0,0036	32,08
7	**0,435**	**0,085**	**0,0036**	**31,03**
8	0,435	0,085	0,0046	32,03
9	0,435	0,085	0,0026	31,72
10	0,435	0,085	0,0031	31,52
11	0,435	0,085	0,0041	36,28

3.3 – SIMULAÇÃO – ALGORITMO CAMPOS POTENCIAIS

Buscando um primeiro passo para a automatização do VANT construído, foi criado um algoritmo na linguagem *Python* objetivando a aplicação dos conceitos de campos potenciais da robótica para obter um programa de planejamento de trajetória autônomo para o veículo. A linguagem de programação escolhida foi esta devido a compatibilidade da mesma com o *software Mission Planner*, nativo da plataforma *ArduCopter*, o qual possui recurso para execução de scripts de execução de comandos para drones dentro de sua interface. O algoritmo criado é mostrado em forma de fluxograma na figura 23.

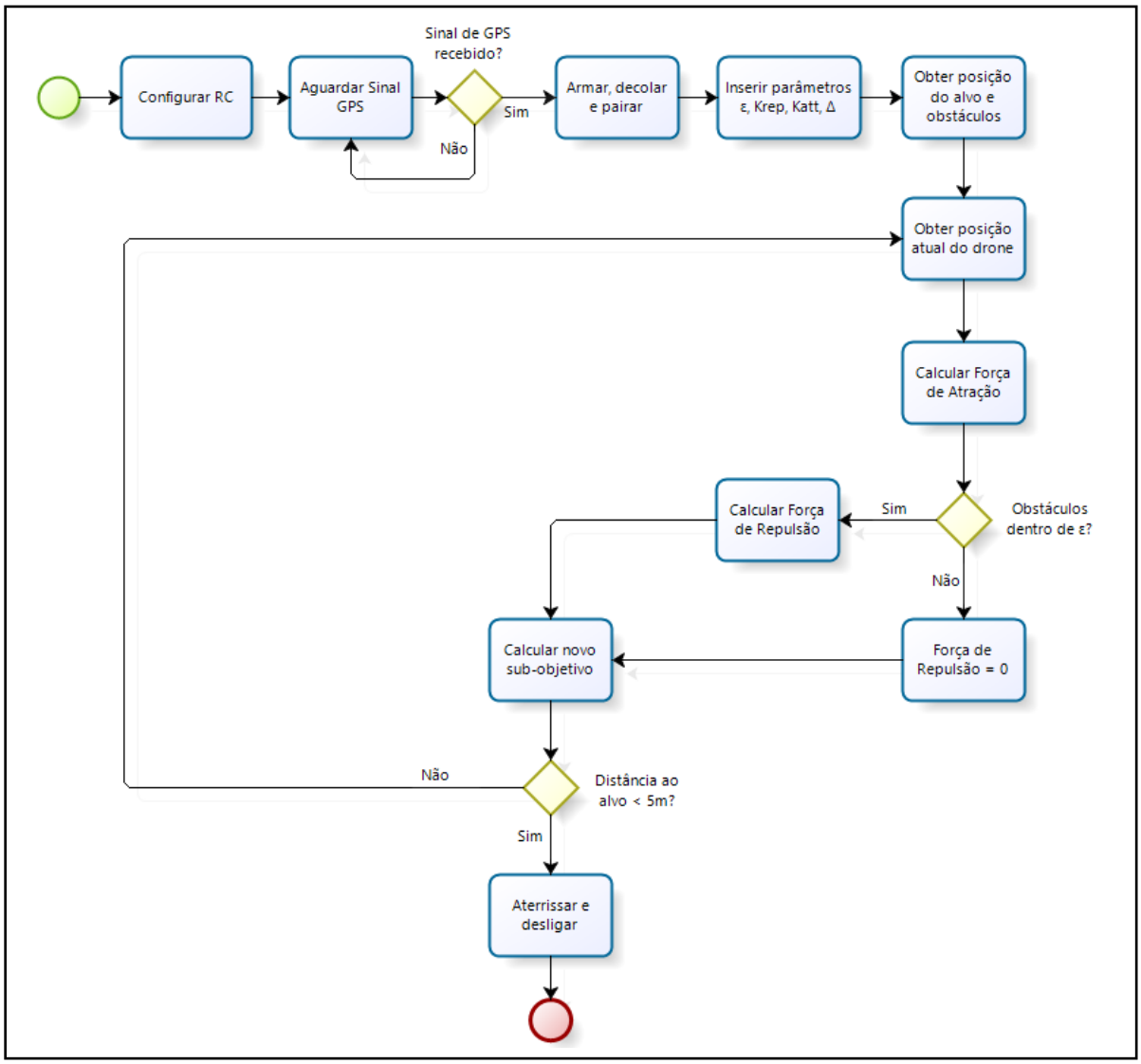

Figura 23 –Fluxograma – Algoritmo em *Python* (Autor)

Uma vez que qualquer interferência na execução do algoritmo através do rádio durante o voo pode prejudicar seu funcionamento, a primeira função executada insere todas as variáveis de entrada de rádio em valor neutro. Esta função, porém, não impossibilita que o algoritmo seja cancelado pelo rádio durante o voo, caso seja necessário. Em seguida, o programa aguarda o recebimento de sinal de GPS para continuar sua execução, e ao recebê-lo, inicializa os motores, realiza decolagem e se mantém estacionário em altitude pré-estabelecida durante alguns segundos, de forma a aguardar estabilidade antes de dar sequência no programa.

Ainda em etapa de pré-configuração, é necessário obter as informações dos parâmetros de campos potenciais. A variável ε indica o tamanho do raio de detecção de obstáculos do *drone* (em metros), as constantes K_{rep} *e* K_{att} são responsáveis por balancear a intensidade das forças

de repulsão e atração, respectivamente, enquanto Δ estabelece a distância do próximo sub-objetivo calculado.

A seguir são estabelecidas as coordenadas geográficas de latitude e longitude de tanto o objetivo final da rota, quanto dos obstáculos. Estes são feitos de forma manual, supondo que o terreno onde será realizado o voo foi previamente mapeado.

Feitas todas as configurações necessárias, o algoritmo de campos potenciais propriamente dito é iniciado, sendo seu trecho principal composto por um *loop* o qual é executado até que a distância detectada entre a posição atual do drone e o objetivo final do algoritmo seja inferior a um valor em metros pré-estabelecido (no caso, o valor utilizado foi de 5 metros, devido à precisão do sistema de localização utilizado estar por volta de 4 metros). Após obter a nova localização da aeronave, o vetor força de atração é então calculado.

A seguir, o algoritmo necessita de detectar os obstáculos ao redor do drone, e reagir conformemente. Assim, são medidas as distâncias em metros do veículo a cada um dos obstáculos pré-informados, e são filtrados aqueles que estão a uma distância maior que o raio **ε** estabelecido. Os obstáculos dentro do raio de detecção são depois utilizados para o cálculo da força de repulsão, que mudará a direção de movimento do *drone* adequadamente.

Calculadas ambas as forças de atração e repulsão, é possível obter a força resultante através da soma vetorial de ambas. O novo sub-objetivo, então, é calculado a partir do vetor força resultante normalizado, multiplicado pela constante Δ. O objetivo é então efetivamente inserido no *drone* através de comunicação *Mavlink* com o microcontrolador. Ao se aproximar do objetivo final e, portanto, encerrar o *loop* do algoritmo, são enviados ao quadricóptero comandos de aterrissagem e desligamento em segurança no local.

CAPÍTULO 4: RESULTADOS

Neste capítulo, serão apresentados os resultados obtidos em curvas de telemetria comparativas de referência e saída, tanto no voo real quanto na simulação, visando observar a qualidade das respostas dos sistemas em ambas as situações, e buscando melhorias dos resultados com mudanças nos parâmetros de PID. Além disso, serão mostradas as trajetórias e respostas obtidas na aplicação do algoritmo de campos potenciais em simulações.

Todos os gráficos e análises aqui mostrados foram obtidos por meio das leituras de telemetria, geradas automaticamente pela plataforma *Ardupilot* utilizada neste trabalho. Após a realização do voo cada conjunto de leituras era então exportado para o *software MATLAB*, o qual permitia uma análise mais profunda dos dados obtidos.

Nos resultados adquiridos, buscou-se analisar separadamente cada variável de angulação de voo do multicóptero (rolagem, arfagem e guinada). Cada análise tem como objetivo comparar os valores de referência, gerados pelo software a partir dos valores de entrada obtidos, com os valores reais de saída. Todos os gráficos possuem taxa de amostragem de 10 amostras/s, e valores obtidos em graus.

4.1 – RESULTADOS VOO EXPERIMENTAL

Visualmente, o *drone* obteve desempenho satisfatório em voo. Mostrou-se capaz de cumprir algumas manobras simples sem perder sua estabilidade e realizar voos curtos e aterrissagens autônomas sem grandes complicações, apesar de mostrar dificuldade em se manter estável ao receber comandos de maior intensidade (exigindo rolagem ou arfagem acima de 15 graus de inclinação em relação ao horizonte).

