UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Paulo Roberto Fernandes Tavares

Implementação prática de um procedimento de teste utilizado na avaliação de técnicas de controle aplicadas a um *drone* octocóptero

São Carlos

2020

Paulo Roberto Fernandes Tavares

Implementação prática de um procedimento de teste utilizado na avaliação de técnicas de controle aplicadas a um *drone* octocóptero

Monografia apresentada ao Curso de Engenharia Elétrica com Ênfase em Eletrônica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Marco Henrique Terra

VERSÃO CORRIGIDA

São Carlos

2020

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Tavares, Paulo Roberto Fernandes
T324i Implementação prática de um procedimento de teste utilizado na avaliação de técnicas de controle aplicadas a um drone octocóptero / Paulo Roberto Fernandes Tavares; orientador Marco Henrique Terra. São Carlos, 2020.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2020.

1. Drone. 2. Octocóptero. 3. Vicon. 4. Pyvicon. 5. ESP8266. 6. Autônomo. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Nome: Paulo Roberto Fernandes Tavares

Título: "Implementação prática de um procedimento de teste utilizado na avaliação de técnicas de controle aplicadas a um drone octocóptero"

Trabalho de Conclusão de Curso defendido e aprovado em 26 / 03 / 2020 ,

com NOTA 9,5 (nove , cinco), pela Comissão Julgadora:

Prof. Titular Marco Henrique Terra - Orientador - SEL/EESC/USP

Mestre João Roberto Soares Benevides - Doutorando - SEL/EESC/USP

Prof. Dr. Roberto Santos Inoue - UFSCar

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

Este trabalho é dedicado aos meus pais, Marycleide e Nelson, que me criaram, me cuidaram quando eu não era capaz de fazê-lo, abriram mão de muitas coisas em suas vidas por mim e consumiram uma energia incalculável me proporcionando o possível em sua alçada para que hoje eu fosse quem sou e chegasse onde estou. Sem eles, essa caminhada não seria possível. Na verdade, nenhuma seria.

AGRADECIMENTOS

Agradeço, primeiramente, a Deus pelo dom da vida. Aos meus pais, Marycleide e Nelson, e meus familiares que me criaram, educaram e proveram todos os meios para que eu pudesse me encontrar onde estou hoje. Ao meu primo Dennis, cujos passos eu estou seguindo na Engenharia Elétrica. À Letícia, minha namorada, que me apoiou durante o curso todo e enfrentou a distância comigo. Aos meus amigos Igor, Rafael e Vinícius que trilharam o caminho da graduação ao meu lado, compartilhando conhecimentos e momentos difíceis e descontraídos.

Agradeço à empresa Avibra Indústria Aeroespacial, por ceder as instalações do EATI (Espaço Avibras de Tecnologia e Inovação), bem como os responsáveis pelo espaço, Hélio e Eduardo. Ao Parque Tecnológico de São José dos Campos, especialmente às equipes de administração e segurança, pela liberação do espaço para realização dos voos.

Agradeço ao meu orientador, Prof. Marco Terra, pela oportunidade do trabalho, pela atenção e disponibilidade. Ao meu colega Leonardo, que acompanhou de perto o desenvolvimento do trabalho e compartilhou seus conhecimentos. Aos colegas do laboratório LASI e do departamento, pelo auxílio que me deram quando necessitei. À Escola de Engenharia de São Carlos da Universidade de São Paulo, por promover estrutura, material de pesquisa e professores que permitiram a evolução e conclusão dessa graduação.

“O insucesso é apenas uma oportunidade para recomeçar com mais inteligência.”

Henry Ford

RESUMO

TAVARES, P. R. F. **Implementação prática de um procedimento de teste utilizado na avaliação de técnicas de controle aplicadas a um *drone* octocóptero**. 2020. 72p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2020.

O presente trabalho trata da implementação prática do procedimento de teste proposto por Farçoni (2019)[1]. Nele, o autor propõe um perfil de voo, caracterizado por uma Lemniscata de Gerono, que é utilizado na comprovação teórica de arquiteturas de controle aplicadas a um *drone* octocóptero. Aqui, busca-se tornar real o referenciado procedimento de teste para que, futuramente, as técnicas de controle abordadas por Farçoni possam ser validadas de forma prática. Para tal, são concretizadas, nessa monografia, soluções de posicionamento de precisão com o sistema Vicon®, comunicação sem fio via Wi-Fi® com o módulo ESP8266, compatibilização de dados e protocolos através da biblioteca Pyvicon[2] e execução autônoma do perfil de voo com o *drone* octocóptero, via MAVROS. As análises dos dados de telemetria e posicionamento gerados a cada voo mostraram que os métodos adotados permitiram que o procedimento de teste proposto fosse concluído satisfatoriamente pelo veículo.

Palavras-chave: *Drone*. Octocóptero. Vicon. Pyvicon. ESP8266. Autônomo.

[1] FARCONI, L. B. **Comparative Analysis of Robust Fault-Tolerant Controllers Applied toMulti-rotor Autonomous Aerial Vehicles**. 2019. 228 p. Dissertação (Mestrado em EngenhariaElétrica) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

[2] PYVICON - python 3 wrapper over Vicon DataStream SDK (1.7.0+). Mathieu Garon, 2018.Disponível em: <https://github.com/MathGaron/pyvicon>.

ABSTRACT

TAVARES, P. R. F. **Practical implementation of a test procedure used in the evaluation of control techniques applied to an octocopter drone**. 2020. 72p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2020.

This work deals with the practical implementation of the test procedure proposed by Farçoni (2019)[3]. In it, the author proposes a flight profile, described by a Lemniscate of Gerono, which is used in the theorical validation of control architectures applied to an octocopter drone. This work seeks to make real the cited test procedure in a way that, in the future, it allows the practical validation of the control techniques brought by Farçoni. To achieve this, the present monograph materializes solutions for precise positioning with the Vicon® system, wireless Wi-Fi® communication using the ESP8266 module, protocol and data compatibilization using the Pyvicon[4] library and autonomous execution of the flight profile by the octocopter, using MAVROS. The data analysis from the positioning system and the telemetry acquired in every flight exposed that the methods herein used allowed the vehicle to conclude the proposed test procedure in a satisfactory way.

Keywords: Drone. Octocopter. Vicon. Pyvicon. ESP8266. Autonomous.

[3] FARÇONI, L. **Comparative Analysis of Robust Fault-Tolerant Controllers Applied to Multirotor Autonomous Aerial Vehicles**. 2019. 228 p. Dissertação(Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

[4] PYVICON - python 3 wrapper over Vicon DataStream SDK (1.7.0+). Mathieu Garon, 2018.Disponível em: <https://github.com/MathGaron/pyvicon>.

LISTA DE FIGURAS

Figura 1 – Curva Lemniscata de Gerono com parâmetro $\alpha = 1$ 20
Figura 2 – *Drone* octocóptero 26
Figura 3 – Controle remoto Spektrum DX7s 26
Figura 4 – Espaço utilizado para voos internos com esquema de adaptação para fixação das câmeras Vicon® MX T40-S 28
Figura 5 – Representação de um volume útil arbitrário 29
Figura 6 – Bastão para calibração - Vicon Active Wand® 29
Figura 7 – Procedimento de definição de objeto virtual para *drone* 30
Figura 8 – Esquema de conexões do sistema Vicon® 30
Figura 9 – Placa de desenvolvimento LoLin com módulo Wi-Fi® ESP8266 31
Figura 10 – Captura de tela da página de configuração da rede Wi-Fi® do módulo ESP8266 32
Figura 11 – Console e mapa do controlador MAVProxy com localização padrão 35
Figura 12 – Comandos de inicialização da biblioteca Pyvicon e respectivas respostas contendo localização padrão do controlador 35
Figura 13 – Tela do controlador de voo indicando parâmetros em conformidade 36
Figura 14 – Esquema completo de conexões entre equipamentos e veículo 37
Figura 15 – Lemniscata de Gerono - Dimensões usadas na simulação de (FARCONI, 2019) 38
Figura 16 – Lemniscata de Gerono - Dimensões para voo real 39
Figura 17 – Diagrama de blocos com fluxo de dados da configuração completa usada em voos autônomos 40
Figura 18 – Modelo de *drone* octocóptero com numeração arbitrária das hélices 41
Figura 19 – Erros de predição normalizados do filtro de Kalman (*innovations*) do voo preliminar 44
Figura 20 – Dados em tempo real do controlador de voo 45
Figura 21 – Perfil de voo - Primeiro voo com sistema Vicon® 46
Figura 22 – Dados de rolagem, arfagem e guinada do *drone* durante o primeiro voo com sistema Vicon® - Comparativo desejado e real 47
Figura 23 – Coordenadas normalizadas geradas pelo sistema de posicionamento 48
Figura 24 – *Drone* em pleno voo no galpão da Figura 4a 49
Figura 25 – Perfil de voo Lemniscata - Comparativo comandado e executado 50
Figura 26 – Perfil de voo Lemniscata - Comparativo comandado e executado - Planos ortogonais 51
Figura 27 – Altitude do veículo durante trajetória - Comparativo barômetro e GPS (sistema Vicon®) 51
Figura 28 – Dados de rolagem, arfagem e guinada do *drone* durante o voo com perfil Lemniscata 52

Figura 29 – PWM comandado ao rotores - Numeração Figura 18 53
Figura 30 – Dados da bateria do *drone* durante o voo com perfil Lemniscata 54
Figura 31 – Destaque em breve falha na trajetória executada com perfil Lemniscata em 60 segundos 54
Figura 32 – Altitude relativa em relação à origem e consumo de corrente sob mesma escala de tempo 55
Figura 33 – Perfil de voo Lemniscata - Comparativo comandado e executado - Execução 120 segundos 56
Figura 34 – Perfil de voo Lemniscata - Comparativo comandado e executado - Planos ortogonais - Execução 120 segundos 57
Figura 35 – Perfil de voo Lemniscata - Comparativo comandado e executado - Execução 90 segundos 57
Figura 36 – Perfil de voo Lemniscata - Comparativo comandado e executado - Planos ortogonais - Execução 90 segundos 58
Figura 37 – Diagrama do circuito do kit de desenvolvimento LoLin ESP8266 65

SUMÁRIO

1 INTRODUÇÃO 19
1.1 Lemniscata de Gerono 20
1.2 Alteração de performance de rotores em configuração coaxial . . 21

2 OBJETIVOS 23
2.1 Objetivo principal 23
2.2 Objetivos específicos 23
2.3 Objetivo secundário 23

3 DESENVOLVIMENTO 25
3.1 Materiais e métodos 25
3.1.1 Veículo e controladores 25
3.1.2 Configuração computacional 26
3.1.3 Sistema de Posicionamento Vicon® 27
3.1.3.1 Posicionamento e fixação das câmeras 27
3.1.3.2 Calibração 27
3.1.3.3 Definição de objetos 29
3.1.4 Módulo Wi-Fi® ESP8266 29
3.1.5 Biblioteca Pyvicon 32
3.1.5.1 Controlador MAVProxy 34
3.1.5.2 Inicialização da biblioteca 34
3.1.6 Perfil de voo 37
3.1.6.1 Lemniscata de Gerono 37
3.1.6.2 Comando do perfil 38
3.1.7 Filtro de Kalman 39
3.1.7.1 Erro de predição - *Innovation* 40
3.1.8 Contribuição para o modelo de simulação do *drone* 41
3.2 Resultados e discussão 43
3.2.1 Voo preliminar 43
3.2.2 Primeiro voo 45
3.2.3 Voo autônomo 48
3.2.3.1 Análise comparativa - duração do voo 55
3.2.4 Contribuição para o modelo de simulação do *drone* 56

4 CONCLUSÃO 59
4.1 Trabalhos futuros 60

REFERÊNCIAS 61

ANEXOS 63

ANEXO A – DIAGRAMA DO CIRCUITO DO MÓDULO WI-FI 65

ANEXO B – CÓDIGO EM C++ PARA EXECUÇÃO DAS TRAJETÓRIAS AUTÔNOMAS 67

1 INTRODUÇÃO

Em pesquisa realizada como dissertação de mestrado, (FARCONI, 2019) investigou cinco arquiteturas de controle submetidas a certas configurações diferentes acerca de suas robustez e performance. Essas análises levam em conta a aplicação de tais técnicas de controle em um *drone* octocóptero sujeito a falhas nos rotores, a fim de se verificar como as diferentes arquiteturas se comportam nesse tipo de situação. É proposto, também, um perfil de voo para o veículo, caracterizado por uma Lemniscata de Gerono (LAWRENCE, 1972), durante o qual ocorreriam as falhas de motores citadas. Para a comprovação, o autor realiza uma serie de simulações computacionais, que contam com o modelo matemático do veículo, e que implementam cada solução de controle analisada para o guiamento do *drone*, submetendo-o ao perfil proposto. Através dos resultados obtidos por meio de tais simulações, as conclusões são tiradas e a investigação dos controladores é concluída.

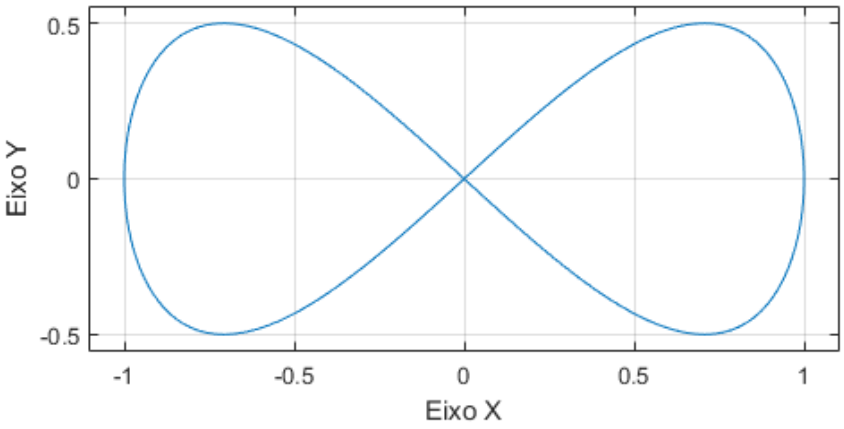
Ainda em (FARCONI, 2019), é apresentado um VANT (Veículo Aéreo Não Tripulado) real, incluindo etapas de seu processo de montagem, cujo propósito é a validação prática do estudo realizado pelo autor.

Visando dar continuidade na pesquisa desenvolvida por (FARCONI, 2019), esse trabalho concentra seus esforços exclusivamente na implementação prática do procedimento de teste trazido pelo autor na forma de simulação computacional. Procedimento esse que conta com a execução, por parte do veículo, do perfil de voo caracterizado pela Lemniscata, desconsiderando-se qual controlador se encontra regendo o *drone*. Com isso, tem-se em mente dar um importante passo na validação prática da dissertação referenciada, uma vez que antes mesmo da implementação de qualquer controlador (que não será abordada no presente trabalho), faz-se necessário ter em mãos o teste real a ser cumprido.

Contudo, o maior desafio para efetivar tal avanço reside no método de aquisição de posição utilizado. Isso vem em decorrência da imprecisão dos meios de tradicionais de navegação, fazendo com que não seja possível descrever a trajetória proposta com a acurácia necessária para a validação futura das técnicas de controle.

A seguir, será apresentada a curva teórica utilizada como perfil de voo na referência citada e também a forma como a posição dos rotores num conjunto coaxial pode alterar sua performance e a pertinência disso no presente trabalho.

Figura 1 – Curva Lemniscata de Gerono com parâmetro $\alpha = 1$



Fonte: o autor (2020)

1.1 Lemniscata de Gerono

A Lemniscata de Gerono (LAWRENCE, 1972), que pode ser interpretada, entre outras formas, como um caso especial da figura de Lissajous (com $\delta = \frac{\pi}{2}$, $a = 1$ e $b = 2$), é descrita matematicamente pelas Equações (1.1), (1.2) e (1.3) (WEISSTEIN..., s.d.), que trazem as representações da curva em coordenadas cartesianas, polares e em equações paramétricas, respectivamente. Nelas, o parâmetro α é o máximo valor de x da curva, que representa metade de seu comprimento no eixo x. Uma representação da curva pode ser vista na Figura 1, na qual é aplicado $\alpha = 1$.

$$x^4 = \alpha^2(x^2 - y^2) \tag{1.1}$$

$$r^2 = \alpha^2 sec^4(\theta) cos(2\theta) \tag{1.2}$$

$$\begin{cases} x = \alpha cos(\phi) \\ y = \frac{\alpha}{2} sen(2\phi) \end{cases} \tag{1.3}$$

Analisando a Equação (1.3) e a Figura 1, nota-se que a mesma é composta por uma oscilação regida por um cosseno no eixo x e por um seno com metade da amplitude e o dobro da frequência no eixo y. Assim, altera-se a Equação (1.3) para a Equação (1.4) de forma que essa última torna as dimensões da figura configuráveis, sendo α o máximo valor de x e β o máximo valor de y, compondo uma curva análoga à Lemniscata com comprimento 2α e largura 2β.

$$\begin{cases} x = \alpha cos(\phi) \\ y = \beta sen(2\phi) \end{cases} \tag{1.4}$$

1.2 Alteração de performance de rotores em configuração coaxial

Rotores dispostos em configuração coaxial, como os do veículo aqui tratado, apresentam alteração de performance quando comparados com rotores dispostos em configuração unitária. Isso se deve, principalmente, ao fato do rotor superior gerar turbulências no ar coletado pelo rotor inferior, prejudicando esse último no que diz respeito à sua eficiência. O distanciamento entre as hélices também deve ser levado em consideração, apesar de ser necessária uma distância muito grande para que não haja a influência citada (RAMASAMY, 2013).

Apesar de apresentar vantagens quando se trata de contornar falhas de motores, a configuração coaxial merece atenção especial. Determinadas configurações de formato e proximidade das hélices podem resultar numa redução de performance tal que o rotor inferior necessitaria de 35% mais potência induzida para produzir a mesma quantidade de empuxo que um rotor unitário (RAMASAMY, 2013), por exemplo. Por esse motivo, análises a respeito da eficiência dos motores se mostram válidas a fim de se evitar configurações desvantajosas.

Uma maneira de realizar tais análises é através de dados reais. Por esse motivo, elas farão parte do escopo desse estudo com caráter secundário.

Apresentado o problema central, traça-se os objetivos primordiais a serem atingidos para que a contribuição planejada seja concretizada, que serão apresentados no próximo capítulo.

2 OBJETIVOS

2.1 Objetivo principal

O presente trabalho de conclusão de curso objetiva implementar de forma prática o procedimento de teste, proposto por (FARCONI, 2019) em forma de simulação computacional, aplicado a um *drone* octocóptero.