Figura 24 – Quadricóptero durante primeiro voo (Autor)

Após a realização do voo, foram analisados os dados de telemetria obtidos, e assim as perdas de estabilidade e dificuldades de voo da aeronave puderam ser observadas em maior detalhe. Abaixo, são apresentadas as respostas de sistema adquiridas no voo real experimental do *drone*, comparando durante um mesmo período de tempo cada uma das três angulações de voo.

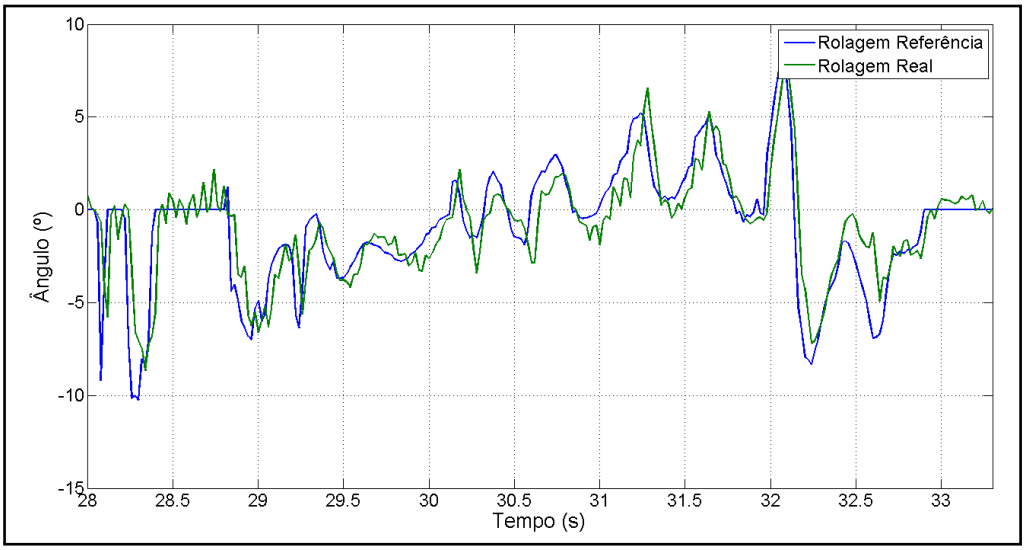

Figura 25 – Comparativo de rolagem referencial e saída – voo real (Autor)

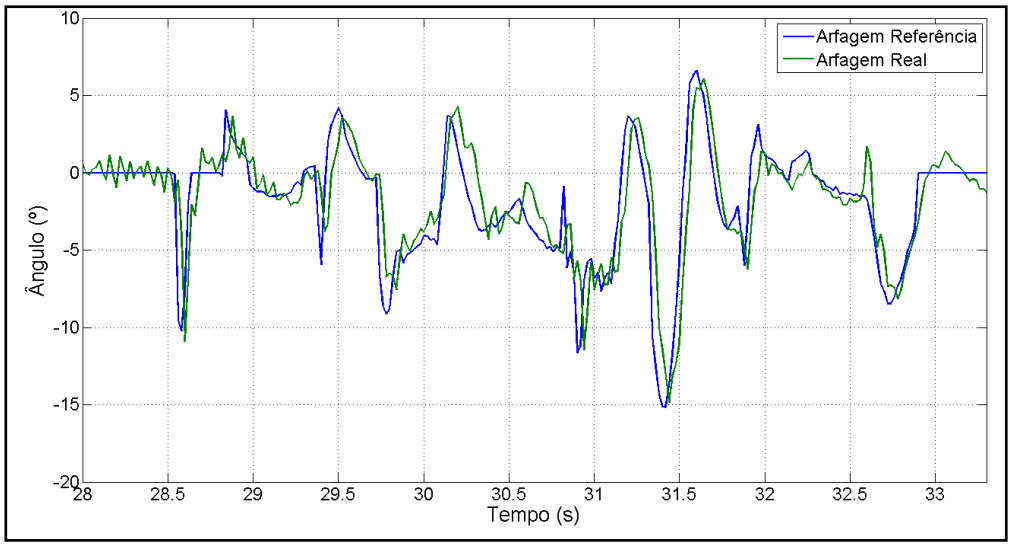

Figura 26 – Comparativo de rolagem referencial e saída – voo real (Autor)

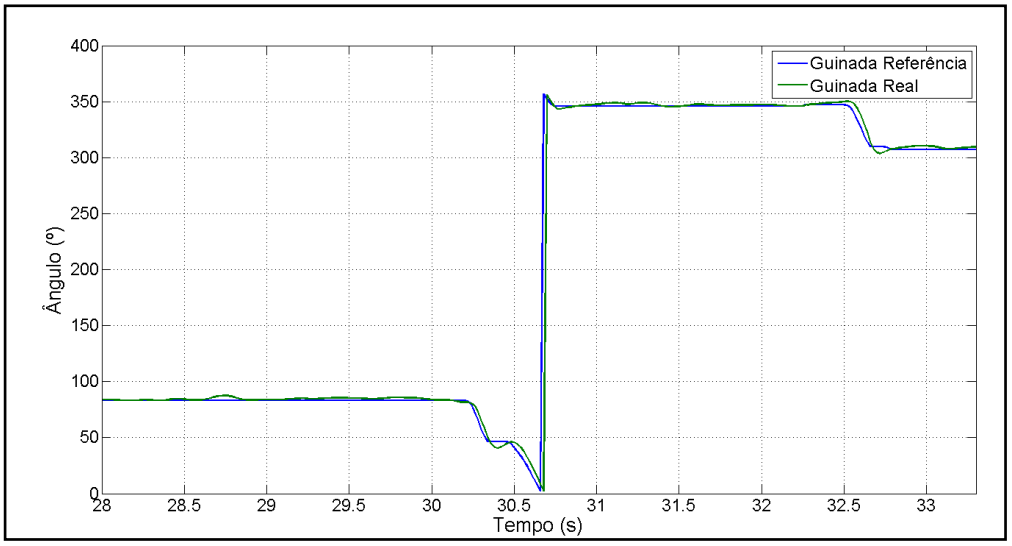

Figura 27 – Comparativo de guinada referencial e saída – voo real (Autor)

Nas três variáveis analisadas, mostradas nas figuras 25, 26 e 27, a saída obtida acompanha sua referência, porém com diferentes complicações, de maneira geral difíceis de observar a olho nu durante voo, como atraso significativo em diversos momentos, constantes vibrações, excesso de *overshoot* e, especialmente no caso da rolagem e arfagem, grandes oscilações nos momentos em que a referência se manteve constante, mostrando a dificuldade do veículo em se manter estável durante voo estacionário.

Tais problemas podem ter como causas: fragilidade da estrutura, causando dobras nos braços do *frame*, vibrações dos motores devido a problemas em sua fixação, turbulências causadas por "efeito de solo" devido à baixa altitude dos voos (HSIUN et al., 1996),

interferência de ventos intensos durante o voo e, finalmente, falta de um processo prévio de sintonia dos parâmetros de PID dos controladores.

Utilizando os dados aqui mostrados, foram realizados trabalhos de correção, como revisão da estrutura montada, melhoria das fixações dos motores e inserção de materiais para reduzir sua vibração, entre outras ações. Entretanto, este trabalho teve como foco apenas a obtenção de um método de sintonia de PID adequado para este modelo.

4.2 – RESULTADOS SINTONIA PID

As figuras 28, 29 e 30 mostram as curvas de reação de rolagem, arfagem e guinada obtidas em simulação com os parâmetros PID originais do sistema, num teste de voo livre próximo ao realizado com o *drone* construído, buscando um comparativo entre ambos. Conforme esperado, os resultados em simulação eliminam a maior parte dos fatores de erros observados em voo real, comparando ambos em manobras de voos similares. O *software* de simulação assume uma estrutura de aeronave e condições de voo próximas a ideais, levando em consideração apenas a inércia, as saídas do controlador e pequenas perturbações para gerar as curvas de resposta. Tal modelo permite enfoque maior na melhoria dos parâmetros de controle da plataforma.

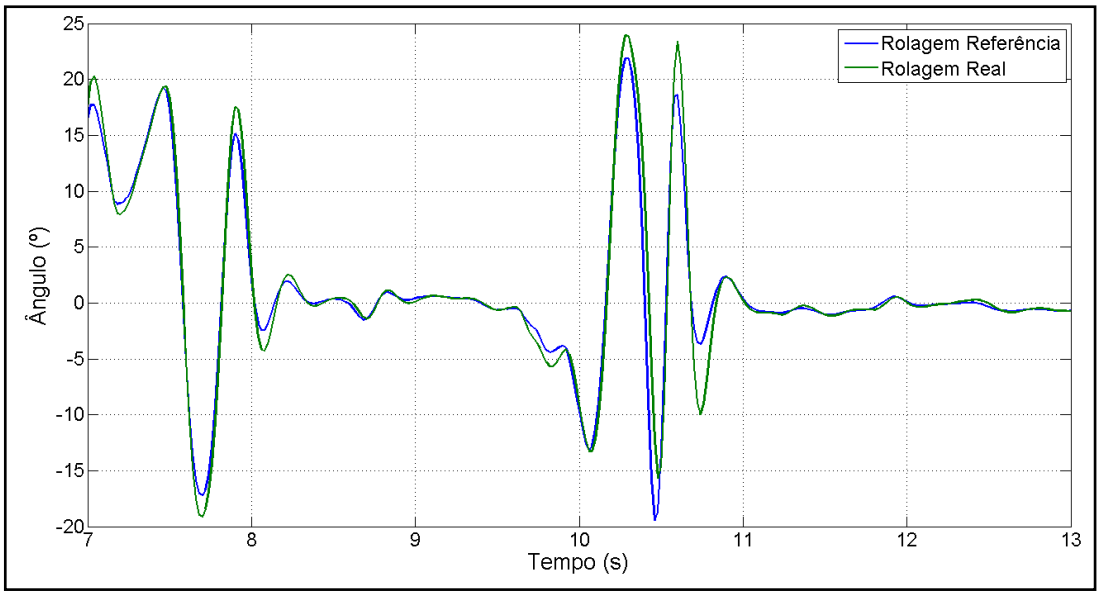

Figura 28 – Comparativo de rolagem referencial e saída – simulação (Autor)

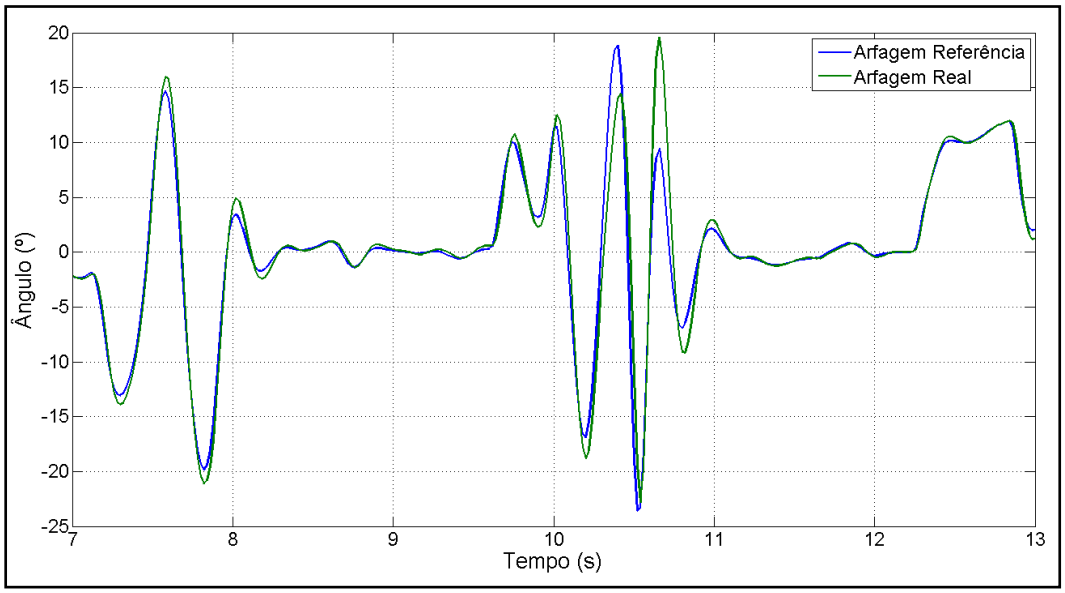

Figura 29 - Comparativo de arfagem referencial e saída – simulação (Autor)

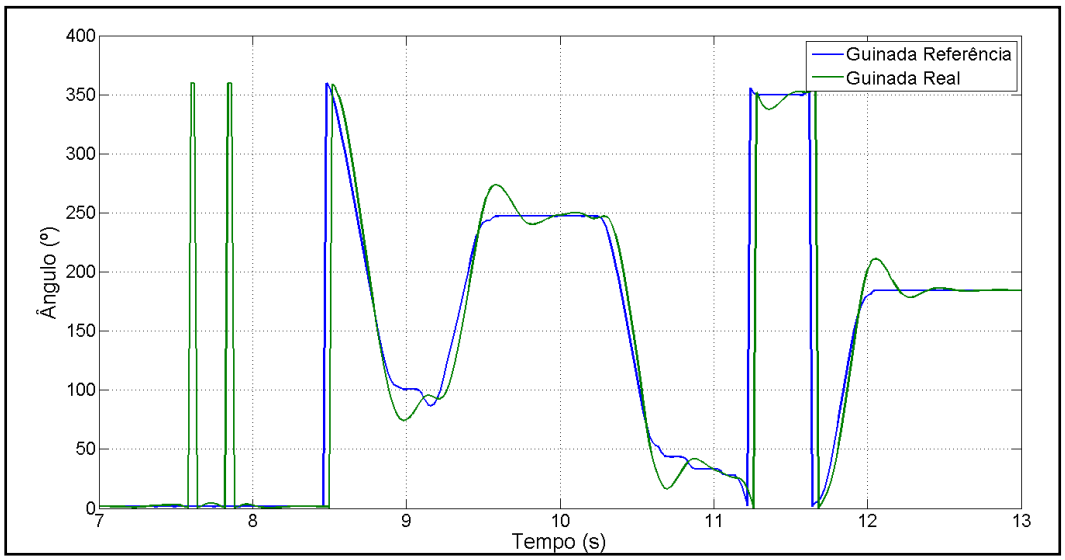

Figura 30 - Comparativo de guinada referencial e saída – simulação (Autor)

As figuras 26, 27 e 28 mostram as curvas de resposta do sistema com os parâmetros PID originais do *software*, com Kp = 0,135, Ti = 0,090 e Td = 0,0036. Apesar das condições ideais consideradas na simulação, podem ser observadas algumas falhas no controlador, como excesso de sobressinal em algumas situações, como mostram as figuras 28, 29 e 30. Tais fatores demonstram a oportunidade de melhoria do controle deste sistema. O indicador de performance IAE medido neste sistema foi de 146,60 para um teste de resposta ao degrau em rolagem, mostrado na figura 31.

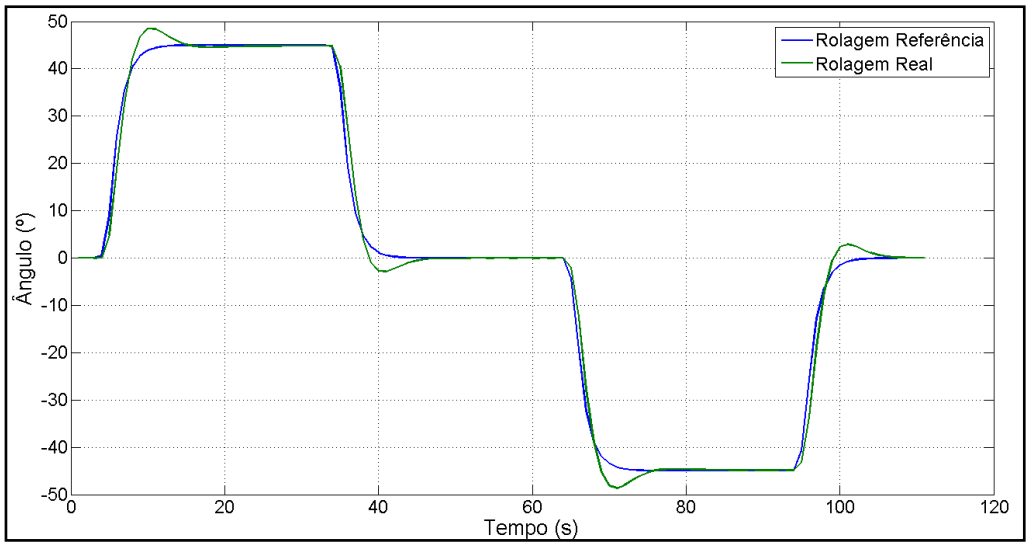

Figura 31 – Resposta ao degrau com parâmetros PID originais – simulação (Autor)

Aplicando no controlador os parâmetros de PID obtidos para rolagem e arfagem pelo método ITAE de análise de curva de reação, as respostas obtidas tiveram boa performance, conforme o esperado para o método, tendo como parâmetros PID os valores Kp = 0,5089, Ti = 0,2516 e Td = 0,0538. O sistema teve resposta ao degrau com indicador IAE = 27,45 (Redução de IAE de 81,2% em relação ao sistema original), conforme mostrado na figura 32, além de trazer um resultado satisfatório em testes simulando uma situação de voo real, ou "voo livre", como pode ser visto nas figuras 33 e 34. É importante ressaltar que o voo livre realizado em cada uma das etapas (antes e depois das sintonias PID) não foram exatamente iguais, por se tratarem de voos em controle manual, por isso não podem ser diretamente comparados. Apesar disto, algumas mudanças importantes podem ser notadas entre eles, conforme são observadas nesta seção.