2.2 Objetivos específicos

A fim de se concluir o principal propósito desse trabalho, os seguintes objetivos específicos devem ser cumpridos:

- Definir e configurar um sistema de aquisição de posição com precisão e rapidez;
- Definir e configurar equipamentos adicionais a serem instalados no veículo;
- Compatibilizar os dados do sistema de aquisição de posição com o formato de comunicação do veículo;
- Efetivar a comunicação sem fio com o veículo para transmissão dos dados;
- Validar a implementação prática proposta.

2.3 Objetivo secundário

Fora do cunho central, mas pertinente ao assunto, o objetivo secundário é analisar como as diferenças de desempenho nos rotores dos conjuntos coaxiais se mostram num caso real.

3 DESENVOLVIMENTO

Neste capítulo serão expostos, primeiramente, os materiais, equipamentos, técnicas e metodologias utilizados no desenvolvimento do trabalho e, posteriormente, os resultados que foram obtidos acompanhados de análise dos mesmos.

3.1 Materiais e métodos

A presente seção descreverá os procedimentos realizados e as ferramentas usadas. Serão apresentados os equipamentos iniciais, como o veículo que esteve presente desde o começo das atividades, bem como os que foram agregados no decorrer do estudo e como eles foram aplicados, tendo-se sempre em mente o cumprimento dos objetivos específicos.

3.1.1 Veículo e controladores

O veículo em questão é um octocóptero na configuração "octa-quad" por possuir quatro braços com oito rotores coaxiais dois a dois. O *drone* é um produto comercial e está equipado com um controlador igualmente disponível no mercado, o Pixhawk 2, que inclui sensores inerciais, bússola, GPS, controle PID e funcionalidades padrões de *drones* comerciais (CUBE..., 2019). A Figura 2 mostra uma imagem do veículo.

A forma tradicional de se operar o *drone* como produto comercial é através dos controladores de voo ou estações de solo. Com eles é possível planejar rotas e planos de voo para o VANT, enviar comandos de ações básicas, obter dados de telemetria, observar e alterar variáveis do computador de bordo, atualizar *firmware*, calibrar sensores inerciais e outras aplicações a nível usuário principalmente. O controlador adotado foi o Mission Planner (MISSION..., 2019), que é compatível apenas com o sistema operacional Windows®, mas apresenta uma interface amigável e intuitiva para o tratamento inicial do *drone*.

O controle remoto utilizado para voos manuais e como ferramenta de auxílio em voos autônomos é o Spektrum DX7s (DX7S..., 2014), que pode ser visto na Figura 3. Além do fato de operar o veículo manualmente, uma importante funcionalidade que foi aproveitada nos voos autônomos é a presença de uma chave seletora mecânica à qual pode-se associar três modos de voo, um para cada posição da chave. A escolha dos modos se dá através de configuração feita no Mission Planner e promove mudanças fáceis e rápidas entre os modos de operação especificados, sendo de grande serventia tanto pela praticidade do comando, quanto como medida de segurança em casos de comportamentos inesperados por parte do VANT.

Figura 2 – *Drone* octocóptero

Fonte: o autor (2020)

Figura 3 – Controle remoto Spektrum DX7s

Fonte: (DX7S. . . , 2014)

3.1.2 Configuração computacional

O conjunto computacional utilizado para desenvolver o presente trabalho foi assim escolhido para atender a todas as necessidades que a configuração de equipamentos apresentava. Ele conta com dois computadores de mesa e um portátil, que desempenham os seguintes papeis:

- O primeiro computador de mesa conta com o sistema operacional Windows 8® e é dedicado à operação do sistema de posicionamento de câmeras Vicon®, incluindo o *software* necessário para sua operação. Esse será tratado por "*ViconHostPC*";

- O segundo usa o sistema Linux® com Ubuntu 18.04® e visa promover a interface entre *drone* e sistema de posicionamento, iniciando o controlador MAVProxy e a biblioteca Pyvicon, além de realizar a comunicação Wi-Fi® com o módulo do veículo. Esse será tratado por "*PythonHostPC*";

- O último, portátil, conta com dois sistemas operacionais, permitindo alternância entre Windows® e Linux®. Atuou como computador auxiliar, comportando a estação de controle de solo para coleta de dados de telemetria via rádio, tanto na fase de desenvolvimento, quanto após cada teste realizado. Executou, também, os códigos que regiram os voos autônomos. Esse será tratado por "*SupportPC*".

3.1.3 Sistema de Posicionamento Vicon®

Com a finalidade de cumprir o primeiro objetivo específico, que trata da definição e configuração de um sistema de aquisição de posição adequado, escolheu-se o sistema de câmeras Vicon® para posicionamento de precisão. O sistema conta com quatro câmeras de alta definição e alta velocidade de captura, como a vista na Figura 4b, que trabalham em conjunto identificando marcadores refletivos colados no corpo sob análise. Através da sobreposição da informação de movimentação dos marcadores nas imagens obtidas por cada câmera, o conjunto calcula a posição do objeto no espaço com grande precisão. A configuração conta também com um servidor, o MX Giganet Box®, que recebe a informação proveniente das câmeras e as retransmite via protocolo Ethernet® com taxa de transmissão gigabit (10^6bits/s) (VICON. . . , 2010).

Na parte de *software*, está o Vicon Tracker® e o Vicon DataStream®. O primeiro trata-se do programa principal de calibração e operação das câmeras, contando com funções de configuração de objetos, definição de máscaras (zonas mortas) em cada câmera, funcionalidades de gravação e reprodução de ensaios, entre outros. O segundo é o programa de transmissão rápida dos dados originados no primeiro. Esse conjunto de *software* usa o sistema operacional Windows® e possui um computador dedicado à sua operação (*ViconHostPC*).

3.1.3.1 Posicionamento e fixação das câmeras

Um passo fundamental na utilização correta do sistema Vicon® consiste no posicionamento adequado das câmeras.Nesse caso, as mesmas foram posicionadas no galpão mostrado na Figura 4a, no parapeito do mezanino, apontadas para a região central do piso inferior. O dispositivo metálico visto na Figura 4b realizou a interface entre o suporte original de fábrica das câmeras e o local escolhido para apoio, garantindo a rigidez necessária. Após a definição de cada local, as câmeras foram apontadas para o centro aproximado do volume útil de atuação e não foram mais movidas para que se iniciasse o procedimento de calibração. Por volume útil, entende-se como o espaço sujeito à cobertura e atuação das câmeras, tendo em vista que, para um marcador refletivo ser corretamente identificado, é necessário que ele seja "visto"por pelo menos duas câmeras (VICON. . . , 2010). A figura 5 ilustra uma representação gráfica do volume útil num caso arbitrário.

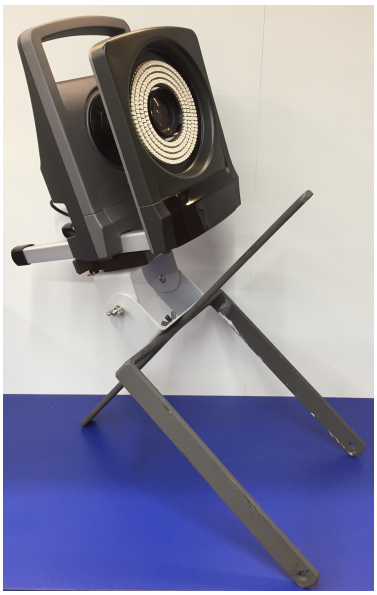
3.1.3.2 Calibração

No ambiente de uso é normal que haja um pequeno ruído nas leituras causado por pontos refletivos não intencionais ou mesmo reflexo do próprio holofote de uma câmera visto pelas outras. Por esse motivo, a calibração se inicia com a definição das máscaras, que são regiões compostas por conjuntos de píxeis que passarão a ser ignorados por cada câmera. Dessa forma, durante a aplicação do sistema, os pontos de ruído anteriormente citados não influenciam nas

Figura 4 – Espaço utilizado para voos internos com esquema de adaptação para fixação das câmeras Vicon® MX T40-S

(a) Galpão - Destaque para parapeito do mezanino

(b) Câmera com dispositivo de interface



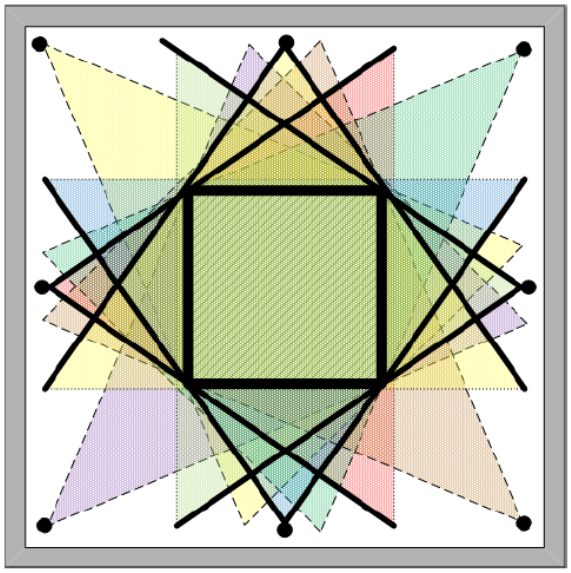
Fonte: o autor (2020)

leituras visto que não são mais levados em consideração. Vale notar que deve-se ter cautela nessa etapa, pois uma região de máscara muito grande pode indicar que o posicionamento das lentes deve ser alterado para minimizar os efeitos de uma zona cega no volume de captura.

Na sequência, ocorre a calibração propriamente dita. Para tal, utilizou-se a Vicon Active Wand®, vista na Figura 6, que é um bastão em forma de "T"com cinco LEDs posicionados em sua estrutura e que é acenado para o conjunto de câmeras durante a calibração. O procedimento é feito de forma ampla, contemplando todas as lentes ao mesmo tempo, e seguindo o parâmetro da quantidade de quadros da imagem a serem capturados, definido no Vicon Tracker®. Nesse caso, definiu-se para o tal parâmetro o valor de 2000, que significa que cada câmera deverá captar 2 mil quadros em que consegue visualizar todos os cinco LEDs do bastão. Como forma de boa prática, é recomendado acenar o instrumento pelo volume todo e para cada par de câmeras, de forma a interligá-las duas a duas. Concluídas as 2000 capturas mencionadas, o procedimento é repetido na configuração de ajuste fino. Essa etapa ajuda a reduzir o erro de medição cada câmera para a ordem de 10^{-1}, garantindo uma ótima precisão na obtenção das imagens.

A última etapa da calibração consiste na definição da origem do sistema de coordenadas do volume útil. Para tal, posiciona-se o bastão com o LED central onde deseja-se que seja definida a origem. Os braços do instrumento serão os eixos do sistema de coordenadas de referência (VICON. . . , 2019).

Figura 5 – Representação de um volume útil arbitrário



Fonte: (VICON. . . , 2010)

Figura 6 – Bastão para calibração - Vicon Active Wand®

Fonte: (VICON. . . , 2019b)

3.1.3.3 Definição de objetos

Finalizados os passos anteriormente descritos, o sistema está pronto para reconhecer os marcadores refletivos. Posicionou-se, então, os marcadores no *drone* conforme visto na Figura 7a e colocou-se o veículo dentro do volume de atuação do sistema. No Vicon Tracker®, é possível observar os marcadores identificados e associar o conjunto dos mesmos a um objeto específico, o qual se tornará a base da aquisição da posição, calculada como $[x, y, z]$ em relação à origem do volume. O objeto definido para o VANT pode ser visto na captura de tela da Figura 7b, contando com as interconexões entre os marcadores, bem como a origem do objeto e seu sistema de coordenadas local.

Concluídos os passos de configuração do sistema de posicionamento, o Vicon Tracker® passa a receber os dados crus vindos das câmeras, os processa vinculando à posição de um objeto definido via *software*, repassa a informação ao Vicon DataStream®, que retorna-a ao MX Giganet Box®, disponibilizando o dado de posição em suas portas Ethernet® gigabit para quaisquer equipamentos conectados.

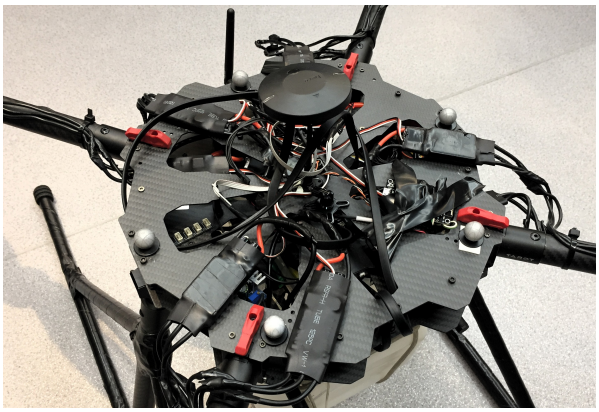
O esquema de conexões que abrange o sistema Vicon® pode ser visto na Figura 8.

3.1.4 Módulo Wi-Fi® ESP8266

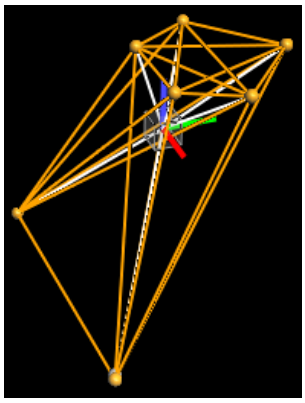
Esta subseção concretiza o segundo e o quarto objetivos específicos, que dizem respeito aos equipamentos adicionais a serem instalados no veículo e à efetivação da comunicação sem fio com o mesmo. Para tal, foi necessário estabelecer uma conexão rápida entre o computador

Figura 7 – Procedimento de definição de objeto virtual para *drone*

(a) Marcadores refletivos fixados no veículo

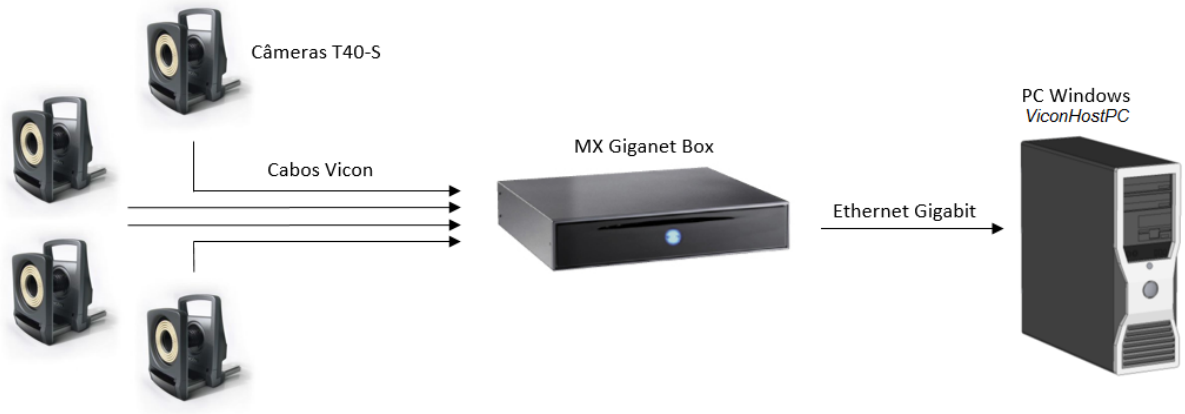


(b) Vicon Tracker®



Fonte: o autor (2020)

Figura 8 – Esquema de conexões do sistema Vicon®



Fonte: o autor (2020)

responsável pelo processamento dos dados (*PythonHostPC*) e o veículo. Apesar de já possuir uma antena de comunicação via rádio dedicada para telemetria, o octocóptero não dispunha de outro método de baixa latência que possibilitasse a efetivação do envio de dados conforme era necessário, uma vez que a taxa de transmissão via rádio é muito baixa para a aplicação aqui trabalhada. A comunicação padrão estabelecida por rádio trabalhava com a velocidade de 57600 bps (bits por segundo), podendo ser configurada até 250 kbps com redução do alcance (SIK. . . , 2019), ao passo que o valor indicado para a comunicação estabelecida nessa aplicação era de 921 kpbs. Com isso, a solução adotada foi estabelecer uma conexão Wi-Fi®, caracterizada por alta velocidade de transmissão e baixa perda de informação. Para efetivação dessa solução, escolheu-se o módulo Wi-Fi® ESP8266, conforme indicado em (VICON. . . , 2019a), e visto na Figura 9. O módulo em questão trata-se do kit de desenvolvimento LoLin que agrega em uma única placa eletrônica os seguintes itens (ESP8266. . . , 2019a):

Figura 9 – Placa de desenvolvimento LoLin com módulo Wi-Fi® ESP8266

Fonte: (ESP8266..., 2019a)

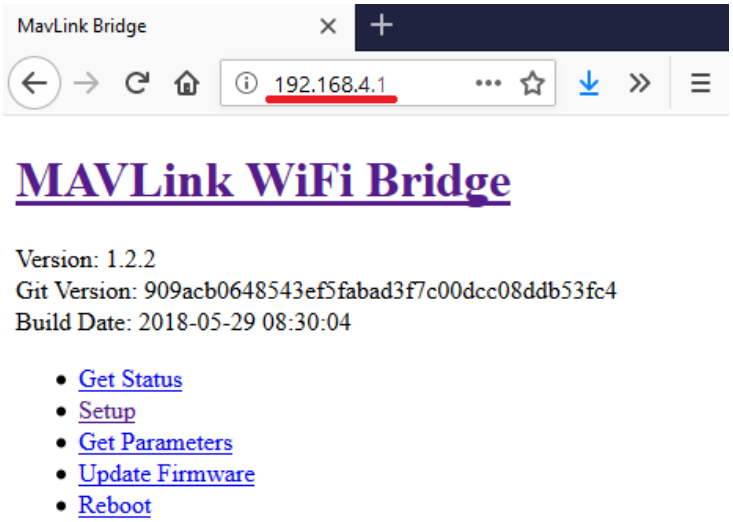
- Módulo Wi-Fi® ESP8266 com antena embutida;
- Conversor USB-Serial CH340G acoplado à porta micro-USB;
- Regulador de tensão de 3,3V AMS1117;
- Pinagem compatível com placa de prototipagem.

Conforme mencionado anteriormente, a comunicação com o veículo se deu via protocolo MAVLink. Então, para que o módulo Wi-Fi® pudesse receber os pacotes de mensagens corretamente, foi gravado nele o *firmware* MAVESP8266 através de sua porta micro-USB, que acessa automaticamente a memória do kit. Após regravado, o módulo foi conectado ao *drone* através da porta de telemetria conforme indicado em (ESP8266..., 2019b), que fornece alimentação 5V para a placa e estabelece comunicação através dos pinos RX/TX.

Aqui cabe uma nota importante quanto à operação correta do módulo. O kit de desenvolvimento, visando facilitar a configuração por parte do usuário, insere o conversor USB-Serial integrado à placa, permitindo que gravações de *firmware* e outras configurações sejam realizadas de maneira simples, sem a necessidade de colocar o dispositivo em modo de inicialização ou de se adquirir um conversor USB-Serial externo. Isso se mostra de grande utilidade no primeiro uso do kit, quando o *firmware* MAVESP8266 foi carregado. Porém, os pinos de comunicação RX e TX, que compõem parte da porta Serial, também são responsáveis pela comunicação com o módulo Wi-Fi® propriamente dito, como mostra a Figura 37 do Anexo A. Essa configuração do circuito, por sua vez, interfere na comunicação direta entre a Pixhawk 2 e o kit LoLin, provocando uma falha na comunicação como se o kit não estivesse conectado ao *drone*. A solução para essa questão está na remoção do chip conversor USB-Serial, ou pelo menos dos pinos que se conectam a RX e TX. Isso inutiliza a porta micro-USB da placa, mas isso não se torna um problema, visto que gravado uma vez o *firmware*, é possível fazer atualizações via Wi-Fi® e regravações diretamente pelos pinos da porta Serial.

Concluída a montagem da placa, conectou-se o computador (*PythonHostPC*) ao ponto de acesso do kit, chamado "Ardupilot" e acessou-se sua configuração via navegador, usando o IP padrão 192.168.4.1. Na página, cuja tela inicial é ilustrada na Figura 10, é possível ver a estatística dos pacotes recebidos, enviados e perdidos, por exemplo, além de consultar e

Figura 10 – Captura de tela da página de configuração da rede Wi-Fi® do módulo ESP8266



Fonte: (ESP8266..., 2019b)

configurar os parâmetros da conexão Wi-Fi®. A configuração que deve ser feita no kit para concluir o uso do ESP8266 é definir sua taxa de transmissão como 921600. Esse é o valor adotado para a comunicação e garantirá uma conexão rápida entre computador (*PythonHostPC*) e *drone*, conforme é requerido para um sistema de posicionamento. Por parte da Pixhawk 2, é necessário conectá-la via USB ao controlador de voo e alterar os seguintes parâmetros (ESP8266..., 2019b):

- SERIAL1_PROTOCOL = 2 - Define MAVLink 2 como o protocolo de comunicação da porta de telemetria número 1;
- SERIAL1_BAUD = 921600 - Define 921000 como a taxa de transmissão da porta de telemetria número 1 (COMPLETE..., 2019).

3.1.5 Biblioteca Pyvicon

Após a configuração do sistema de câmeras Vicon® e o estabelecimento da comunicação com o *drone* via Wi-Fi®, é necessário efetivar o envio dos dados de posicionamento ao veículo, concluindo assim, o terceiro objetivo específico, que traz a necessidade de compatibilização dos dados de posição obtidos com o protocolo usado pelo VANT. Para tal, foi utilizada a biblioteca Pyvicon (PYVICON..., 2018) que, como o próprio nome expõe, é baseada na linguagem Python, utiliza o sistema operacional Linux® e realiza a conversão dos dados crus originados no sistema Vicon® para um formato adequando ao *drone*: o protocolo MAVLink.

A obtenção dos dados se inicia nas câmeras, que identificam os marcadores refletivos no espaço e enviam dados básicos de posicionamento ao servidor MX Giganet Box®. Através da conexão cabeada o servidor envia esses dados ao computador Windows® dedicado ao sistema

Vicon® (*ViconHostPC*), onde o Vicon Tracker® vincula os marcadores e a posição obtida pelas imagens ao objeto registrado e devolve essa informação, através do Vicon DataStream®, ao servidor, que passa a retransmitir os dados (objeto + posição) a outras conexões existentes. Com isso, o computador Linux® (*PythonHostPC*), também conectado via cabo Ethernet® ao MX Giganet Box®, passa a receber o posicionamento do objeto do sistema Vicon®. Finalmente, a biblioteca Pyvicon converte os dados de posição $[x, y, z]$ para um formato de GPS baseado em latitude, longitude e altitude, que é facilmente processado pelo *drone*, em mensagens segundo o protocolo MAVLink.

Como o veículo receberá os dados como se fossem de seu GPS real, alguns parâmetros devem ser alterados para o funcionamento correto da nova configuração (VICON..., 2019a). As fontes de informação devem ser alteradas para o GPS, uma vez que esse representa o sistema Vicon®, e os dispositivos GPS originais devem ser desconsiderados. As alterações foram as seguintes:

- EK3_ENABLE = 1 - Habilita os cálculos do terceiro Filtro de Kalman Estendido implementado no controlador do veículo;
- EK2_ENABEL = 0 - Desabilita os cálculos do segundo Filtro de Kalman Estendido implementado no controlador do veículo;
- AHRS_EKF_TYPE = 3 - Define que o terceiro Filtro de Kalman deverá ser usado para estimar atitude e posição;
- EK3_GPS_TYPE = 0 - Define que as informações de velocidade 3D e posição 2D serão obtidas pelos dados do GPS;
- EK3_MAG_CAL = 5 - Define que a informação de guinada será obtida através de uma fonte externa, desabilitando o uso do magnetômetro conforme recomendado para um ambiente onde o comportamento do campo magnético não é estável (aplicável em ambientes internos em geral);
- EK3_ALT_SOURCE = 2 - Define que o GPS será a fonte primária de altitude para os cálculos do Filtro de Kalman;
- GPS_TYPE = 14 - Define que os dados de GPS virão através do protocolo MAVLink;
- GPS_DELAY_MS = 50 - Define o tempo de atraso nos dados do GPS que o piloto automático deve ser capaz compensar;
- COMPASS_USE = 0 - Desabilita o uso da bússola para obtenção do dado de guinada, passando a ser usado o proveniente do GPS;
- COMPASS_USE2 = 0 - Desabilita o uso da segunda bússola para obtenção do dado de guinada;

- COMPASS_USE3 = 0 - Desabilita o uso da terceira bússola para obtenção do dado de guinada (COMPLETE..., 2019).

3.1.5.1 Controlador MAVProxy

A biblioteca Pyvicon tratada realiza apenas o tratamento dos dados, de forma que ela deve ser executada de dentro de um controlador que já possua os métodos de comunicação com o VANT. O controlador aqui utilizado é o MAVProxy, uma estação de controle operada por linha de comando para veículos que efetivam sua comunicação via protocolo MAVLink. Ele visa ser leve e minimalista e permite a importação de módulos que complementam suas funções e operação (MAVPROXY, 2019). E é na forma desses módulos que se dará a ação da biblioteca Pyvicon.

Na inicialização do controlador para o caso em questão, executa-se o seguinte comando:

mavproxy.py –master :14550 –console –map

sendo que *mavprox.py* executa a inicialização propriamente dita, *–master :14550* especifica que a porta para estabelecer a comunicação com o veículo é a 14550, que corresponde à porta Wi-Fi®, e *–console –map* importa os módulos do console e do mapa. A Figura 11 mostra a captura de tela após a execução do comando descrito, podendo-se ver o mapa e o console com a definição das características do *drone*, bem como os parâmetros que foram carregados ao controlador.

3.1.5.2 Inicialização da biblioteca

Estabelecida a comunicação através do controlador, importa-se e habilita-se a biblioteca Pyvicon usando os seguintes comandos (VICON..., 2019a):

module load vicon

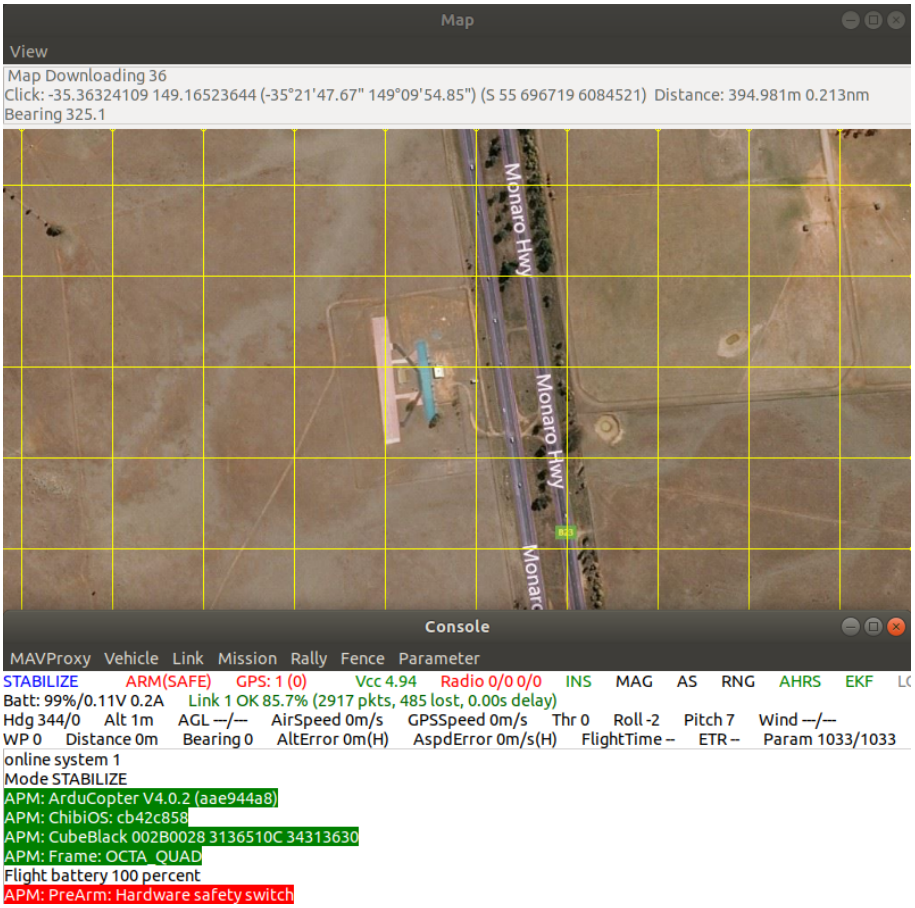
vicon set

vicon start

sendo que o primeiro importa o módulo gerado pela biblioteca, o segundo define os parâmetros de operação do módulo e o terceiro concretiza o funcionamento da Pyvicon, dando início à transmissão dos dados de posicionamento ao *drone*.

Vale notar que, ao executar o segundo comando descrito anteriormente, aparecem na tela

Figura 11 – Console e mapa do controlador MAVProxy com localização padrão



Fonte: o autor (2020)

Figura 12 – Comandos de inicialização da biblioteca Pyvicon e respectivas respostas contendo localização padrão do controlador

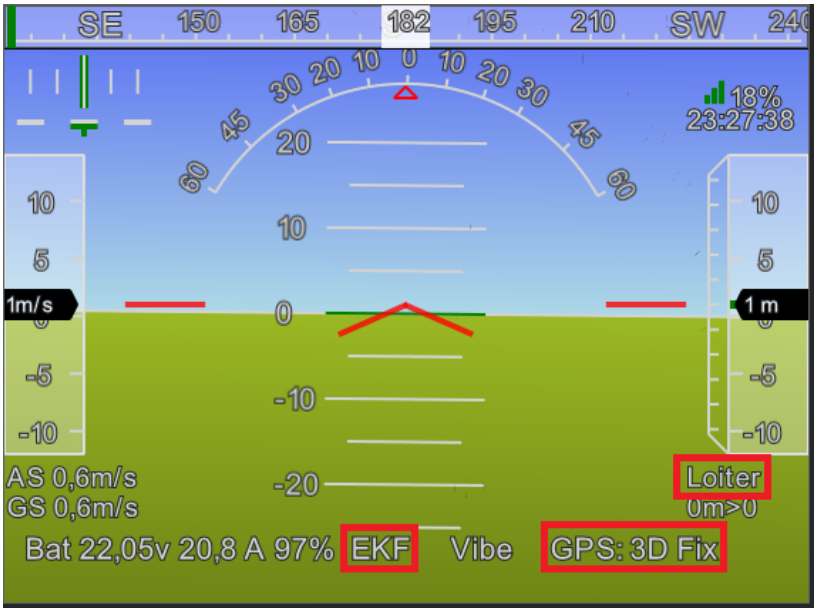
```
STABILIZE> module load vicon
STABILIZE> Loaded module vicon

STABILIZE> vicon set
STABILIZE>            gps_nsats 16
            gps_rate 5
                host vicon
          origin_alt 584.0
          origin_lat -35.363261
          origin_lon 149.16523
       vel_filter_hz 30.0
         vision_rate 14
          yaw_offset 0.0

STABILIZE> vicon start
STABILIZE> Opening Vicon connection to vicon
Configuring vicon
vicon ready
Connected to subject 'black_box' segment 'black_box'
Vicon frame rate 100.0
```

Fonte: o autor (2020)

Figura 13 – Tela do controlador de voo indicando parâmetros em conformidade



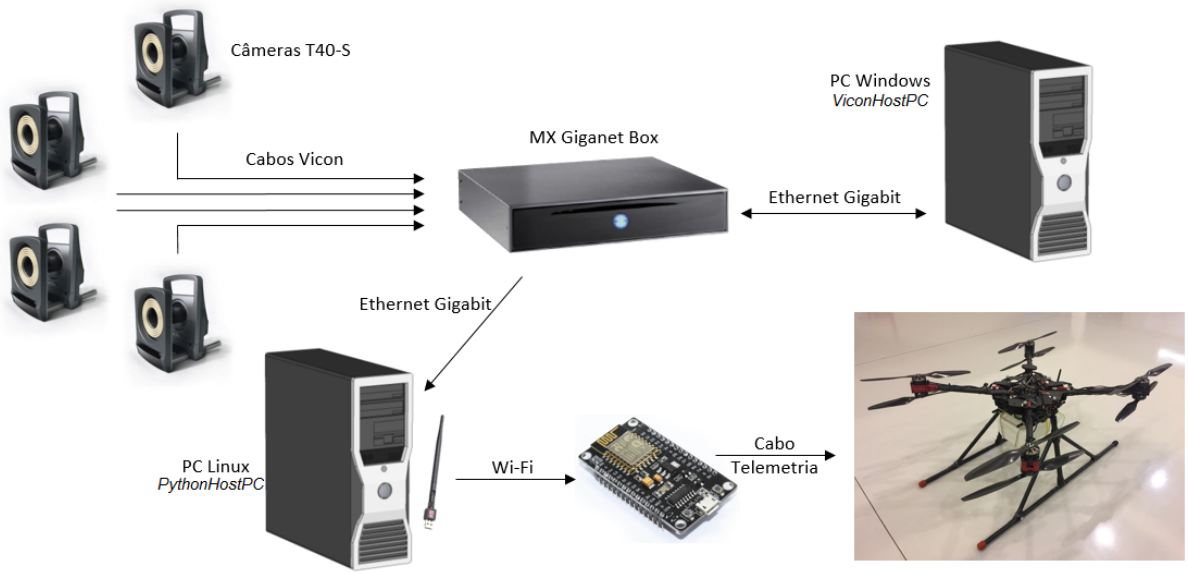
Fonte: o autor (2020)

os parâmetros vistos na Figura 12, que correspondem à localização (latitude, longitude e altitude) padrão do controlador MAVProxy conforme visto anteriormente na Figura 11. Porém isso não impacta no objetivo de enviar dados de posição ao veículo, uma vez que tais coordenadas serão referenciadas na origem do volume útil do sistema Vicon®, conforme anteriormente definido no procedimento de calibração. Com isso, a leitura dos dados de telemetria do *drone* indicarão que ele se encontra nas dadas coordenadas padrão, mas essas têm caráter unicamente arbitrário e a operação do VANT será conforme se espera.

Estando concluída a execução do terceiro comando, a transmissão dos dados de posição obtidos pelas câmeras, processados pelo Vicon Tracker® e traduzidos pela Pyvicon passam a ser enviados ao *drone*, que os recebe como dados de GPS. Para fins de confirmação do sucesso da conexão, conectou-se a segunda porta de transmissão de telemetria do *drone* a um segundo controlador de voo via rádio, de forma que foi possível observar que no momento do início da transmissão o VANT acusa o travamento do GPS e assume as informações de latitude, longitude e altitude padrão anteriormente citadas como sendo verdadeiras. A Figura 13 mostra parte da tela do controlador Mission Planner sendo possível ver os dados do *drone*. As informações de bateria, filtro de Kalman, níveis de vibração e travamento de GPS vistas em branco na porção inferior da imagem indicam que tais parâmetros apresentam boas condições para voo. Também é possível ver o modo de voo *loiter* no canto inferior direito.

A Figura 14 traz o esquema completo das conexões que abrangem o sistema de posicionamento descrito.

Figura 14 – Esquema completo de conexões entre equipamentos e veículo



Fonte: o autor (2020)

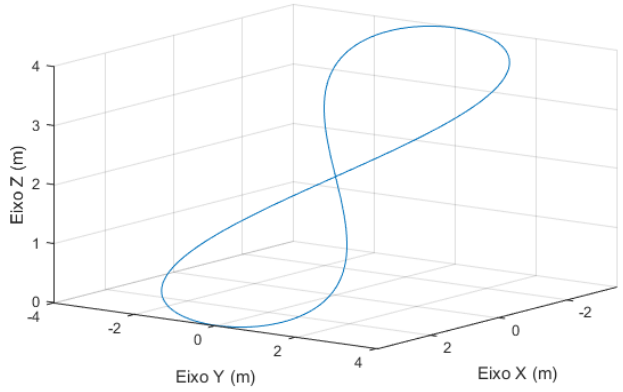
3.1.6 Perfil de voo

A presente subseção tratará do quinto objetivo específico proposto, que traz a validação da implementação. Tal validação é um processo complexo que conta, por exemplo, com a realização da montagem mostrada na Figura 14 e também com a concretização do perfil de voo que deseja-se cumprir. Esse segundo aspecto será aquele tratado a seguir.