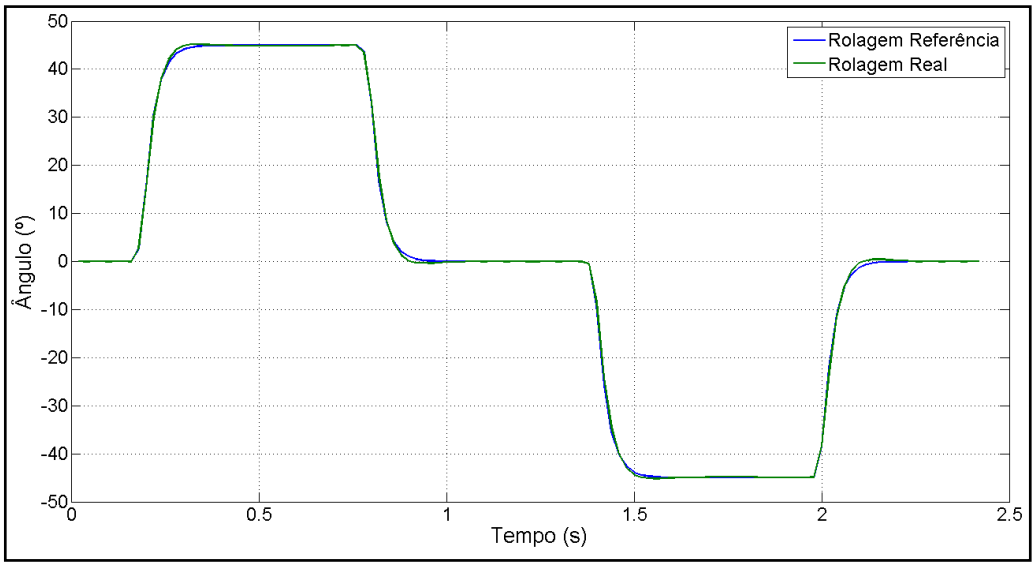

Figura 32 – Resposta ao degrau com parâmetros ITAE – simulação (Autor)

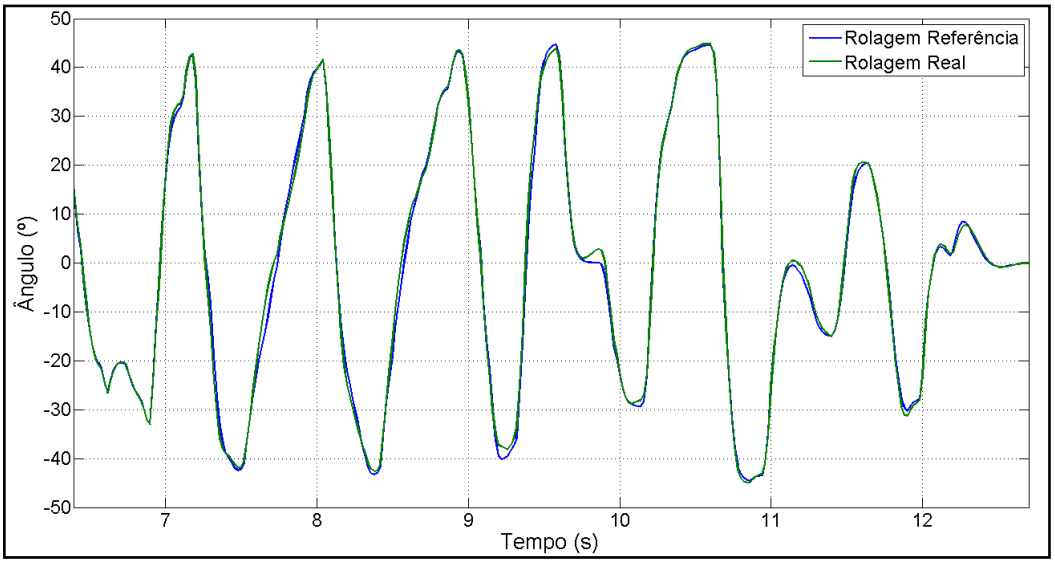

Figura 33 – Rolagem em voo livre com parâmetros ITAE – simulação (Autor)

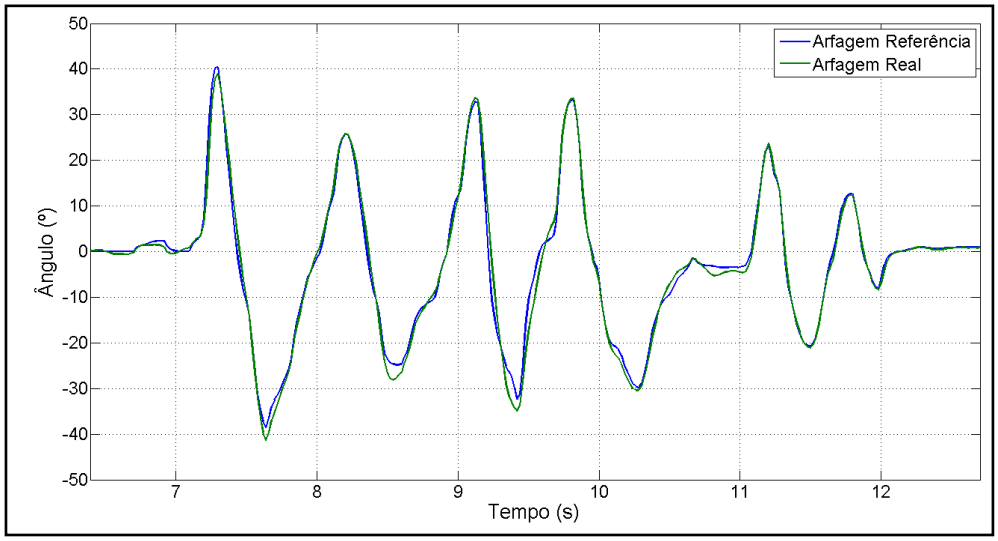

Figura 34 – Arfagem em voo livre com parâmetros ITAE – simulação (Autor)

Apesar de todas as melhorias observadas em resposta ao degrau e voo livre, como redução de sobressinal e do índice de erro medido, a aplicação deste método de sintonia traz alguns fatores os quais requerem maior observação, como por exemplo o alto valor da constante derivativa Td obtido (cerca de 15 vezes maior que o original), o qual, caso seja aplicado em sistema real, pode gerar graves problemas na saída devido a oscilações causadas por erros na leitura dos sensores, o que é muito comum em sistemas do tipo.

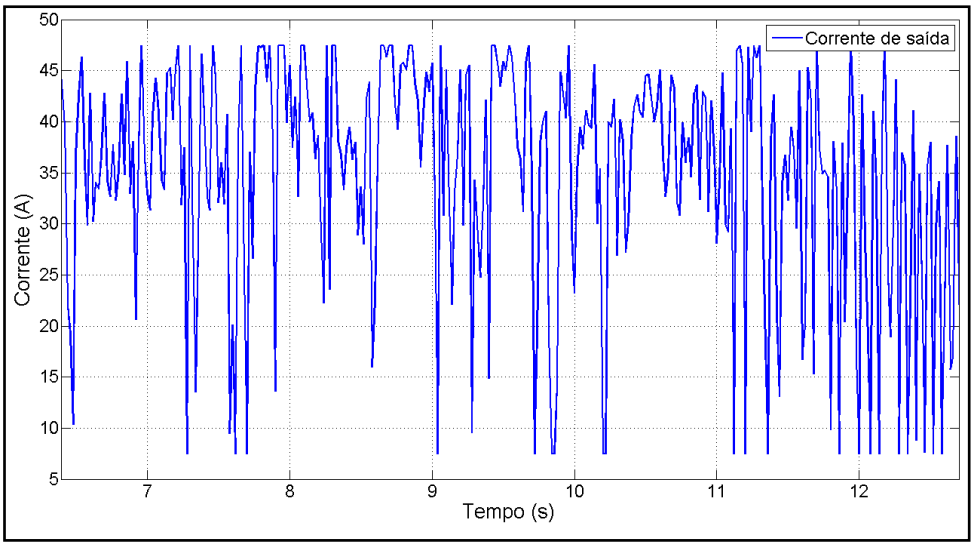

Figura 35 – Corrente em voo livre com parâmetros ITAE – simulação (Autor)

Outro fator importante pode ser observado na figura 35, a qual mostra a corrente total de saída durante a simulação de voo livre. Esta teve fortes oscilações durante todo o teste, além de atingir o limite máximo imposto pelo software de simulação durante diversos momentos. Em sistema real, tal situação poderia causar graves danos à bateria, ao circuito e aos motores do drone, podendo sobrecarregar os componentes durante o voo.