3.1.6.1 Lemniscata de Gerono

Conforme explicitado anteriormente, o perfil de voo que deseja-se realizar nesse trabalho é aquele utilizado em (FARCONI, 2019) na forma de simulação computacional, que é caracterizado por uma Lemniscata de Gerono (LAWRENCE, 1972). No caso da aplicação que busca-se atingir, a Lemniscata descrita é tal que posiciona-se no espaço com sete metros de comprimento (eixo x), quatro metros de largura (eixo y) e com uma inclinação em relação ao plano do solo de forma que uma extremidade do comprimento se situa a quatro metros acima da outra (eixo z). Tal curva pode ser descrita matematicamente pelas equações paramétricas mostradas na Equação (3.1) e uma representação tridimensional da mesma pode ser vista em 15. A ideia por trás de um perfil peculiar como tal é desafiar o veículo e seu método de controle, que passam a ser submetidos a curvas acentuadas, momentos de aceleração e variações de altitude de

Figura 15 – Lemniscata de Gerono - Dimensões usadas na simulação de (FARCONI, 2019)



Fonte: o autor (2020)

forma simultânea.

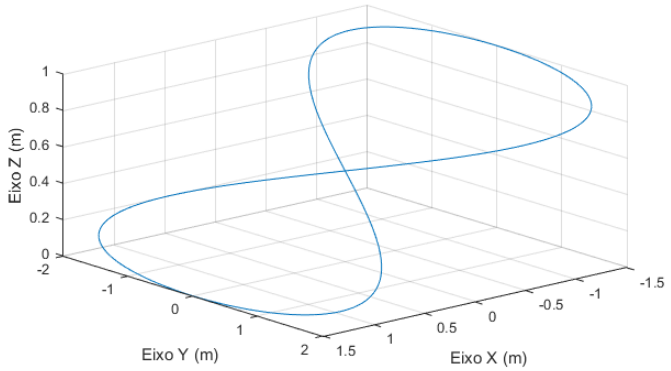
$$\begin{cases} x = 3.5cos(\phi) \\ y = 2sen(2\phi) \\ z = 2(-cos(\phi) + 1) \end{cases} \tag{3.1}$$

3.1.6.2 Comando do perfil

Para converter o perfil descrito em trajetória real, foi necessário, primeiramente, reduzir as dimensões da trajetória em função de limitações espaciais do volume útil definido pelo sistema de posicionamento por câmeras. A Lemniscata passou uma de forma espacial com dimensões 7m x 4m x 4m para uma com 3m x 4m x 1m, respectivamente, conforme ilustra a Figura 16. Definido o novo perfil adaptado, houve a implementação de um código escrito em C++ para definição dos pontos que compõe o caminho e fazer o envio para o *drone*. O código é baseado na lógica trazida pelo autor em (FARCONI, 2019) e define um conjunto de oito *waypoints* principais que são fracionados de forma a tornar a trajetória um pouco mais suave e contínua. O Anexo B mostra o código principal da execução.

A ferramenta de transmissão utilizada foi o MAVROS (MAVROS..., 2018), que é um pacote que promove comunicação com controladores de veículos através do protocolo MAVLink e faz uso da estrutura de nós, serviços e tópicos característica do ROS (*Robot Operating System*) (ROS..., 2018). E para isso, o código em C++ teve de ser baseado nessa estrutura, chamando serviços determinados para comandos básicos e de posição e se comunicando com o nó de conexão em que o *drone* se posiciona. Na aplicação dada, utilizou-se o computador portátil (*SupportPC*) dotado de uma antena de rádio para efetivar a comunicação com o *drone*.

Figura 16 – Lemniscata de Gerono - Dimensões para voo real



Fonte: o autor (2020)

Estabelecida a conexão, o código foi executado e passou a enviar as informações de posição para que o VANT seguisse o perfil proposto.

Observa-se que pelo fato de que o comando da trajetória ocorre via rádio por uma porta de telemetria do *drone* e de que o módulo Wi-Fi® se conecta à outra porta de telemetria, não foi possível conectar o veículo ao controlador Mission Planer (*SupportPC*) nos voos autônomos que recebiam o perfil em tempo real.

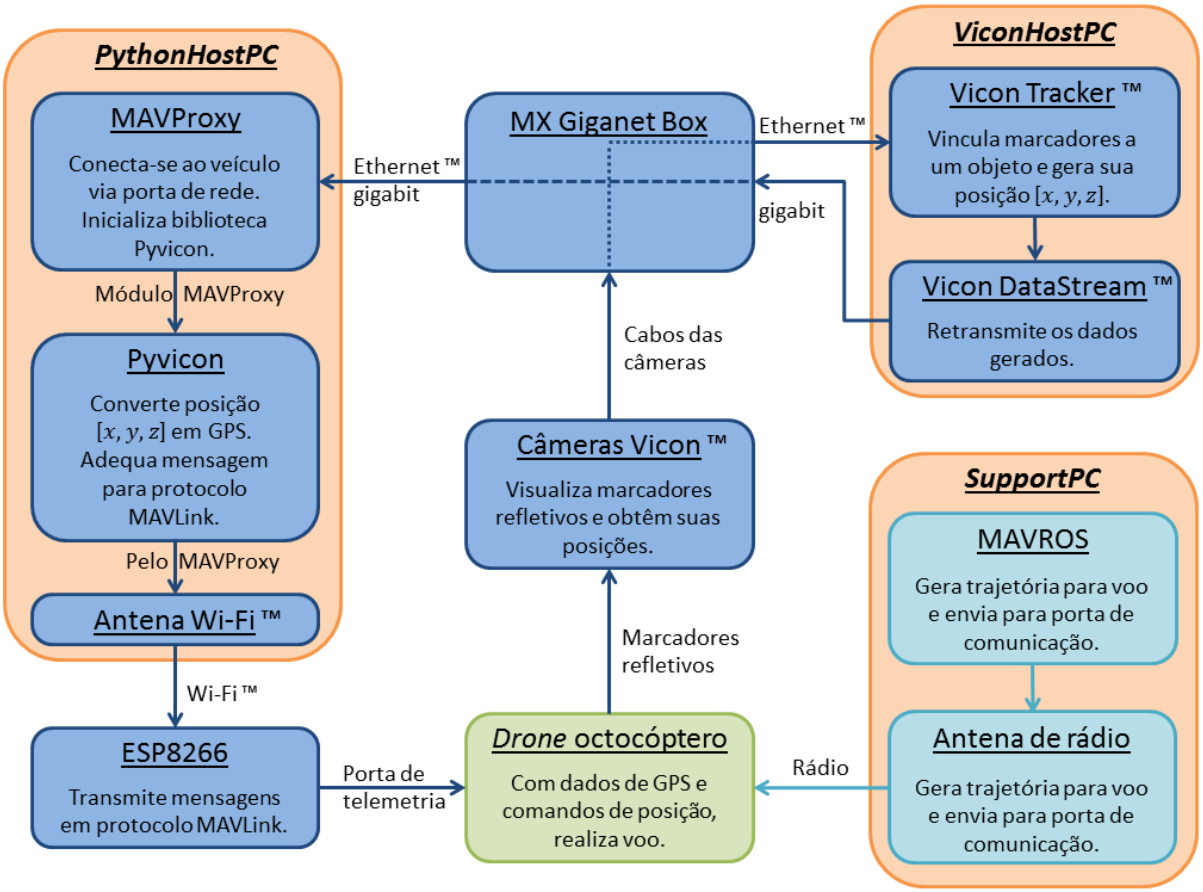
Com isso, completa-se toda a configuração envolvida no sistema de posicionamento e no envio de comandos para execução de trajetórias autônomas. O diagrama de blocos da Figura 17 traz de forma visual como se dá o fluxo de informação na configuração final. Os blocos em laranja são os computadores utilizados, o veículo é apresentado em verde, o sistema Vicon® é mostrado em azul-escuro e o envio de comandos de posição tem a cor verde-água.

3.1.7 Filtro de Kalman

Aqui será tratado, de forma breve e prática, a operação do Filtro de Kalman, como ele se insere na aplicação estudada e como seus resultados contribuirão para a validação da implementação, reforçando o cumprimento do quinto objetivo específico.

O filtro de Kalman é um filtro recursivo usado extensivamente para estimar estados de um sistema dinâmico nas mais diversas aplicações. Ele opera agregando informações a respeito do modelo do sistema e suas incertezas, variáveis de controle, medições de todos os sensores e de seus ruídos, para gerar uma estimativa dos estados desse sistema. Essa estimativa calculada pela fusão de todos os elementos produtores de dados é melhor que qualquer análise de um único elemento. E pelo fato de ser recursivo, tem a vantagem de não armazenar todos os dados já adquiridos, uma vez que as informações registradas a cada iteração atualizam o resultado acumulado, que é passado a diante (NEGENBORN, 2003).

Figura 17 – Diagrama de blocos com fluxo de dados da configuração completa usada em voos autônomos



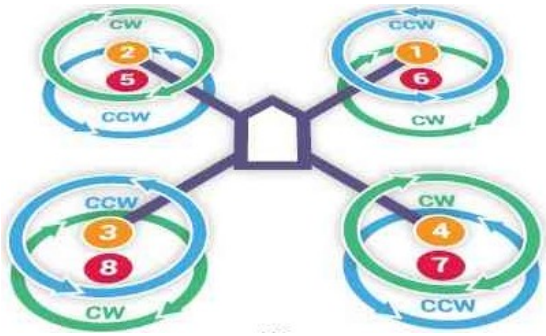
Fonte: o autor (2020)

Para ponderar a informação vinda de cada fonte, o filtro utiliza pesos associados a cada uma delas de acordo com a sua confiabilidade, que é estimada com base na covariância. Tratando de forma simplificada, o filtro analisa a interdependência das variáveis que ele recebe e a incerteza da predição dos estados do sistema, definindo pesos maiores para entradas que se associam melhor entre si e que produzem estimativas melhores sobre os estados. Se um sensor, por exemplo, apresenta leituras inconsistentes ou discrepantes dos demais, o filtro associa um peso menor a esse sensor, considerando que a confiabilidade do mesmo não é tão grande quanto às dos outros.

3.1.7.1 Erro de predição - *Innovation*

O erro de predição (ou *innovation*) será tratado futuramente nesse texto como medida de qualidade de sensoriamento. O erro pode ser definido como a diferença entre a estimativa de estado feita pelo filtro e a medição real do estado ao qual a estimativa correspondia (Zanni et al., 2017). É através dessa grandeza que será possível avaliar se o acréscimo de determinado

Figura 18 – Modelo de *drone* octocóptero com numeração arbitrária das hélices



Fonte: (COPTER. . . , 2014)

sensoriamento trará resultados positivos, agregando informação, ou negativos, ocasionando erros nas predições do filtro.

3.1.8 Contribuição para o modelo de simulação do *drone*

A seguir, será abordado o objetivo secundário desse trabalho, que busca estudar os efeitos de redução de performance nos rotores de um conjunto coaxial, como dos vistos no *drone* aqui tratado.

Para obter uma relação entre as eficiências das pás de acordo com seu posicionamento, considerou-se um modelo de *drone* idêntico ao do trabalho em questão. Definiu-se que os motores seguem a numeração que é indicada na Figura 18. Estabelece-se, então, as seguintes relações:

$$T_i = C_u P_i \quad \text{e} \quad T_j = C_d P_j \tag{3.2}$$

$$T_1 + T_6 = T_2 + T_5 = T_3 + T_8 = T_4 + T_7 = T_{arm} \tag{3.3}$$

$$4T_{arm} = T_t \tag{3.4}$$

$$T_t = mg \tag{3.5}$$

sendo que $i = 1, 2, 3, 4$, $j = 5, 6, 7, 8$, T_i é o empuxo produzido por cada rotor, T_t é o empuxo total do *drone*, C_u é a constante de eficiência dos motores de cima e C_d a dos de baixo, P_i é o PWM comandado para cada rotor superior, P_j é o PWM comandado para cada rotor inferior, T_{arm} é o empuxo resultante em cada braço do veículo, m é a massa do *drone* e g a aceleração da gravidade.

Com base nessas, pode-se estabelecer outras relações: das Equações (3.2) e (3.3), tem-se (3.6), e das Equações (3.4) e (3.5), tem-se (3.7):

$$C_u P_i + C_d P_j = T_{arm} \tag{3.6}$$

$$T_{arm} = \frac{mg}{4} \tag{3.7}$$

Consequentemente, de (3.6) e (3.7), obtêm-se (3.8):

$$C_u P_i + C_d P_j = \frac{mg}{4} \tag{3.8}$$

Na Equação (3.8) tem-se conhecimento das variáveis P_i e P_j pelos dados de telemetria, de m que é a massa do *drone* e de g, valor constante. Deseja-se, portanto, obter os coeficientes C_u e C_d que indicarão as eficiências dos motores superiores e inferiores, respectivamente, das quais será possível tirar conclusões a respeito do questionamento iniciado nessa seção. Porém, nota-se que há duas incógnitas em apenas uma equação, fazendo com que não seja possível encontrá-las de forma direta. A fim de solucionar a equação proposta, utiliza-se o método dos mínimos quadrados (Lai; Robbins; Wei, 1978) a partir da seguinte relação:

$$\begin{bmatrix} P_{i(t_0)} & P_{j(t_0)} \\ P_{i(t_1)} & P_{j(t_1)} \\ P_{i(t_2)} & P_{j(t_2)} \\ \vdots & \vdots \\ P_{i(t_f)} & P_{j(t_f)} \end{bmatrix} \cdot \begin{bmatrix} C_u \\ C_d \end{bmatrix} = \begin{bmatrix} \frac{mg}{4} \\ \frac{mg}{4} \\ \frac{mg}{4} \\ \vdots \\ \frac{mg}{4} \end{bmatrix}$$

Referenciando a relação matricial dada pela forma simplificada de $Ax = B$, tem-se que o valor ótimo dos coeficientes C_u e C_d (que definem o vetor x) pode ser encontrado resolvendo a igualdade:

$$x^* = (A^T A)^{-1} A^T B \tag{3.9}$$

Como planeja-se analisar a relação entre as eficiências dos motores em posições diferentes, os valores P não devem ser obrigatoriamente os valores de PWM. Assim, basta que os mesmos sejam proporcionais entre si para que se tenha a informação de como eles estão relacionados. Da mesma forma, o valor da massa do *drone*, além de ser dependente da carga no caso real, pode ser arbitrado para o objetivo comparativo em questão. Dessa forma, arbitra-se o valor de 10 kg para a massa do *drone* a fim de se concluir os cálculos propostos, cujos resultados serão apresentados na seção pertinente.

3.2 Resultados e discussão

A presente seção trará os resultados obtidos nesse trabalho, juntamente com a análise feita pelo autor para cada um deles. Serão apresentados os três voos principais que compõe a validação do procedimento de teste implementado, além de voos de caráter comparativo e os resultados da proposta de contribuição para o modelo do *drone*.

3.2.1 Voo preliminar

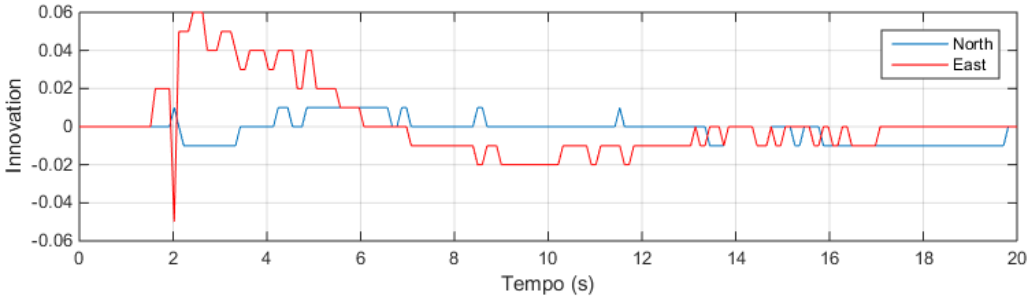
O primeiro voo realizado após a configuração completa do sistema de posicionamento ainda foi feito sem o uso efetivo dos dados oriundos do mesmo. Como os novos dados são vistos pelo *drone* como se fosse seu próprio GPS, essas informações são aplicadas, juntamente com as de outros sensores, no filtro de Kalman implementado no controlador Pixhawk 2. Porém, se essas informações não estiverem condizentes com a realidade, elas afetarão o filtro e ocasionarão erros de medidas que podem atrapalhar a navegação do veículo. Por esse motivo, o primeiro voo após concluído o procedimento da seção anterior foi feito usando-se o modo *althold*, que é um modo que não se baseia nos dados do GPS ao passo que recebe ordens de posição e altitude vindas do piloto pelo controle remoto (FLIGHT..., 2019). Isso garante que eventuais erros no filtro de Kalman causados pela nova fonte de GPS não afetem a estabilidade do octocóptero e que seja possível coletar os dados da telemetria do voo a fim de certificar que tais erros não estejam ocorrendo.

Coletados os dados de telemetria, era necessário verificar se os erros normalizados do filtro correspondentes a posição, velocidade e guinada eram valores baixos, de preferência inferiores a 0,1 (VICON..., 2019a). Isso garantiria que o novo método de posicionamento estaria enviando informações que correspondiam os visto pelas outras ferramentas de sensoriamento do veículo e não estaria atrapalhando os cálculos do filtro. As Figuras 19a, 19b e 19c mostram, respectivamente, os erros de predição normalizados do filtro de Kalman para as grandezas de posição em termos de *North* e *East*, velocidade em termos de *North*, *East* e *Down* e guinada (EXTENDED..., 2019) extraídos desse voo.

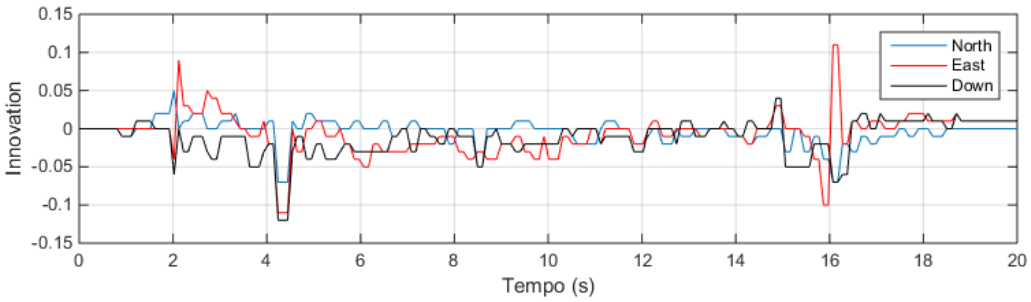
Como pode-se notar, os erros relacionados à posição e à velocidade foram realmente baixos, apresentando boa margem de segurança do valor de 0,1 sugerido. Porém, o erro associado à guinada mostrou-se maior, com patamares de 0,3 e atingindo valores próximos a 0,5 em determinados picos. Ainda assim, houveram faixas de valores dentro do esperado e o comportamento visto presencialmente no veículo não demonstrava indícios de problemas referentes à guinada do *drone*. Com essa análise, associada ao bom resultado do erro do filtro para posição e velocidade, decidiu-se prosseguir para o primeiro voo que efetivamente utilizasse o sistema Vicon®.

Figura 19 – Erros de predição normalizados do filtro de Kalman (*innovations*) do voo preliminar

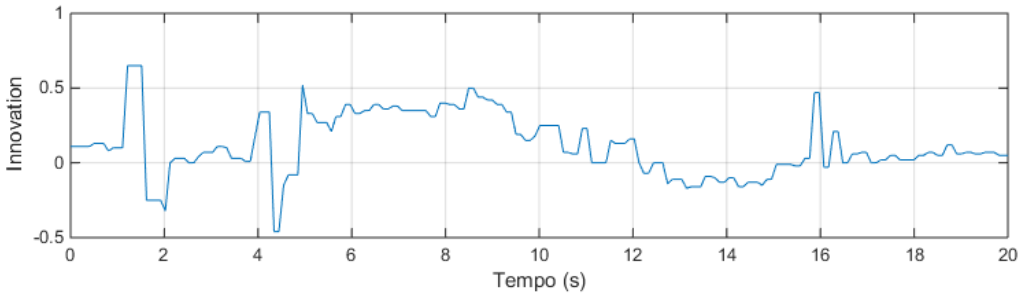
(a) Posição - NE



(b) Velocidade - NED



(c) Guinada



Fonte: o autor (2020)

Figura 20 – Dados em tempo real do controlador de voo

(a) Fase inicial do voo - Movimentação

(b) Fase final do voo - Descida

Fonte: o autor (2020)

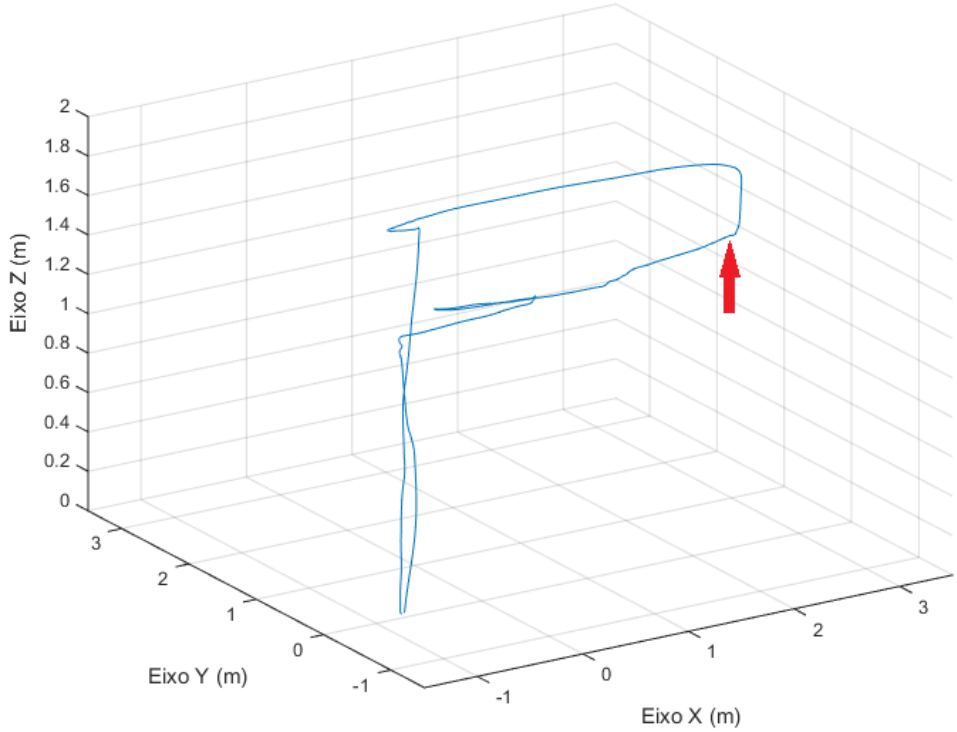
3.2.2 Primeiro voo

Para o primeiro voo, escolheu-se o modo *loiter* como principal, conforme indicado em (FLIGHT..., 2019). Nesse modo, o *drone* se orienta pelo seu GPS e o piloto controla taxas de elevação e de movimentação através do controle remoto. De forma auxiliar, foram definidos os modos *land* e RTL (*Return To Launch*), que são autônomos e realizam, respectivamente, as ações de pousar e de retornar ao ponto de lançamento em solo (FLIGHT..., 2019). Esses três modos podiam ser alternados pelo comando da uma chave de três posições localizada no controle remoto do piloto, conforme citado anteriormente, de maneira que tinha-se capacidade de definir e alterar o modo de voo a qualquer momento necessário.

O voo foi iniciado no modo *loiter*, decolando-se o veículo e realizando simples manobras manuais. Desde o primeiro momento, já foi visualmente perceptível o aumento na estabilidade do *drone*, que mantinha sua posição com pouca oscilação. Na parte final do primeiro voo, alterou-se o modo para o RTL, que fez o *drone* assumir o controle automático do voo, subir a dois metros de altitude conforme configurado e retornar à posição em que decolou, descendo ao solo com velocidade definida por meio do controlador de voo. Observou-se também que ao entrar em modo autônomo, o VANT realizou manobras mais rápidas e com grande precisão no seu processo de retorno ao local de partida. Os dados de telemetria trazidos em sequência colaboram com uma melhor visualização do comportamento descrito. Como esse não se tratou de um voo completamente autônomo, foi possível conectar o controlador Mission Planner ao *drone* e observar em tempo real alguns parâmetros básicos de seu voo. As Figuras 20a e 20b mostram capturas de parte da tela da estação controle em dois momentos distintos do voo: no primeiro, é possível consultar a altitude do veículo no momento e que o mesmo estava se locomovendo com uma velocidade de 0,58 m/s com relação ao solo; no segundo, o *drone* estava parado no plano das coordenadas, apresentava uma altitude mais baixa e estava em fase de descida, com 0,23 m/s de velocidade em direção ao chão.

A Figura 21 mostra espacialmente o perfil de voo executado pelo *drone*. Nela, nota-se o

Figura 21 – Perfil de voo - Primeiro voo com sistema Vicon®



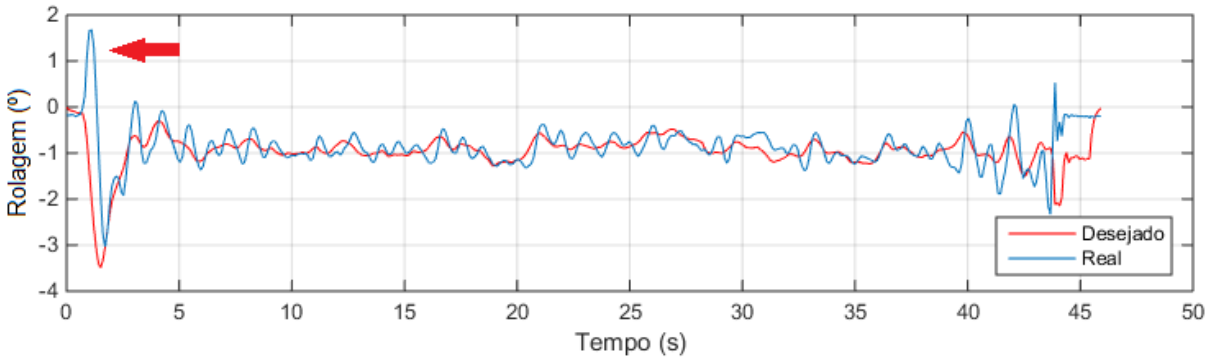
Fonte: o autor (2020)

aumento na suavidade da trajetória a partir do momento em que o modo RTL é definido, logo antes da manobra de subida, em ponto indicado na figura. Também destaca-se a precisão da posição de pouso no retorno ao ponto de decolagem. Tais aspectos apontam que o sistema de posicionamento está operando de forma adequada, enviando dados de GPS ao *drone* com valores pertinentes e uma taxa de transmissão adequada. Da mesma forma, a mudança de comportamento após a mudança de modo de operação traz indícios que um voo autônomo, quando dotado de posicionamento preciso, é mais suave e certeiro que aquele controlado manualmente.

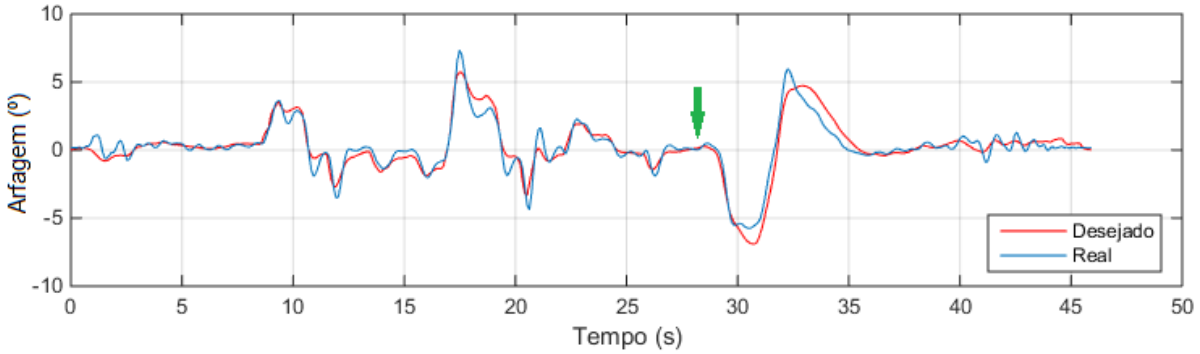
Nas Figuras de 22a a 22c é possível ver os valores de rolagem, arfagem e guinada do *drone*, num comparativo entre o valor desejado e o valor real de cada parâmetro. O valor desejado leva em conta não só a posição atual do *drone* como também qual comportamento o mesmo deve ter para se direcionar ao seu objetivo. Como a trajetória em questão é simples, há pouca variação angular nos dados observados, sendo que a maior delas se dá na arfagem, uma vez que é através da variação da inclinação que o veículo é capaz de se locomover no espaço. De forma geral, o comportamento das curvas reais segue o das curvas desejadas, mas cabem duas ressalvas: no gráfico de rolagem, observa-se que logo no início do voo, mais especificamente na decolagem, há uma oposição entre as curvas, destacada pela seta vermelha, causada por uma pequena instabilidade no alinhamento do *drone* que será discutida na próxima seção; no gráfico de guinada, vê-se que o esperado era manter um ângulo fixo, porém o veículo rodeia o valor desejado com um erro muito pequeno, menor que 1 grau, naturalmente causado por movimentações indiretas causadas pelas manobras do veículo, de forma que não representam um

Figura 22 – Dados de rolagem, arfagem e guinada do *drone* durante o primeiro voo com sistema Vicon® - Comparativo desejado e real

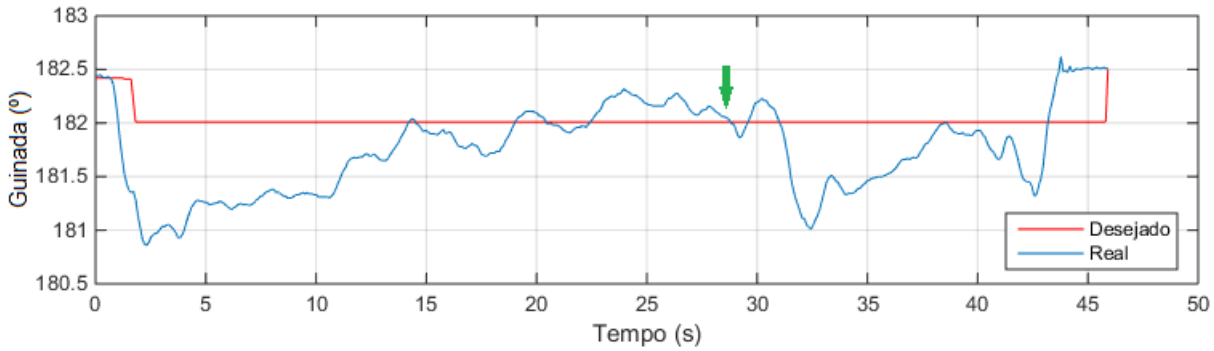
(a) Rolagem



(b) Arfagem



(c) Guinada



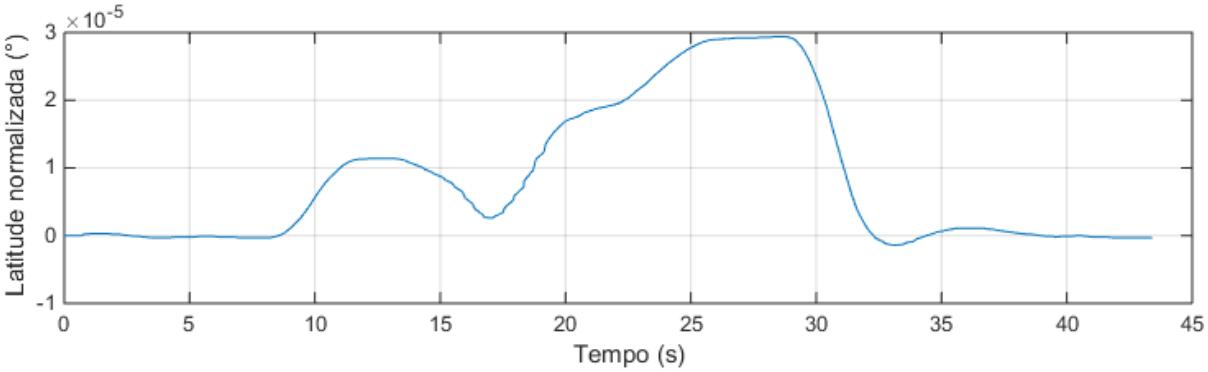
Fonte: o autor (2020)

problema. Também é possível notar com clareza o instante de mudança de modo de voo, tanto na curva de arfagem, quanto na de guinada, destacado pelas setas verdes.

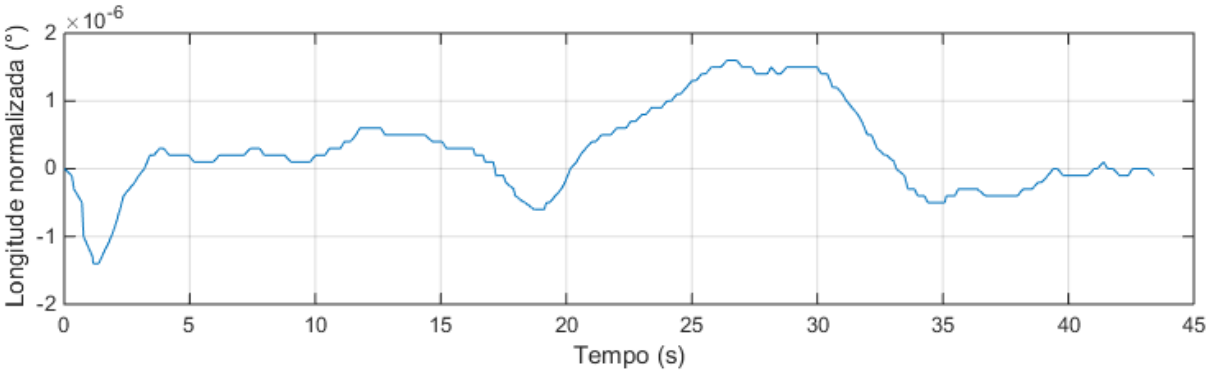
Já nas Figuras 23a e 23b, é esclarecido de que forma um sistema de posicionamento de precisão transmite dados adequadamente disfarçados de GPS. Nelas, vê-se as curvas de latitude e longitude geradas pela biblioteca Pyvicon que descrevem a movimentação do *drone* pelo volume útil do galpão. O destaque aqui é dado para a ordem de grandeza dessa informação de GPS, que

Figura 23 – Coordenadas normalizadas geradas pelo sistema de posicionamento

(a) Latitude



(b) Longitude



Fonte: o autor (2020)

tem variações na escala de 10^{-6} graus nas coordenadas. Graças à precisão disponibilizada para as variáveis do posicionamento via GPS associada à correta conversão da posição $[x, y, z]$ para coordenadas e altitude, a simulação de um GPS extremamente preciso torna-se factível.

O primeiro voo realizado foi a confirmação após o voo preliminar de que o sistema de posicionamento implementado era capaz de promover dados adequados para a navegação do *drone*, sem prejudicar a operação do filtro de Kalman, tornando o VANT ainda mais estável no ar e mostrando-se promissor em momentos de voo autônomo. Concluindo-se tais testes e verificações prévios, prosseguiu-se para a etapa de execução do perfil de voo utilizado em (FARCONI, 2019).

3.2.3 Voo autônomo

Aproximando-se do objetivo do presente trabalho, tratou-se de executar o código implementado para gerar a trajetória com perfil de Lemniscata de Gerono descrita anteriormente. Para o teste em questão, o modo de voo utilizado foi o *guided* (FLIGHT..., 2019), que é o modo mais utilizado em aplicações autônomas. Ao ser habilitado, o modo faz com que o *drone* siga o plano

Figura 24 – *Drone* em pleno voo no galpão da Figura 4a

Fonte: o autor (2020)

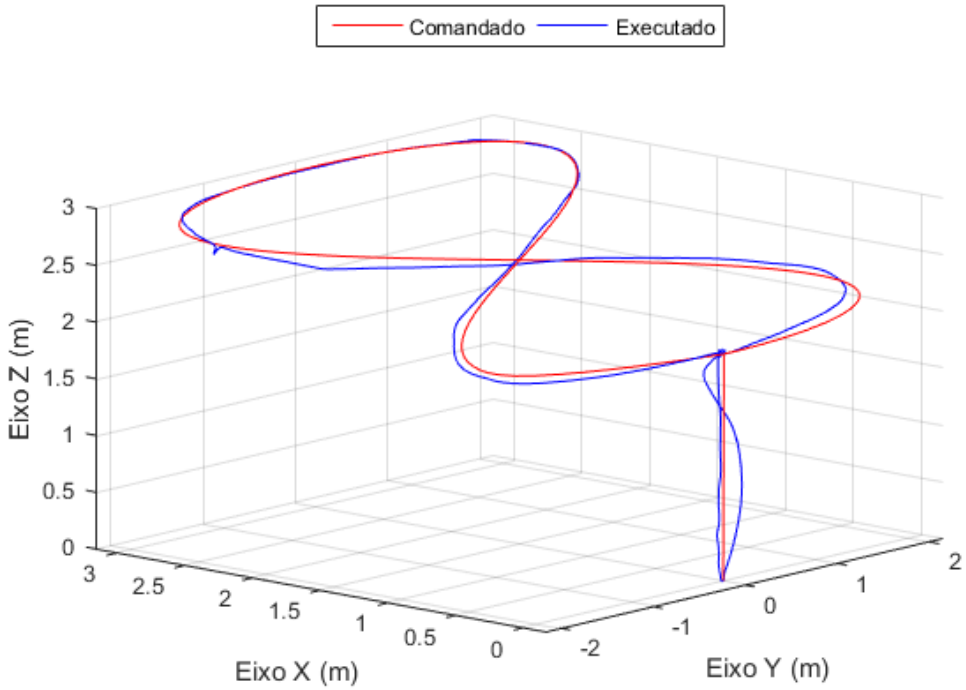
de voo especificado, os *waypoints* definidos ou os comandos enviados. Para fins preventivos, o controle remoto permaneceu conectado ao VANT durante todo o teste com os modos *land* e RTL configurados na chave seletora, uma vez que se tratava de um período mais longo no ar e com uma trajetória que acessava mais regiões do volume útil que ainda não haviam sido percorridas.