Com estes resultados pode-se concluir que este tipo de método possivelmente não seria o mais adequado para esta aplicação, inclusive pelos riscos de sua obtenção em sistemas reais (voo muito instável em malha aberta, e risco de dano dos motores, mesmo em ambiente controlado, devido à alta corrente).

O método *Twiddle*, por outro lado, mostrou performance satisfatória em diversos aspectos, ao longo dos testes realizados, conforme o esperado. Sua resposta a degrau de entrada minimizou consideravelmente o índice de erro IAE medido, tendo resultado similar ao método anterior, com redução de 78,8% em relação aos parâmetros originais do *software*, como pode ser visto na figura 36.

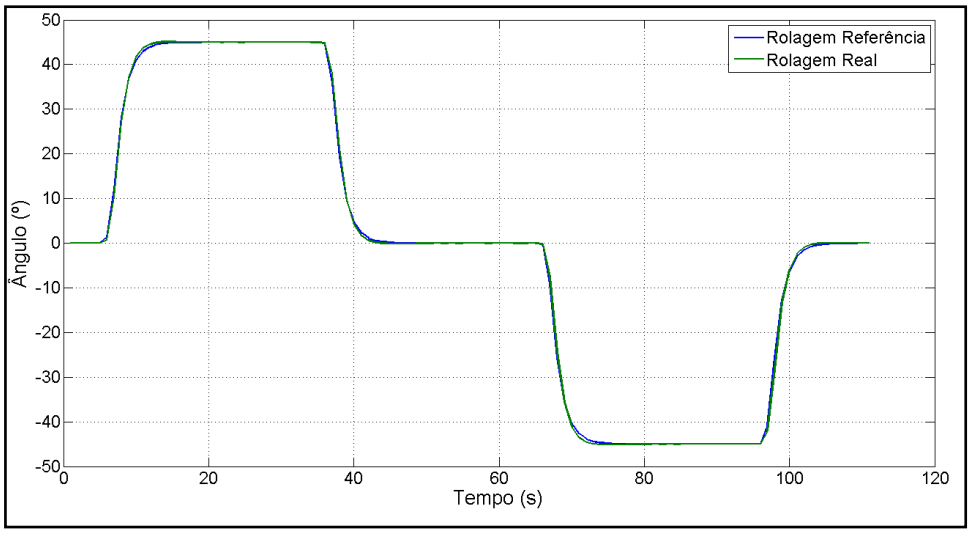

Figura 36 – Resposta ao degrau (Método *Twiddle*) – simulação (Autor)

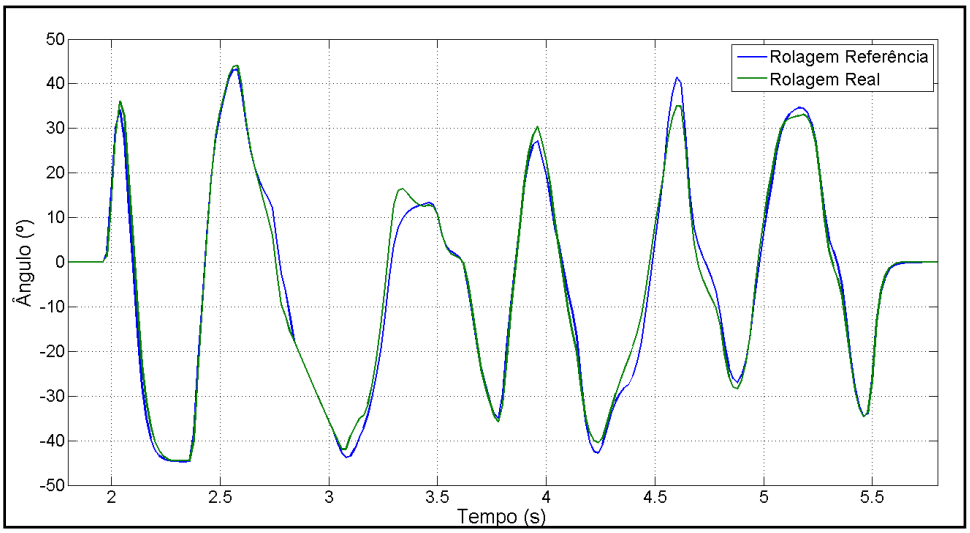

Figura 37 – Rolagem em voo livre com parâmetros *Twiddle* – simulação (Autor)

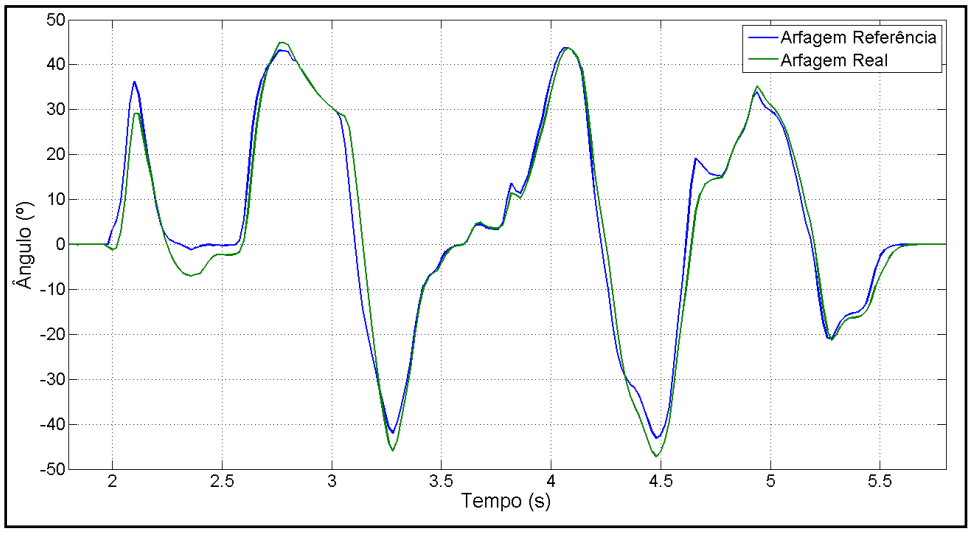

Figura 38 – Arfagem em voo livre com parâmetros *Twiddle* – simulação (Autor)

As figuras 37 e 38 mostram os resultados obtidos de rolagem e arfagem durante voo livre, utilizando os parâmetros obtidos pelo método, sendo Kp = 0,435, Ti = 0,085 e Td = 0,0036. Como pode ser visto, as respostas tiveram melhora significativa em relação ao sistema original, tendo erros e sobressinais reduzidos, mesmo em testes com manobras de angulações mais elevadas. Além disso, a corrente de saída total dos 4 motores se manteve dentro do limite considerado seguro para os modelos de motores utilizados na montagem do drone.

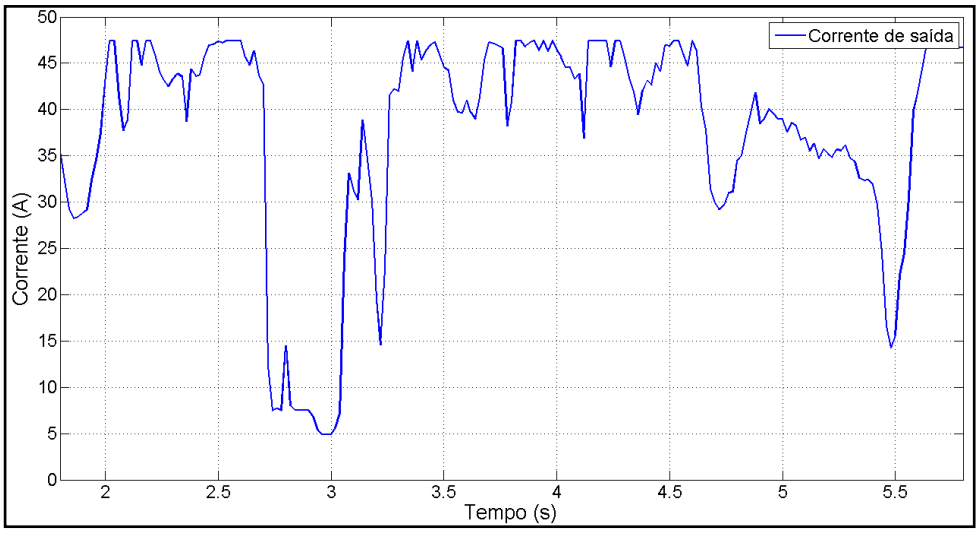

Figura 39 – Corrente de saída em voo livre com parâmetros *Twiddle* – simulação (Autor)

4.3 – RESULTADOS PLANEJAMENTO DE TRAJETÓRIA

O algoritmo de campos potenciais trouxe resultados muito próximos do esperado. Após algumas sequências de testes para ajustar os parâmetros de raio de detecção de obstáculos, delta de distância dos sub-objetivos e constantes de forças de atração e repulsão, o drone em simulação foi capaz de chegar ao seu objetivo principal em curto espaço de tempo, realizando desvio dos obstáculos pontuais sem necessidade de movimentos bruscos para tal. Os testes de pré-ajuste dos parâmetros do algoritmo, essenciais para o funcionamento correto deste, reforçam a necessidade de execução experimental do programa em simulação previamente ao teste em voo real.