Iniciando-se o teste, o *drone* foi posicionado no ponto de partida. Vale destacar que esse ponto é independente da origem do volume útil definida na calibração do sistema Vicon®, uma vez que essa origem servirá apenas para receber o valor da coordenada padrão do controlador MAVProxy, como viu-se na Figura 12. Dessa forma, o veículo marcará o local onde ele foi ligado como sendo seu ponto de partida (*home*), que receberá uma determinada coordenada de acordo com aquela referenciada pela origem do volume. Quaisquer movimentações indicadas pelo código serão feitas tomando por base o *home* do *drone*.

Na sequência, o modo *guided* foi definido e executou-se o código responsável pelos comandos da trajetória. Nele estavam incluídas as ações de armar o *drone*, decolar para atingir uma altitude de dois metros e, a partir desse ponto, iniciar a trajetória da Lemniscata, tudo de forma completamente autônoma. A Figura 24 mostra um registro do veículo durante o voo dentro do galpão anteriormente mostrado e a Figura 25 traz uma visualização tridimensional do perfil de voo realizado, mostrando um comparativo entre a curva que foi efetivamente comandada e a curva que o veículo executou no espaço, enquanto que a Figura 26 ilustra o mesmo comparativo da perspectiva de cada plano ortogonal do sistema de coordenadas. É possível observar que o VANT realiza a trajetória com uma boa precisão, de forma que os maiores valores de erro atingem aproximadamente 13 cm e em breves momentos durante o percurso. Observa-se, também, que o pouso atingiu o local da decolagem com excelente precisão e que o erro de voo analisado nos planos ortogonais foi similar de forma geral, com destaque para o plano x-z, cuja curva mostra variações de altitude mais bruscas que o desejado, principalmente na primeira metade do voo.

O erro visto pode ser associado, entre outras coisas, à forma como a informação de altitude é obtida, considerando que essa grandeza apresentou as maiores imperfeições do perfil

Figura 25 – Perfil de voo Lemniscata - Comparativo comandado e executado

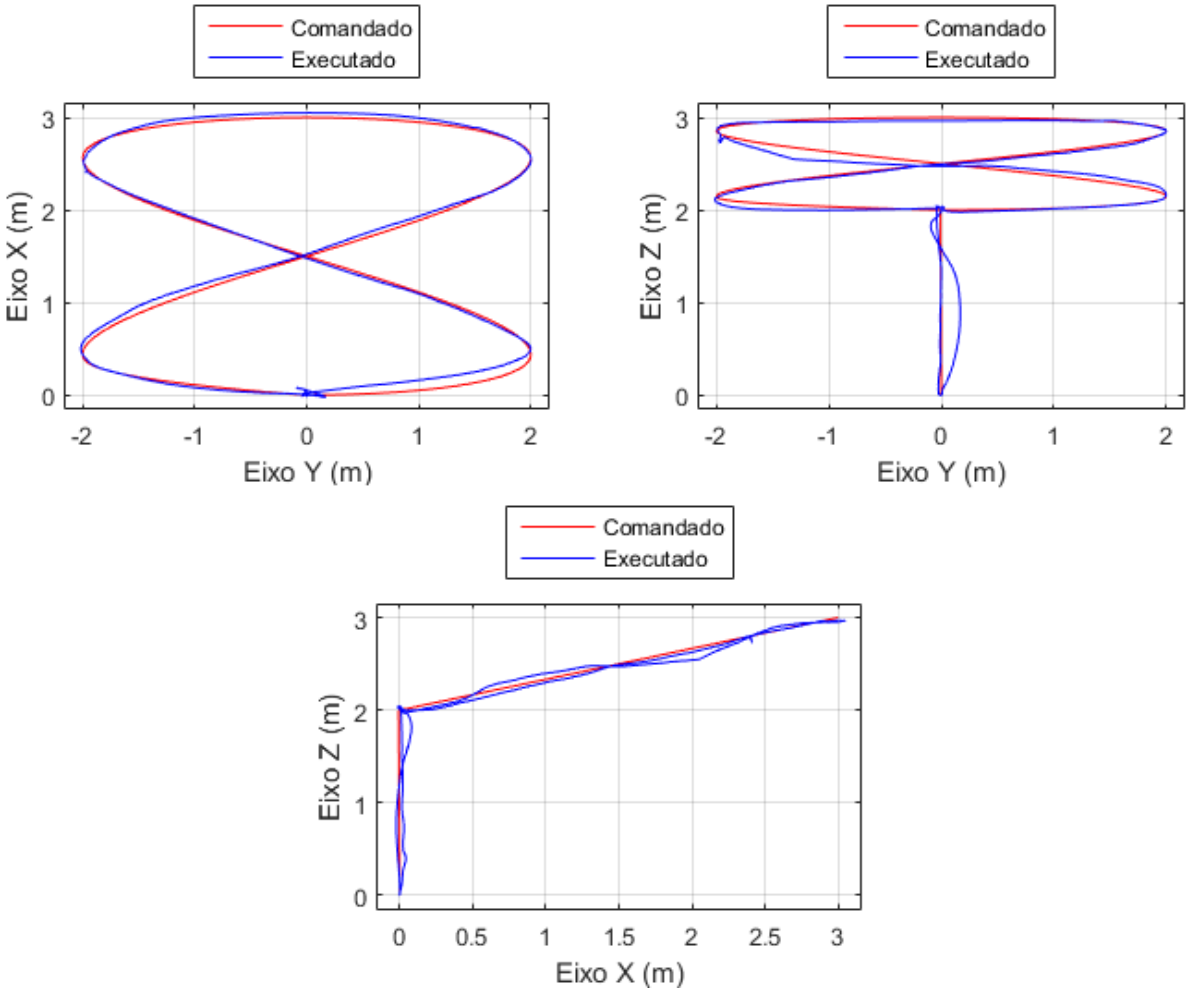


Fonte: o autor (2020)

executado. A altitude é obtida tanto pelo dado de GPS quanto pelo dado do barômetro, um sensor de pressão do qual se calcula a elevação. A questão das duas fontes de medição é que a do barômetro se mostra muito ruidosa frente à do GPS com a aplicação do sistema Vicon®, o que pode acarretar em imprecisões no valor final a ser considerado pelo controlador. A Figura 27 traz um comparativo entre os dados obtidos pelo barômetro e pelo sistema Vicon®, representado pelo GPS. Como os parâmetros sensoriais do voo são tratados pelo filtro de Kalman, variações indevidas nas medições acabam tendo seu impacto reduzido no controle e não representam risco até certo nível de aplicação. Outros pontos a serem considerados são a malha de controle original do *drone* e seu tamanho, uma vez que a lógica de controle não necessita ter tamanha precisão de movimentação num veículo de aproximadamente um metro de diâmetro, visto que espera-se que o mesmo realize voos e manobras guiado por GPS, cuja margem de erro pode chegar a poucos metros (GPS. . . , 2020), e num ambiente onde 13 cm (menos de 15% do diâmetro) não representam um problema.

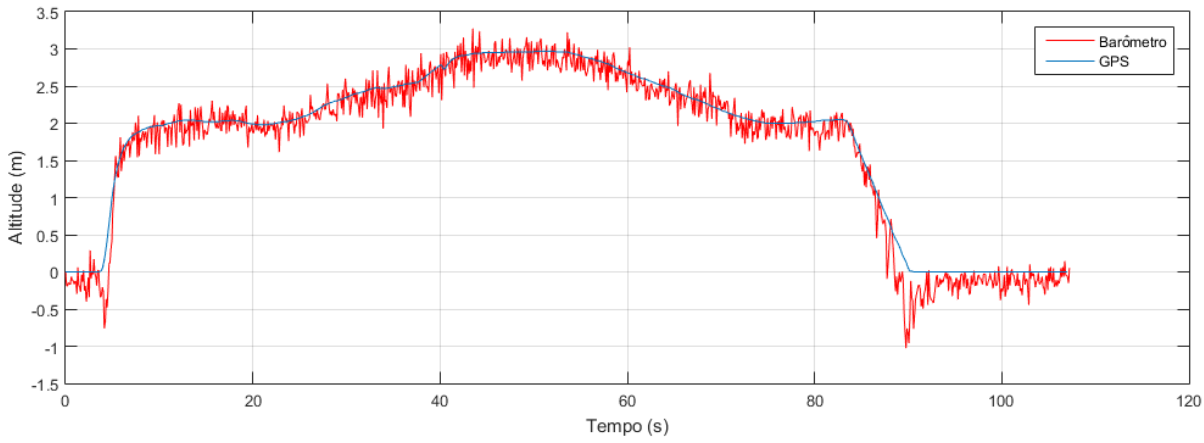
Analisando-se os dados de telemetria vistos nas Figuras de 28a a 28c, nota-se que os valores de rolagem, arfagem e guinada se mantiveram conforme o esperado de maneira geral. A guinada apresentou variações pequenas que surgem em decorrência das outras manobras do veículo, pelo fato de o mesmo não se inclinar perfeitamente em uma única direção. Apesar disso, manteve-se dentro de uma faixa de aproximadamente quatro graus. Na arfagem, destaca-se o pico observado ao redor de $t = 38s$, que será tratado em breve nessa mesma seção. Na rolagem,

Figura 26 – Perfil de voo Lemniscata - Comparativo comandado e executado - Planos ortogonais



Fonte: o autor (2020)

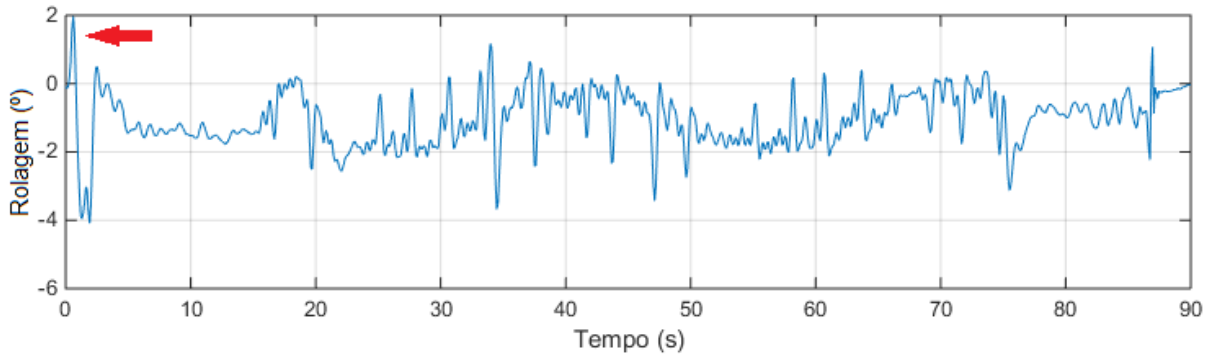
Figura 27 – Altitude do veículo durante trajetória - Comparativo barômetro e GPS (sistema Vicon®)



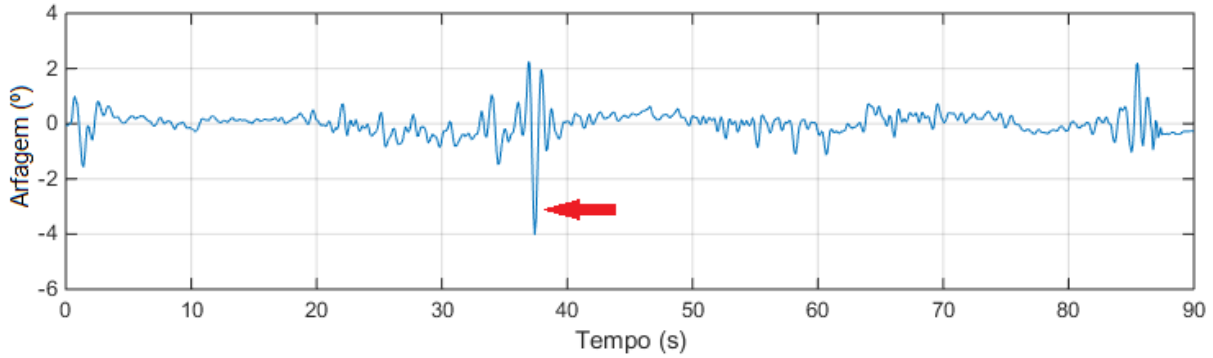
Fonte: o autor (2020)

Figura 28 – Dados de rolagem, arfagem e guinada do *drone* durante o voo com perfil Lemniscata

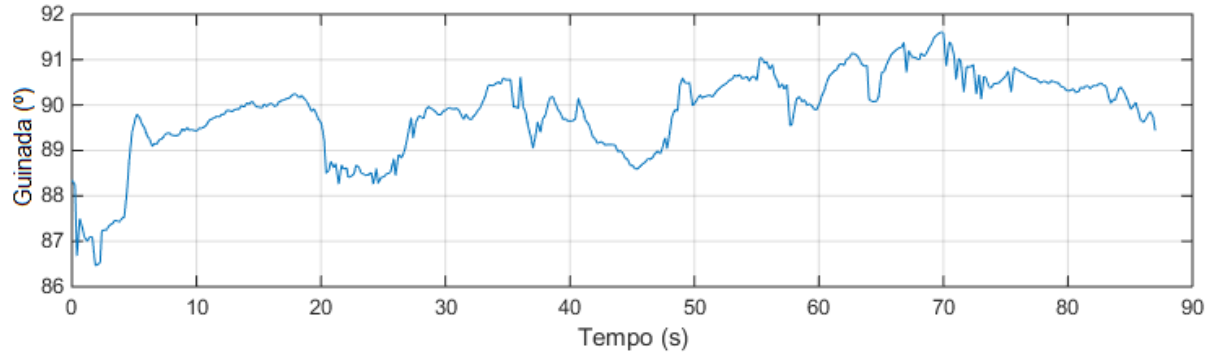
(a) Rolagem



(b) Arfagem



(c) Guinada

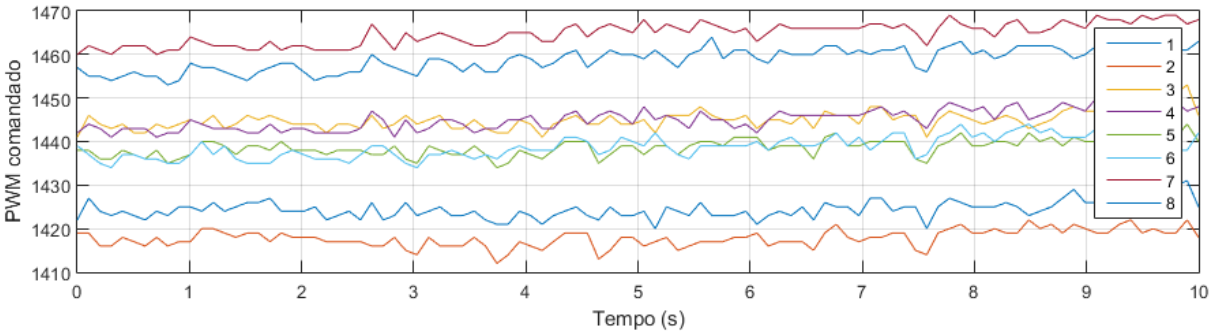


Fonte: o autor (2020)

nota-se um comportamento oscilatório a nível macro que representa a orientação para realizar as manobras de zigue-zague destacadas pela vista do plano y-z (figura 26). Também vê-se no início do voo um comportamento similar ao exposto na seção anterior, na Figura 22a, quando o *drone* inclina no sentido de rolagem positiva e rapidamente retorna no sentido de rolagem negativa, estabilizando logo na sequência e mantendo um valor médio negativo no restante do tempo. A causa desse comportamento na rolagem será explicada a seguir.

Após análise dos dados de telemetria referentes ao PWM comandado para cada rotor,

Figura 29 – PWM comandado ao rotores - Numeração Figura 18



Fonte: o autor (2020)

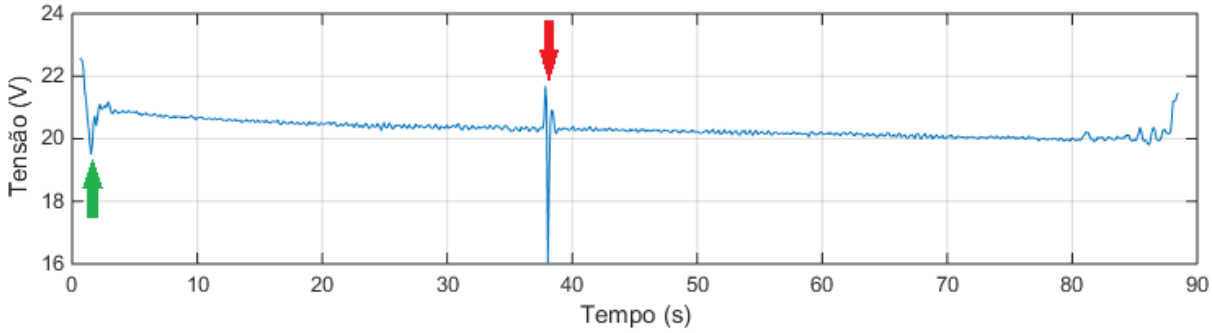
observou-se que quando o veículo está pairando de forma estática no ar, dois motores do lado direito do *drone* recebem comandos com valor acima do esperado e dois motores do lado esquerdo do *drone* recebem comandos com valor abaixo do esperado, conforme ilustra a Figura 29. Isso sugere que os dois motores do lado direito possuem alguma redução de eficiência, de modo que para se manter nivelado, o controlador deve enviar comandos desbalanceados para os rotores. Justifica-se, também, o comportamento observado na curva da Figura 28a, uma vez que o *drone* tenta decolar enviando o mesmo PWM para os oito motores, mas os mais "fracos"do lado direito o fazem girar com rolagem positiva. A atitude de rolagem média negativa predominante no voo registrada pelas Figuras 22a e 28a é explicada da mesma forma, visto que o veículo tem intenção de manter-se rotacionado de um grau para ficar efetivamente nivelado.

Concentrando-se nos dados de telemetria referentes à bateria, obtêm-se informações sobre a tensão de alimentação e o consumo de corrente do veículo que são vistos, respectivamente, nas Figuras 30a e 30b. Primeiramente, é possível observar o aumento rápido no consumo de corrente e, de forma associada, a queda abrupta no nível de tensão no momento da decolagem, e o reestabelecimento aproximado desses valores quando o voo se encerra no pouso. Também é perceptível a gradual redução no nível de tensão da bateria conforme a mesma vai sendo consumida durante o teste.

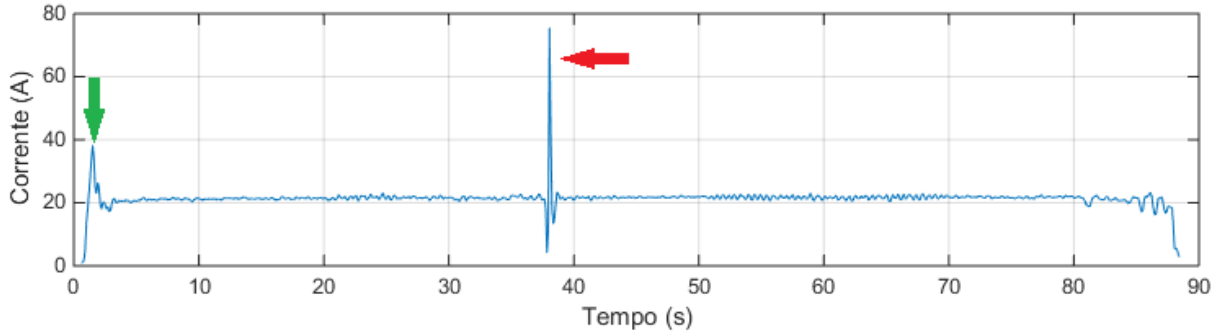
O ponto de destaque aqui é o pico de corrente observado ao redor dos 38 segundos, conforme visto também na Figura 28b, que coincide com a pequena queda de altitude vista na porção esquerda da trajetória da Figura 25, mostrada em destaque pela Figura 31. Em uma análise mais atenta aos dados de telemetria, encontrou-se a origem do comportamento visto na variável que determina a altitude do veículo com relação à origem do sistema de coordenadas em que ele se encontra. Num dado momento, essa variável sofreu um aumento repentino em seu valor, provavelmente causado por algum ruído na composição da informação de altitude, fazendo o veículo acreditar que tinha se elevado levemente. Isso provocou uma redução no PWM comandado ao rotores e, consequentemente, diminuição no consumo total de corrente, levando o veículo a perder altitude. Contudo, o valor da variável citada retornou ao nível devido

Figura 30 – Dados da bateria do *drone* durante o voo com perfil Lemniscata

(a) Tensão

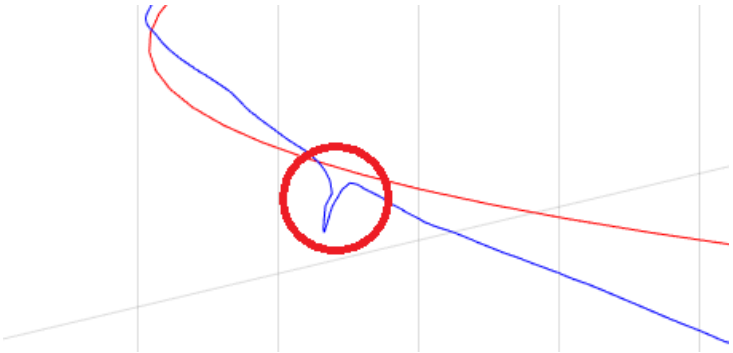


(b) Corrente



Fonte: o autor (2020)

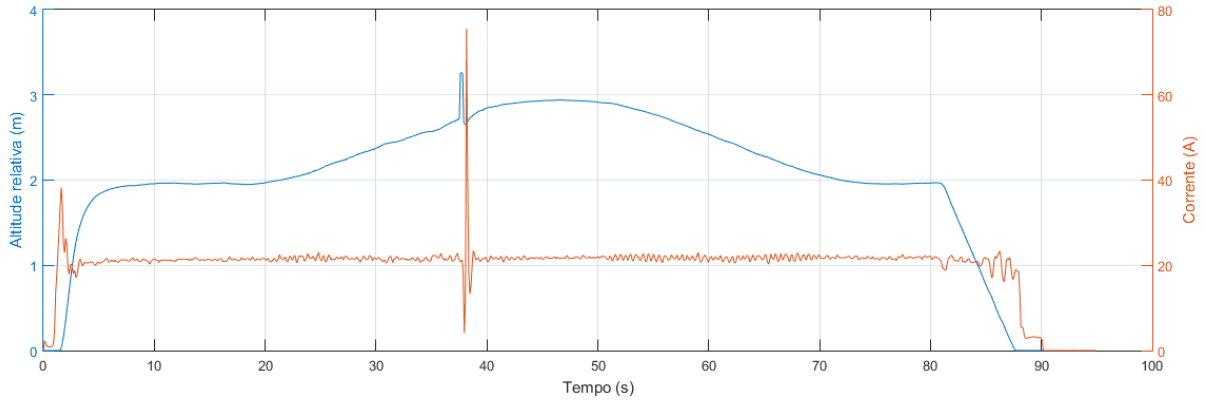
Figura 31 – Destaque em breve falha na trajetória executada com perfil Lemniscata em 60 segundos



Fonte: o autor (2020)

logo na sequência, provocando uma atitude corretiva de retomar sustentação e altitude prévia, ocasionando o pico de corrente e queda de tensão associada que são vistos nas figuras a seguir. Após o acontecido, o *drone* prosseguiu em seu perfil de voo e os níveis de consumo se mantiveram conforme o esperado. A Figura 32 traz as curvas da variável de altitude relativa em relação à origem e do consumo de corrente de forma conjunta de maneira que pode-se observar a sequência de acontecimentos descrita.

Figura 32 – Altitude relativa em relação à origem e consumo de corrente sob mesma escala de tempo



Fonte: o autor (2020)

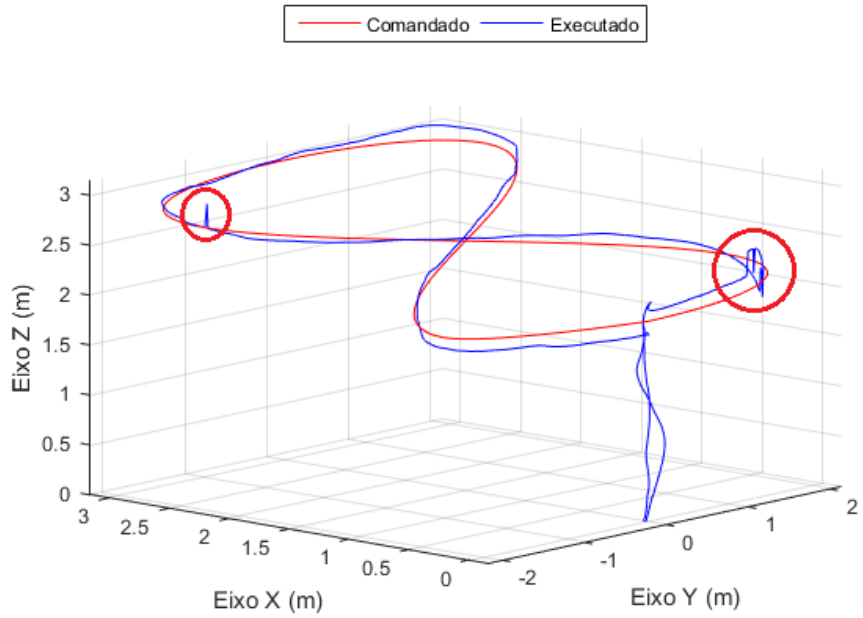
3.2.3.1 Análise comparativa - duração do voo

A fim de se obter dados comparativos, foram realizados no total três voos com o perfil da Lemniscata. Todos contavam com exatamente a mesma trajetória comandada, porém cada um especificava um tempo diferente para a realização da trajetória. No primeiro, o veículo concluiu o percurso (com exclusão da decolagem e do pouso) em 120 segundos, no segundo, em 90 segundos, e no terceiro, em 60 segundos. Como o último surtiu melhores resultados, a análise detalhada feita anteriormente foi acerca dele, mas a comparação entre cada voo realizado traz mais informações válidas.

A Figura 33 e a Figura 34 mostram o caso de 120 segundos. É possível observar uma divergência considerável entre as curvas comandadas e executadas, principalmente no plano y-z e na oscilação na porção direita da trajetória. Também é visto um ponto de erro na medição que causou um pequeno pico de movimentação na trajetória, mas esse não causou efeitos na movimentação real do *drone*. A Figura 35 e a Figura 36 são para o voo caso de 90 segundos. Observa-se a clara melhora na execução do movimento proposto frente ao caso anterior, com um erro menor de maneira geral e com o mesmo erro de medição sem efeitos reais visto no voo de 120 segundos. Conforme já colocadas anteriormente e devidamente tratadas, a Figura 25 e a Figura 26 trazem o voo com duração de 60 segundo e mostram uma precisão ainda melhor no acompanhamento da trajetória desejada, com redução dos erros, principalmente no plano y-z.

A causa desse comportamento mais suave nos perfis mais rápidos pode ter diversas origens, desde a forma como os *waypoints* estão sendo enviados pelo código, até inércia num movimento mais rápido ou forma de operação do controle de trajetória. Porém, deve-se notar que o fato do erro de posição ser reduzido conforme o tempo total de completude se torna menor se mostra promissor frente ao procedimento descrito por (FARCONI, 2019). Nele, o perfil da

Figura 33 – Perfil de voo Lemniscata - Comparativo comandado e executado - Execução 120 segundos



Fonte: o autor (2020)

Lemniscata é realizado num tempo de 15 segundos, quatro vezes mais rápido que o tempo de 60 segundos visto nos resultados analisados. Claramente, há um limite ainda desconhecido para esse comportamento de melhora de performance com a redução do tempo de execução, mas essa análise comparativa acerca da duração do voo contribui com indícios de que o perfil de 15 segundos produzirá resultados satisfatórios.

3.2.4 Contribuição para o modelo de simulação do *drone*

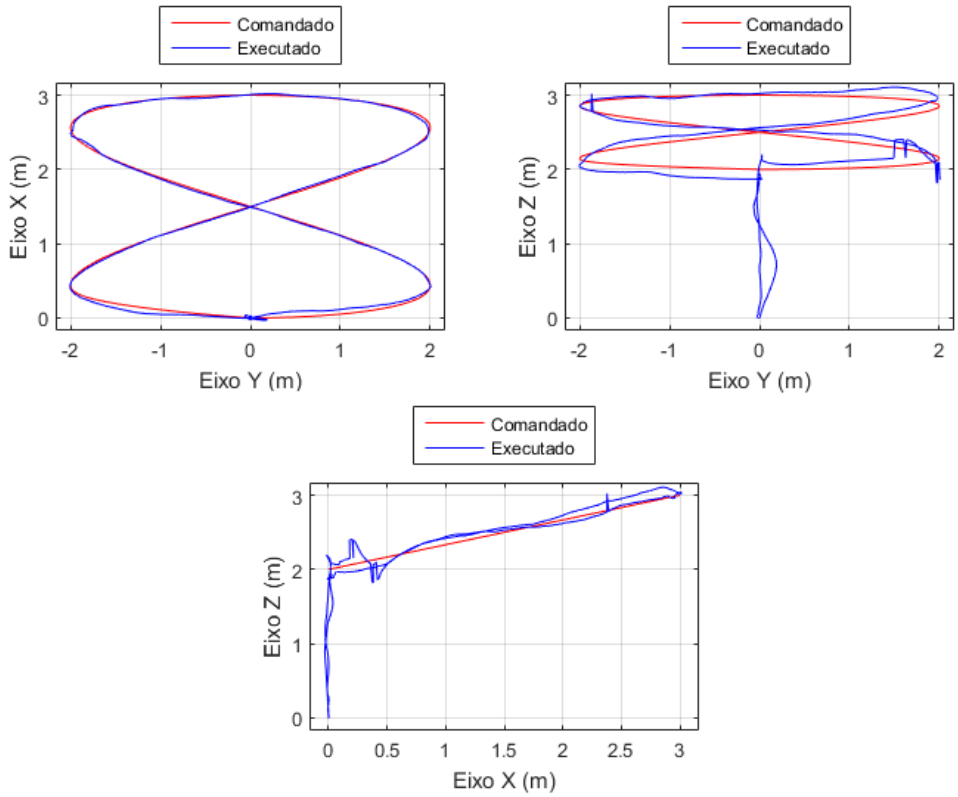
Assim como foi proposto na seção de métodos desse trabalho, os dados de PWM comandado para cada rotor quando o veículo está pairando foram coletados da telemetria e aplicados no equacionamento do método dos mínimos quadrados. Como resultado, obtêve-se os seguintes valores:

$$C_u = 0.0083 \quad \text{e} \quad C_d = 0.0087$$

Conforme dito, os valores absolutos podem variar seu significado de acordo com a parametrização das entradas do sistema, de forma que a informação retirada desse modelo é a perda de eficiência dos motores inferiores frente aos superiores.

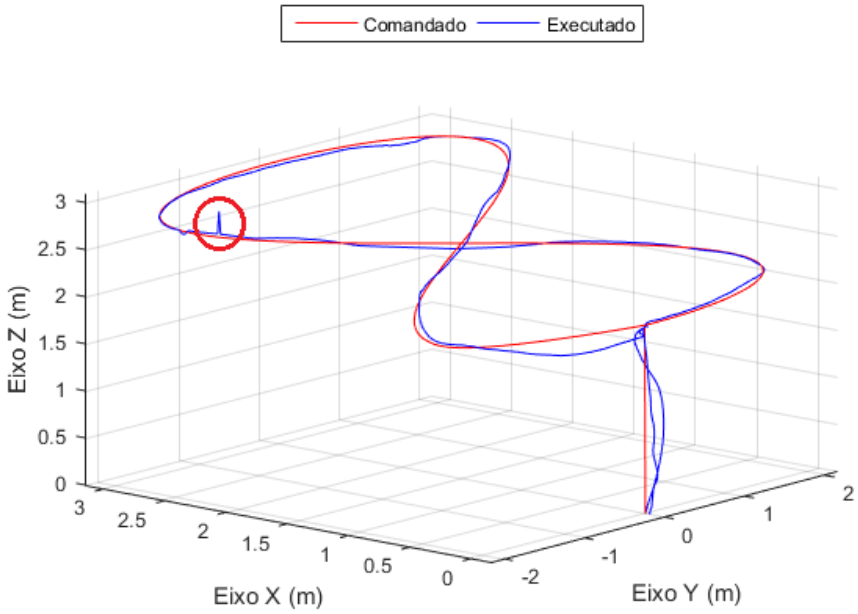
Porém, encontra-se uma inconsistência inicial, uma vez que os valores indicam que a eficiência dos rotores de baixo é maior que dos de cima. Isso decorre do fato de que alguns rotores do *drone* não estão operando em sua eficiência plena por algum motivo operacional, conforme foi explicitado na análise dos dados de rolagem do voo autônomo. Portanto, a análise preliminar

Figura 34 – Perfil de voo Lemniscata - Comparativo comandado e executado - Planos ortogonais - Execução 120 segundos



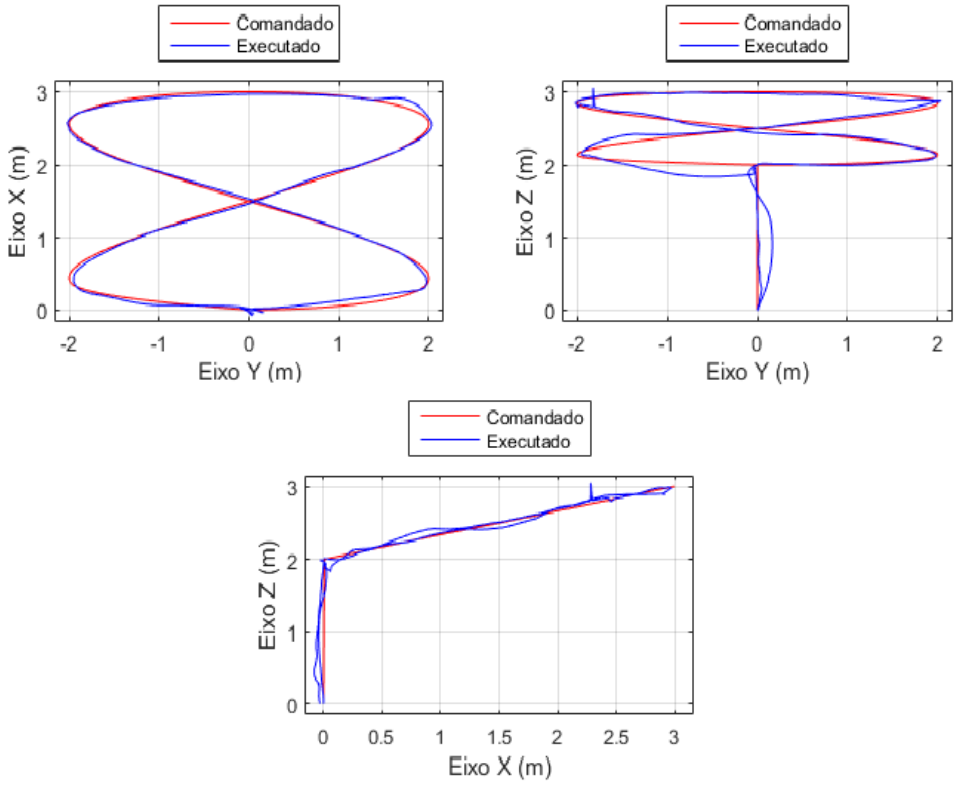
Fonte: o autor (2020)

Figura 35 – Perfil de voo Lemniscata - Comparativo comandado e executado - Execução 90 segundos



Fonte: o autor (2020)

Figura 36 – Perfil de voo Lemniscata - Comparativo comandado e executado - Planos ortogonais - Execução 90 segundos



Fonte: o autor (2020)

proposta, apesar de válida, não é capaz de estimar a relação entre as eficiências propostas devido a outras formas de interferência embutidas nessas grandezas que não são previstas nos equacionamentos dados. Com isso, não é possível tirar conclusões acerca do posicionamento dos rotores com os dados calculados.