A figura 40 ilustra o algoritmo durante execução de um teste simples, porém capaz de mostrar o potencial do mesmo, contendo um ponto objetivo final por volta de 500 metros de distância do ponto inicial da aeronave e dois obstáculos pontuais em lados opostos ao longo de seu percurso. Na figura podem ser vistos cada um destes pontos de interesse, além do sub-objetivo gerado a cada iteração do algoritmo, da posição do drone no momento da execução, e da trajetória realizada até o momento, representado pelo traço roxo na imagem.

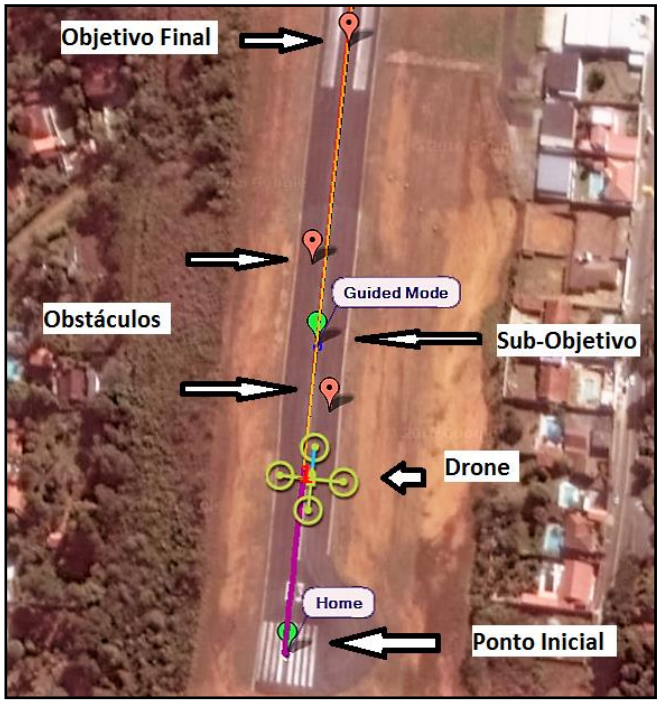

Figura 40 – Algoritmo Campos Potencias em execução – simulação (Autor)

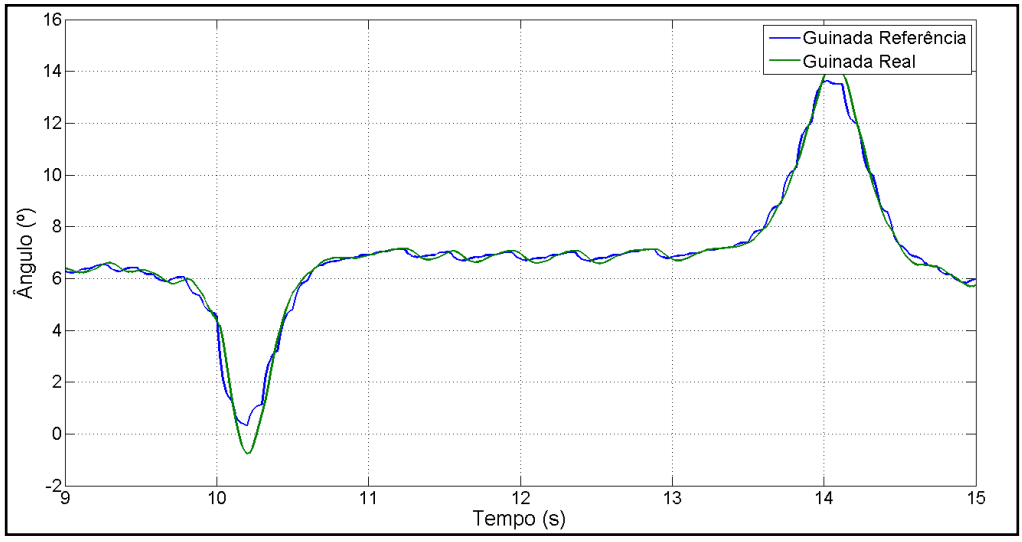

Figura 41 – Guinada durante passagem por obstáculos – simulação (Autor)

O voo de teste do algoritmo gerou o gráfico de reação de guinada mostrado na figura 41. Por volta da marca de 10 segundos de execução, o *drone* se aproxima do raio de detecção do primeiro obstáculo, situado na sua direita, causando o desvio de angulação no sentido anti-horário, observado no primeiro pico do gráfico. Após a saída da área de influência do primeiro obstáculo e aproximação do segundo (por volta dos 13,5 segundos), à esquerda do caminho, o programa provoca o segundo desvio na trajetória, este no sentido horário, gerando o segundo pico da figura.

Figura 42 – Trajetória final do algoritmo – simulação (Autor)

O programa chega ao fim da sua execução ao se aproximar do objetivo final, dentro de um raio de 5 metros, iniciando neste momento seu pouso automático em segurança no local final. A trajetória final obtida na execução pode ser vista na figura 42, onde apesar de pequenos, os desvios de trajetória em cada obstáculo podem ser observados na curvatura do traçado mostrado.

CAPÍTULO 5: CONCLUSÕES

Neste trabalho, foi construído de maneira própria um robô móvel aéreo, sendo que suas etapas de desenho de peças, montagem e escolha de componentes foram feitas inteiramente pelo projeto, visando a criação de um protótipo de multirotor de baixo custo construído a partir de peças feitas em impressão 3D. Além disso, buscou-se melhorar os resultados de voo obtidos encontrando um método adequado de sintonia de PID para o *drone* através de simulações, e por fim, foi implementado um algoritmo em Python de geração de trajetória na plataforma *Ardupilot*, com objetivo de oferecer maior grau de autonomia à aeronave construída, permitindo a esta futuras aplicações autônomas.

Nos voos experimentais, o *drone* construído teve performance satisfatória, mostrando capacidade de realizar voos simples de curta distância e com manobras de leve intensidade, tendo visualmente boa estabilidade para tal, apesar de mostrar dificuldades em alguns momentos, principalmente ao realizar movimentos mais extremos, causando parcialmente perda de controle nestes momentos.

Ao analisar as curvas de telemetria, as falhas de voo puderam ser melhor observadas, mostrando sobressinal na resposta da maior parte das entradas do controle, além de vibrações inclusive durante voo estacionário, indicando, portanto, possíveis melhorias não somente no controle PID utilizado, mas também na estrutura montada. A estrutura pôde ser beneficiada com melhorias na fixação dos motores, inserindo borrachas para absorver a vibração destes, mudanças nos braços da estrutura, fazendo-os mais rígidos e menos sujeitos a torções durante voo, e troca dos modelos de motores utilizados por outros de maior capacidade de empuxo.

Analisando os métodos de sintonia PID utilizados, pôde-se observar que o método ITAE trouxe, a princípio, excelentes resultados em simulação, trazendo grande diminuição dos índices de performance medidos. Porém, o método não leva em consideração fatores importantes para a aplicação deste em cenário real, como os níveis de corrente dos motores necessários para atingir a performance desejada pelos parâmetros obtidos, fazendo com que estes só possam ser testados no modelo real caso este esteja em ambiente controlado, mantendo o drone fixo em uma estrutura para testes, e contendo elementos de proteção nos circuitos de acionamento dos motores e alimentação do protótipo.

O método *Twiddle*, por sua vez, teve como resultado final uma performance um pouco inferior ao método ITAE, porém mostrou que possivelmente este resultado poderia ser

alcançado pelo modelo real, sem causar fortes oscilações ou valores excessivos nas correntes de acionamento dos motores. Este método, por ser composto de tentativa-e-erro, possivelmente poderia ser facilmente aplicado ao *drone* construído, sem a necessidade de ambiente controlado para os testes, pois tendo mudanças graduais dos parâmetros obtidos em cada iteração, cada resultado obtido poderia ser analisado com mais cuidado, observando os possíveis riscos antes que estes tomassem proporções prejudiciais aos componentes do veículo.

O algoritmo de campos potenciais obteve performance próxima ao esperado, tendo o *drone* em simulação sido capaz de se mover até seu objetivo, realizando desvios suaves aos obstáculos detectados em seu caminho. Entretanto, o algoritmo utilizado não é capaz de superar o possível problema de mínimos locais, conforme descrito na seção 2. Porém este problema poderia ser contornado adicionando uma terceira dimensão, adicionando leituras de altitude ao seu funcionamento, permitindo assim contornar obstáculos de mínimo local alterando a altura do voo.

5.1 - SUGESTÕES PARA TRABALHOS FUTUROS

Conforme apontado anteriormente, o modelo construído poderia ser amplamente beneficiado por modificações e revisões em sua estrutura a qual permitissem melhorias significativas da estabilidade de voo deste. Além disto, seria imprescindível em trabalhos futuros, que tanto os métodos de sintonia PID, quanto o algoritmo de campos potenciais criado, pudessem ir além das simulações e serem testados no *drone* real.

O próprio algoritmo de campos potenciais tem espaço para muitas melhorias, visando aumentar ainda mais o grau de autonomia do robô móvel criado. Além da possibilidade de contorno dos mínimos locais, indicada anteriormente, o drone poderia contar com maneiras mais sofisticadas de obtenção de localização dos obstáculos, sem depender do mapeamento do local, conforme foi implementado no algoritmo originalmente. Sensores como sonares ou lasers poderiam ser instalados a fim de se obter leituras em tempo real dos obstáculos, e com maior precisão.

Outra sugestão seria a possível melhoria do subsistema de processamento do *drone* com a adição de um microprocessador, ampliando as possibilidades de aplicações do robô, permitindo uso em mapeamentos de terrenos ou busca de objetivos, por exemplo. O

microprocessador poderia trabalhar em alto nível, recebendo informações de estações terrestres ou realizando processamento de imagem do ambiente através de câmeras instaladas no veículo, e transmitindo comandos correspondentes ao microcontrolador utilizado.

REFERÊNCIAS

1. BAK, D. Rapid prototyping or rapid production? 3D printing processes move industry towards the latter. **Assembly Automation**, v. 23, n. 4, p. 340-345, 2003.
2. BBC. Police use drones at Birmingham City clash with Aston Villa. Disponível em: <http://www.bbc.com/news/uk-england-birmingham-37812163> Acesso em: 30 de outubro de 2016.
3. BOM, R. P. Alternativa de reciclagem de resíduos de espuma rígida de poliuretano com ABS. **Matéria (Rio J.)**, Rio de Janeiro, v. 13, n. 2, p. 388-397, Junho, 2008.
4. CARDOZO, E; PINHEIRO, P; OLIVI. L. **IA368-W Métodos Estocásticos em Robótica Móvel** (Notas de aulas da disciplina de Métodos Estocásticos em Robótica Móvel). Campinas, 2014. Disponível em <http://143.106.148.168:9080/Cursos/IA368W/parte1.pdf> Acesso em: 25 de Outubro de 2016.
5. COELHO, L. S; ALMEIDA, O. M.; COELHO, A. A. R.. Projeto e estudo de caso da implementação de um sistema de controle nebuloso. **Sba Controle & Automação**, Campinas, v. 14, n. 1, p. 20-29, Mar. 2003.
6. CORKE, P. **Robotics, vision and control: fundamental algorithms in MATLAB**. 1st Edition. Brisbane: Springer, 2011.
7. DIÁRIO DIGITAL. Alunos do instituto federal desenvolveram um drone para ajudar a conter pragas nas lavouras. Disponível em <http://www.diariodigital.com.br/videos/alunos-do-instituto-federal-desenvolveram-um-drone-para-ajudar-a/10743/> Acesso em: 30 de outubro de 2016.
8. FERNANDEZ, G. C. et al. Robotics, the new industrial revolution. **IEEE Technology and Society Magazine**, v. 31, n. 2, p. 51-58. 2012.
9. FRANCHI, C. M. **Controle de Processos Industriais**. 1ª Edição. São Paulo: Érica, 2011.
10. GOLDSTEIN, R. et al. Beohawk: Autonomous Quadrotor. **Symposium on Indoor Flight Issue**, s/v, s/n, p. 1-11, 2012.
11. HASELAGER, W. F G. Robotics, philosophy and the problems of autonomy. **Pragmatics & Cognition,** v. 13, n. 3, p. 515-532, 2005.

12. HOWELL, E. Opportunity Mars Rover Marks 12 Years on Red Planet. **Space.** Disponível em <http://www.space.com/31735-opportunity-rover-12-years-mars.html> Acesso em: 30 de outubro de 2016.
13. HSIUN, C. M; CHEN, C. K. Aerodynamic characteristics of a two-dimensional airfoil with ground effect. **Journal of Aircraft,** v. 33, n. 2, p. 386-392, March-April, 1996.
14. HUANG, H. et al. Autonomy measures for robots. **International Mechanical Engineering Congress, American Society of Mechanical Engineers**, v. 72, n. 2, p. 1-7, November, 2004.
15. INGO, B. V. Quadcopter inkl. APM 2.6 Flight Controller selber bauen für nur ca. 200 Euro. Disponível em: http://hobbyking-fans.de/quadtcopter-inkl-apm-2-6-flight-controller-selber-bauen-fuer-nur-ca-200-euro/ Acesso em: 31 de outubro, 2016.
16. KEMPF, A. O. **Avaliação de desempenho de malhas de controle**. 2003. Dissertação. (Mestrado em Engenharia) - Universidade Federal do Rio Grande do Sul. Escola de Engenharia. Programa de Pós-Graduação em Engenharia Química. Porto Alegre, RS. 2003.
17. KOREN, Y; BORENSTEIN, J. Potential field methods and their inherent limitations for mobile robot navigation. **Robotics and Automation, Proceedings,** v. 2, p. 1398-1404. 1991.
18. NEWTON, C. Facebook Takes Flight. **The Verge**. Disponível em: <http://www.theverge.com/a/mark-zuckerberg-future-of-facebook/aquila-drone-internet> Acesso em: 30 de outubro de 2016.
19. NORVIG, P; THRUN, S. "Udacity – Introdução à inteligência artificial". Disponível em: <https://br.udacity.com/course/intro-to-artificial-intelligence--cs271/>. Acesso em: 05 out. 2016
20. OGATA, K. **Engenharia de Controle Moderno.** 3ª Edição. Rio de Janeiro: LTC, 1998.
21. OLIVI. L. **Geração de Trajetórias** (Notas de aulas da disciplina de Manipuladores Robóticos). Juiz de Fora, 2014. Disponível em <http://www.dropbox.com/sh/tc41end0qow1whz/AACbRkhpW2jwJDyyYdANCzRFa?dl=0> Acesso em: 25 de Outubro de 2016.
22. OLIVI. L. **Introdução à Robótica Móvel** (Notas de aulas da disciplina de Robótica Móvel). Juiz de Fora, 2015. Disponível em <http://www.dropbox.com/sh/qg2x4r2rqnx6v1o/AABdcJX607pp87kTbePRXPuza> Acesso em: 25 de Outubro de 2016.

23. PAIM, C. C. **Técnicas de controle aplicadas a um atuador hidráulico**. 1997. Dissertação. (Mestrado em Engenharia Elétrica) - Universidade Federal de Santa Catarina, Centro Tecnológico. Florianópolis, SC. 1997.
24. PIRES, A. G; CHAIMOWICZ, L. Localização Cooperativa, Descentralizada e Baseada em Marcos Dinâmicos em Enxames de Robôs Móveis. **Simpósio Brasileiro de Robótica (SBR/LARS)**, s/v, s/n, p. 1-10, 2012.
25. RODRIGUES, B. S. **Avaliação de desempenho de malhas de controle**. 2010. Trabalho de Conclusão de Curso (Graduação em Engenharia de Controle e Automação) – Universidade Federal de Ouro Preto. Ouro Preto, MG. 2010.
26. SALICHS, M. A; MALFAZ, M; GOROSTIZA. J. F. Toma de decisiones en robótica. **Revista Iberoamericana de Automática e Informática Industrial RIAI**, v. 7, n. 4, p. 5-16, 2010.
27. SANTINO, R. Drone dobrável pesa apenas 250 gramas e pode revolucionar selfies. **Olhar Digital.** Disponível em <http://olhardigital.uol.com.br/noticia/drone-dobravel-pesa-apenas-250-gramas-e-pode-revolucionar-as-selfies/63054> Acesso em: 30 de outubro de 2016.
28. SARAIVA, F. A. **Métodos de Sintonia em Controladores PID**. 2011. Trabalho de Conclusão de Curso (Graduação em Engenharia de Telecomunicações) – Centro Universitário La Salle – Universalle. Canoas, RS. 2011.
29. SIEGWART, R; NOURBAKHSH, I. R; SCARAMUZZA, D. **Introduction to autonomous mobile robots.** 2nd Edition. London, England: The MIT press, 2011.
30. VANIAN, J. The Multi-Billion Dollar Robotics Market Is About to Boom. Fortune. Disponível em: <http://fortune.com/2016/02/24/robotics-market-multi-billion-boom/> Acesso em: 30 de outubro de 2016.
31. VILANOVA, R; ALFARO, V. M. Control PID robusto: Una visión panorámica. **Revista Iberoamericana de Automática e Informática Industrial RIAI**, v. 8, n. 3, p. 141-158. Julio/Septiembre, 2011.
32. WEBSTER, N. UAE providing life-saving drone delivery service in Africa. **The National**. Disponível em: <http://www.thenational.ae/uae/uae-providing-life-saving-drone-delivery-service-in-africa> Acesso em: 30 de outubro de 2016.

ANEXO - A

Anexo A – Algoritmo em Python da aplicação do método de Campos Potencias (Autor)

```
import sys
from math import*
import clr
import time
import MissionPlanner #.Utilities import *

# incluir a classe Utilities
clr.AddReference("MissionPlanner.Utilities")

takeoff_throttle = 1550
alt = 15

def setup_rc():
    print("Colocando Canais em Neutro")
    for chan in range(1,9):
        Script.SendRC(chan, 1500, False)
    Script.SendRC(3, Script.GetParam('RC3_MIN'), True)
    Script.Sleep(5000)
    print("Pronto")

    return True

def arm_motors():
    "Arming motors"
    print('APM: ARMING MOTORS')
    print("Colocando modo em Stabilize")
    Script.ChangeMode("STABILIZE")
    Script.SendRC(5, 1750, True)
    print("Pronto")
    print('Colocando Throttle no minimo')
    Script.SendRC(3,1000,True)
    print("Pronto")
    print('Set Yaw')
    Script.SendRC(4,2000,True)
    print("Pronto")
    print('Waiting for Arming Motors')
    cs.messages.Clear()
    Script.Sleep(4000)
    print("Pronto")
    print('Reset Yaw')
    Script.SendRC(4,1500,True)
    print("Done")

    return True


def wait_altitude(alt_min, alt_max, timeout=30):
    climb_rate = 0
    previous_alt = 0
    #aguardar por uma dada faixa de altitude
    tstart = time.time()
    print("Aguardando altitude entre %u e %u" % (alt_min, alt_max))
    while time.time() < tstart + timeout:
        climb_rate =  cs.alt - previous_alt
        previous_alt = cs.alt
        if cs.alt >= alt_min and cs.alt <= alt_max:
           print("Altitude atingida:%u"%(cs.alt))
           print("Altitude OK")
           return True
    print("Falha em alcançar faixa de altitude")
    return False
```

```
def takeoff(alt_min=alt): #0.5
   print("Iniciando decolagem")
   print("Prestes a decolar")
   failed_status = False

   print("Certificando que estamos em modo stabilise")
   Script.ChangeMode("STABILIZE")
   Script.SendRC(5, 1750, True)
   print("Pronto")
   print("Aumentando o throttle")
   Script.SendRC(3, takeoff_throttle, True)

   print("Processando controle de altitude")

   found_alt = False

   if (cs.alt < alt_min):
        found_alt = wait_altitude(alt_min, alt_min + 1)

#Agora temos a altitude que desejamos mater

   if found_alt:
        Script.ChangeMode("AltHold")
        #Script.SendRC(5, 1000, True)

        print("Colocando throttle em 40-60% para manter altitude")
        Script.SendRC(3, 1450, True)

        print("Decolagem completa")
   else:
        print("Decolagem falhou em alcançar altidude adequada")
        print("Iniciar procedimento de controle de erro")

        failed_status = True

   return failed_status


def hover(hover_throttle = 1500):
    print("Iniciando modo Pairar")
    print("Inserindo throttle de pairar")
    Script.SendRC(3, hover_throttle, True)

    print("Pronto")
    print("Pairando por 3 segundos")
    Script.Sleep(3000)
    print("Pairar completo")

def land():
#aterrissar o drone'''
    print("INICIANDO ATERRISSAGEM")
    print("Mudando Modo para LAND")
    Script.ChangeMode("LAND") # land mode
    Script.SendRC(5, 1300, True)
    print("Pronto")
    Script.WaitFor('LAND', 5000)
    print("Entrou em modo LAND")
    ret = wait_altitude(-5, 10)
    print("Aterrissagem: ok= %s" % ret)
    print("Aterrissagem completa")
    return ret

def disarm_motors():
#desarmar motores'''
    print("Desarmando motores")
    print("Mudando modo para STABILIZE")
    Script.ChangeMode("STABILIZE") # stabilize mode
    print("Pronto")
    Script.WaitFor('STABILIZE', 5000)
    Script.SendRC(3, 1000, True)
    Script.SendRC(4, 1000, True)
    Script.WaitFor('DISARMING MOTORS', 5000)
    Script.SendRC(4, 1500, True)
    #mav.motors_disarmed_wait()
    print("MOTORES DESARMADOS OK")
    return True


def distance(a, b): #calcula distancia em metros entre duas coordenadas em radianos
    t1 = sin(a[0]) * sin(b[0])
    t2 = cos(a[0]) * cos(b[0])
    t3 = cos(a[1] - b[1])
    t4 = t2 * t3
    t5 = t1 + t4
    rad_dist = atan(-t5/sqrt(-t5 * t5 +1)) + 2 * atan(1)
    dist = ((rad_dist * 3437.74677 * 1.1508) * 1.6093470878864446)*1000
    return dist
```

```
time.sleep(3)

###### Declaração dos parâmetros #######

# Conversão graus/radianos
a = 3.14 / 180
# Raio de detecção de obstáculo do drone
eps = 5
# Constante de repulsão
Krep = 750
# Força de repulsão
Frep = [0, 0]
# Constante de atração
Katt = 0.05
# Força de atração
Fatt = [0, 0]
# Delta de variação de objetivo
delta = 0.00001
# Força resultante
Ftot = [0, 0]


#Preparando RC para delocagem
setup_rc()
# Obter sinal GPS do drone, armar, delocar e pairar
while cs.lat == 0:
    print 'Waiting for GPS'
    Script.Sleep(1000)
print 'Got GPS'
arm_motors()
takeoff()
hover()



# Iniciando algoritmo Campos Potenciais
print 'Iniciando Missão'

# Criando waypoint
goal = MissionPlanner.Utilities.Locationwp()
# Latitude do alvo
goallat = -21.7955015220807
# Longitude do alvo
goallng = -43.3865118026733
# Altitude relativa do alvo
goalalt = 15

# Posição GPS do alvo em radianos
cible = (goallat*a, goallng*a)

# Nº de obstáculos
n = 2
Obst = [[0 for x in range(2)] for y in range(n)]
# Posições GPS dos obstáculos
Obst[0][0], Obst[0][1] = (-21.7972548031156 * a, -43.3866539597511 *a)
Obst[1][0], Obst[1][1] = (-21.7963881271037 * a, -43.3867263793945 * a)
# Vetor de distâcias do drone para cada obstáculo
distObst = [0 for x in range(n)]
# Vetor sub-objetivo
g = [0, 0]

# Inserindo objetivo do drone no alvo final
MissionPlanner.Utilities.Locationwp.lat.SetValue(goal,goallat)
MissionPlanner.Utilities.Locationwp.lng.SetValue(goal,goallng)
MissionPlanner.Utilities.Locationwp.alt.SetValue(goal,goalalt)
print 'Alvo inserido'
MAV.setGuidedModeWP(goal)
dist = 500


# Iniciando Loop de Execução do Algoritmo - Se encerra ao chegar a 5 metros do alvo
while abs(dist)>5:

    # Posição atual do Drone
    pos = (cs.lat*a,cs.lng*a)
    # Distância atual ao Objetivo Final
    dist = distance(pos, cible)

    # Vetor Força de Atração
    for i in range(2):
        Fatt[i] = -Katt * (pos[i] - cible[i])

    # Procura obstáculos dentro do raio de detecção
    indices = [0 for x in range(n)]
    for i in range(n):
        distObst[i] = distance(pos, Obst[i])
        if distObst[i] < eps:
            indices[i] = 1
```

```
    # Vetor Força de Repulsão
    Frep = [0, 0]
    if indices != [0 for x in range(n)]:
        print "Calculando Frep"
        for x in range(n):
            for y in range(2):
                Frep[y] = Frep[y] + indices[x]*Krep*(1/pow(distObst[x],3))*((1/distObst[x])-(1/eps))*(pos[y]-Obst[x][y])

    # Vetor força resultante
    for i in range(2):
        Ftot[i] = Fatt[i] + Frep[i]

    # Calcula novo sub-objetivo
    for i in range(2):
        g[i] = pos[i] + delta*Ftot[i]*(1/(pow(pow(Ftot[0], 2) + pow(Ftot[1], 2), 0.5)))
    print "Inserindo novo objetivo"

    # Converte coordenadas do sub-objetivo de radianos para graus
    goallat = g[0]/a
    goallng = g[1]/a

    # Insere novo objetivo no Drone
    MissionPlanner.Utilities.Locationwp.lat.SetValue(goal,goallat)
    MissionPlanner.Utilities.Locationwp.lng.SetValue(goal,goallng)
    MissionPlanner.Utilities.Locationwp.alt.SetValue(goal,goalalt)
    MAV.setGuidedModeWP(goal)

    # Mostra distância ao alvo e aos obstáculos
    print dist
    for i in range(n)
        print distObst[i]

    Script.Sleep(500)

# Encerra algoritmo, realiza aterrissagem no alvo e desarma o drone
print "Alvo alcançado. Iniciar aterrissagem."
Script.ChangeMode("LOITER")
land()
disarm_motors()
setup_rc()
```