4 CONCLUSÃO

Após a análise dos resultados apresentada, conclui-se que o objetivo principal proposto de implementação prática do procedimento de teste foi atingido com êxito. A principal barreira inicialmente encontrada, que dizia respeito à precisão do posicionamento do *drone*, foi transposta com o uso do complexo e preciso sistema Vicon®, atingindo o primeiro objetivo específico definido.

A conectividade do sistema se mostrou suficientemente rápida e robusta para que fosse possível receber, processar e transmitir dados de posição numa taxa elevada e com conexões sem fio por toda a malha de equipamentos que envolviam a configuração montada. A solução Wi-Fi® adotada também foi essencial para a completude da aplicação, que necessitava de rápida transmissão mas sem se prender a cabos, principalmente devido à sua capacidade de efetivar comunicação usando o protocolo MAVLink, o mais adequado ao caso. Com isso, os objetivos de configuração de equipamentos adicionais e de comunicação sem fio também foram alcançados.

A biblioteca Pyvicon, adotada para cumprir o objetivo específico de compatibilização entre Vicon® e veículo, se mostrou uma excelente ferramenta de conversão dos dados, além de trazer um diferencial por ser independente do ROS e permitir uma gama maior de atuações desse procedimento em situações em que não se tem acesso a uma aplicação como o ROS. A forma como a biblioteca se insere no sistema como um todo também lhe confere vantagens, uma vez que seu ponto de comunicação através do MAVProxy é simplificado e conta com diversas funcionalidades auxiliares na sua implementação.

Em termos gerais, o veículo cumpriu de forma satisfatória o perfil de voo que lhe foi imposto, mantendo erros de posição aceitáveis, com valores máximos atingindo cerca de de 15% de seu diâmetro, e descrevendo uma trajetória considerada desafiadora de maneira suave e estável. Os dados de telemetria coletados de todos os testes aéreos corroboram com a afirmação do que o objetivo maior foi cumprido, mostrando dados das mais diversas grandezas que podem ser obtidas, varrendo desde consumo de energia, até perfil comandado e níveis de aceleração.

Também foi possível concluir a respeito de déficits no funcionamento do VANT, observando-se de que forma determinados comportamentos não ideais implicam na maneira como o veículo se comporta nas mais variadas situações de voo. No caso da redução na eficiência de determinados motores, viu-se como o *drone* lida com esse empecilho, comandando durante o voo uma atitude com rolagem corretiva.

A união, portanto, da completude dos objetivos específicos, acarreta na finalização plena do que foi proposto para o presente trabalho.

4.1 Trabalhos futuros

O trabalho futuro que surge de forma direta do presente trabalho é a efetiva aplicação do procedimento de teste real, implementando-o numa avaliação de estratégia de controle aplicada no *drone*. Tendo em vista a teoria e simulação desenvolvida em (FARCONI, 2019) e a metodologia de teste aqui estudada, o próximo passo é a união dos resultados obtidos em ambos para que se avance na direção prática da análise real dos controladores.

Outra frente de trabalho que é criada diz respeito a uma análise aprofundada das eficiências dos rotores no veículo em questão. Buscando atingir o objetivo secundário desse trabalho, concluiu-se que o modelo simplificado apresentado para obter informações sobre efeitos da posição dos motores coaxiais em sua performance não é amplo o suficiente para prever questões como perda de eficiência prévia causada por quaisquer outros fatores. Dessa forma, é possível desempenhar um estudo que leve em conta não só a influência da posição do rotor na eficiência, mas também outros elementos como fabricação e desgaste por uso.

REFERÊNCIAS

COMPLETE parameter list. Ardupilot, 2019. Disponível em: <https://ardupilot.org/copter/docs/parameters.html>. Acesso em: 30 julho 2019.

COPTER motors library. Ardupilot, 2014. Disponível em: <https://ardupilot.org/dev/docs/code-overview-copter-motors-library.html>. Acesso em: 5 março 2020.

CUBE flight controller. Dronecode, 2019. Disponível em: <https://docs.px4.io/v1.9.0/en/flight_controller/pixhawk-2.html>. Acesso em: 20 maio 2019.

DX7S radio system. Spektrum, 2014. Disponível em: <https://www.spektrumrc.com/Products/Default.aspx?ProdId=SPM7800>. Acesso em: 15 julho 2019.

ESP8266 NodeMcu development kit. FilipeFlop, 2019. Disponível em: <https://www.filipeflop.com/produto/modulo-wifi-esp8266-nodemcu-esp-12/>. Acesso em: 5 novembro 2019.

ESP8266 Wi-Fi telemetry. Ardupilot, 2019. Disponível em: <https://ardupilot.org/plane/docs/common-esp8266-telemetry.html>. Acesso em: 28 julho 2019.

EXTENDED Kalman Filter navigation overview and tuning. Ardupilot, 2019. Disponível em: <https://ardupilot.org/dev/docs/extended-kalman-filter.html#extended-kalman-filter-interpreting-log-data>. Acesso em: 20 janeiro 2020.

FARCONI, L. B. **Comparative Analysis of Robust Fault-Tolerant Controllers Applied to Multirotor Autonomous Aerial Vehicles**. 2019. 228 p. Dissertação (Mestrado em Engenharia Elétrica) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

FLIGHT modes. Ardupilot, 2019. Disponível em: <https://ardupilot.org/copter/docs/flight-modes.html>. Acesso em: 13 maio 2019.

GPS accuracy. U.S. Government, 2020. Disponível em: <https://www.gps.gov/systems/gps/performance/accuracy/>. Acesso em: 14 fevereiro 2020.

Lai, T. L.; Robbins, H.; Wei, C. Z. Strong consistency of least squares estimates in multiple regression. **Proceedings of the National Academy of Sciences of the United States of America**, v. 75, n. 7, p. 3034–3036, 1978.

LAWRENCE, J. D. **A Catalog of Special Plane Curves**. New York: Dover Publications, Inc., 1972. 218 p. ISBN 0-486-60288-5.

MAVPROXY. Ardupilot, 2019. Disponível em: <https://ardupilot.github.io/MAVProxy/html/index.html>. Acesso em: 25 junho 2019.

MAVROS package summary. ROS Wiki, 2018. Disponível em: <http://wiki.ros.org/mavros>. Acesso em: 5 maio 2019.

MISSION Planner overview. Ardupilot, 2019. Disponível em: <https://ardupilot.org/planner/docs/mission-planner-overview.html>. Acesso em: 13 maio 2019.

NEGENBORN, R. **Robot Localization and Kalman Filters: On finding your position in a noisy world**. 2003. 146 p. Dissertação (Master of Science in Intelligent Systems) — Institute of Information and Computing Sciences, Utrecht University, Utrecht, 2003.

NODEMCU documentation. NodeMcu, 2020. Disponível em: <https://nodemcu.readthedocs.io/en/master/>. Acesso em: 6 março 2020.

PYVICON - python 3 wrapper over Vicon DataStream SDK (1.7.0+). Mathieu Garon, 2018. Disponível em: <https://github.com/MathGaron/pyvicon>. Acesso em: 13 agosto 2019.

RAMASAMY, M. Measurements comparing hover performance of single, coaxial, tandem, and tilt-rotor configurations. **AHS 69th Annual Forum - Phoenix, AZ**, v. 4, p. 2439–2461, 01 2013.

ROS documentation. ROS Wiki, 2018. Disponível em: <http://wiki.ros.org/>. Acesso em: 5 maio 2019.

RUN test mavros. Leonardo Farçoni, 2020. Disponível em: <https://github.com/lbf10/ardupilot_ws/blob/master/src/multirotor_test/src/run_test_mavros.cpp>. Acesso em: 15 março 2020.

SIK telemetry radio. Ardupilot, 2019. Disponível em: <https://ardupilot.org/copter/docs/common-sik-telemetry-radio.html>. Acesso em: 26 setembro 2020.

VICON calibration documentation. Vicon, 2019. Disponível em: <https://docs.vicon.com/display/Nexus25/Calibrate+a+Vicon+system>. Acesso em: 17 setembro 2019.

VICON documentation: Go further with vicon max t-series. Vicon, 2010. Rev-1.3. Disponível em: <https://docs.vicon.com/display/LegacyCamDoc/Legacy+cameras+documentation>. Acesso em: 15 julho 2019.

VICON indoor positioning system. Ardupilot, 2019. Disponível em: <https://ardupilot.org/copter/docs/common-vicon-for-nongps-navigation.html>. Acesso em: 28 julho 2019.

VICON motion capture. Vicon, 2019. Disponível em: <https://www.vicon.com/>. Acesso em: 1 julho 2019.

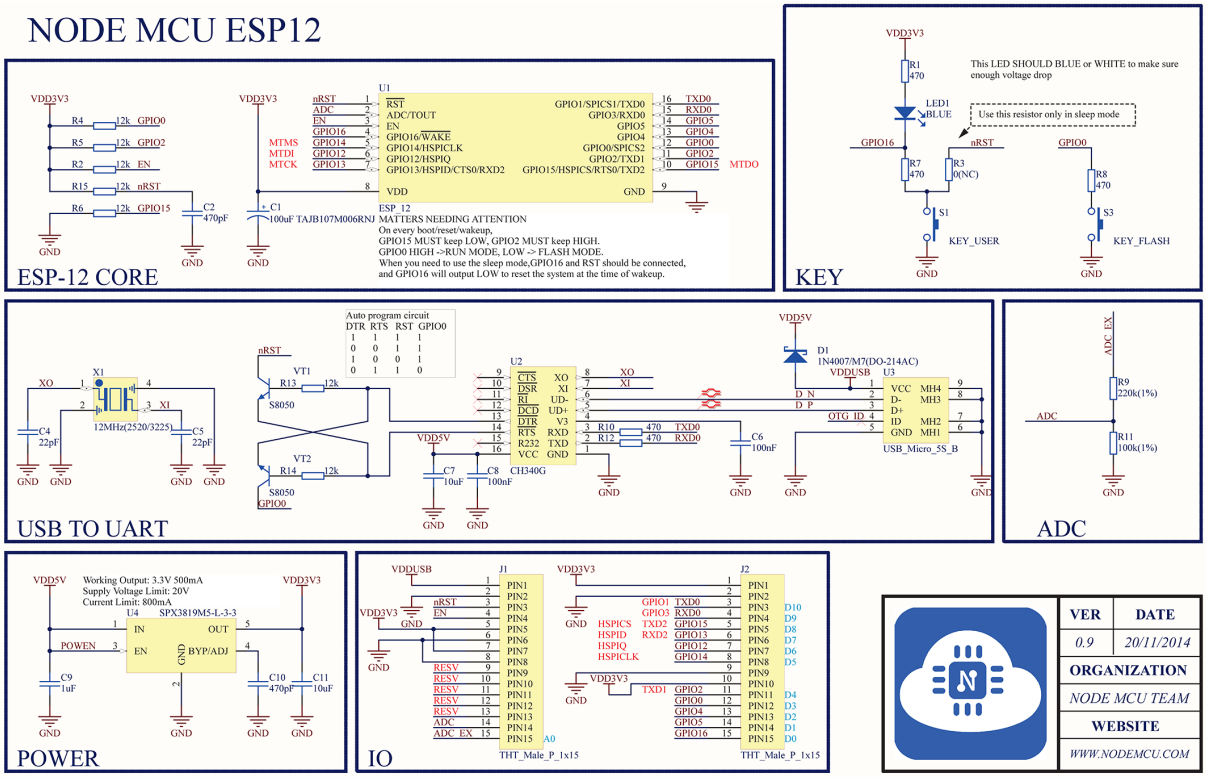
WEISSTEIN, E. W. Eight Curve. From MathWorld–A Wolfram Web Resource, s.d. Disponível em: <https://mathworld.wolfram.com/EightCurve.html>. Acesso em: 27 março 2020.

Zanni, L. et al. A prediction-error covariance estimator for adaptive kalman filtering in step-varying processes: Application to power-system state estimation. **IEEE Transactions on Control Systems Technology**, v. 25, n. 5, p. 1683–1697, 2017.

Anexos

ANEXO A – DIAGRAMA DO CIRCUITO DO MÓDULO WI-FI

Figura 37 – Diagrama do circuito do kit de desenvolvimento LoLin ESP8266



Fonte: (NODEMCU. . . , 2020)

ANEXO B – CÓDIGO EM C++ PARA EXECUÇÃO DAS TRAJETÓRIAS AUTÔNOMAS

```
#include "mavros/mavros.h"
#include <geometry_msgs/PoseStamped.h>
#include "mavros_msgs/CommandBool.h"
#include "mavros_msgs/CommandTOL.h"
#include "mavros_msgs/CommandLong.h"
#include "mavros_msgs/State.h"
#include <cstdlib>
#include <chrono>
#include <thread>
#include "traj_gen.h"

using namespace std;

// Declare drone state message
mavros_msgs::State current_state;

// Callbacks %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
// State message callback
void readStateCallback(const mavros_msgs::State::ConstPtr& msg)
{
  current_state = *msg;
}

// Main loop %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
int main(int argc, char **argv)
{
  ros::init(argc, argv, "run_test");
  ros::NodeHandle nh;

  traj_gen tg;

  // Declare Services %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  ros::ServiceClient arming_client =  nh.serviceClient
  <mavros_msgs::CommandBool>("mavros/cmd/arming");
```

```
ros::ServiceClient takeoff_client =  nh.serviceClient
<mavros_msgs::CommandTOL>("mavros/cmd/takeoff");
ros::ServiceClient command_client =  nh.serviceClient
<mavros_msgs::CommandLong>("mavros/cmd/command");

// Nickname Services/Messages %%%%%%%%%%%%%%%%%%%%%%%%%%

mavros_msgs::CommandBool arming_srv;
mavros_msgs::CommandTOL takeoff_srv;
mavros_msgs::CommandLong command_srv;

// Start
ROS_INFO("Initializing...");

// Declare subscriptions %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ROS_INFO("Subscribing to State msg");
ros::Subscriber sub =
nh.subscribe("/mavros/state", 10, readStateCallback);
ROS_INFO("Subscribed!");

// Declare publishers %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ROS_INFO("Creating position set point publisher");
ros::Publisher local_pos_pub =
nh.advertise<geometry_msgs::PoseStamped>("/mavros/setpoint_
position/local", 10);
ROS_INFO("Created!");

//the setpoint publishing rate MUST be faster than 2Hz
ros::Rate rate(2.0);

// wait for FCU connection
ROS_INFO("Waiting for FCU connection");
while(ros::ok() && !current_state.connected){
    ros::spinOnce();
    rate.sleep();
}
ROS_INFO("Connected!");

// Example setpoint
```

```
geometry_msgs::PoseStamped pose;
pose.pose.position.x = 0;
pose.pose.position.y = 0;
pose.pose.position.z = 0;

//TRAJETORIA GERONO
int steps=8;
MatrixXd wp_matrix(12,steps);
ArrayXd wp_times(steps);

double takeOffTime = 5;
double takeOffAltitude = 2;
double geronoDuration = 60;
tg.geronoToWaypoints(3, 4, 1, geronoDuration, steps, goTo,
0*M_PI,wp_matrix,wp_times);

MatrixXd aux(wp_matrix.rows(),wp_matrix.cols()+2);
aux << ArrayXd::Zero(wp_matrix.rows()).matrix(),
ArrayXd::Zero(wp_matrix.rows()).matrix(),wp_matrix;
aux.row(2) = (aux.row(2).array()+takeOffAltitude).matrix();
wp_matrix = aux;

ArrayXd aux1(2+wp_times.size());
aux1 << ArrayXd::Zero(2).array(), wp_times;
aux1 = aux1+2*takeOffTime;
aux1(0) -= takeOffTime;
wp_times = aux1;

//TRAJETORIA TESTE
// MatrixXd wp_matrix(12,4);
// wp_matrix << 0, 0, 0, 0,
//              0, 1.5, -1.5, 0,
//              2, 2,  2, 2,
//              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;
// ArrayXd wp_times(4);
// wp_times << 10, 20, 40, 50;

cout << "Matriz de waypoints =
" << endl << wp_matrix << endl;
cout << "Array de tempos =
" << endl << wp_times << endl << endl;

Array3d initialPosition;
initialPosition << 0,0,2;
tg.setTrajectory(wp_matrix,wp_times,initialPosition,
Array3d::Zero(),0,0);

Vector3d position;
Vector3d velocity;
Vector3d acceleration;
double yaw;
position << 0.0,0.0,0.0;
velocity << 0.0,0.0,0.0;
acceleration << 0.0,0.0,0.0;

local_pos_pub.publish(pose);

// Arm
ROS_INFO("Arming...");
command_srv.request.broadcast = 1;
command_srv.request.command = 400;
command_srv.request.confirmation = 1;
command_srv.request.param1 = 1;
command_srv.request.param2 = 2989;
command_client.call(command_srv);
if(command_srv.response.success){
        ROS_INFO("Armed!");
}
else{
        ROS_INFO("Arm fail!");
```

```
        ROS_INFO("End");
        return 0;
}
sleep(3);

// Takeoff
ROS_INFO("Taking off...");
takeoff_srv.request.latitude = initialPosition(0);
takeoff_srv.request.longitude = initialPosition(1);
takeoff_srv.request.altitude = initialPosition(2);
takeoff_srv.request.min_pitch = 0;
takeoff_srv.request.yaw = 0;
takeoff_client.call(takeoff_srv);
if(takeoff_srv.response.success){
        ROS_INFO("Lift off!");
}
else{
        ROS_INFO("Take off fail!");
        ROS_INFO("End");
        return 0;
}
sleep(3);

auto t_ini = chrono::steady_clock::now();
auto t_now = t_ini;
double t = 0;
float quaternion[4];
ROS_INFO("Publishing set points.");
while(ros::ok()){
    t_now = chrono::steady_clock::now();
    t = chrono::duration_cast<chrono::milliseconds>(t_now -
    t_ini).count()/1000.0;
    tg.desiredTrajectory(t, position, velocity, acceleration,
    yaw);
    cout << "Position command = " << position.transpose() <<
    " " << yaw << endl;
    pose.pose.position.x = position(0);
    pose.pose.position.y = position(1);
    pose.pose.position.z = position(2);
```

```
        mavlink_euler_to_quaternion(0,0,yaw,quaternion);
        pose.pose.orientation.w = quaternion[0];
        pose.pose.orientation.x = quaternion[1];
        pose.pose.orientation.y = quaternion[2];
        pose.pose.orientation.z = quaternion[3];

        local_pos_pub.publish(pose);
        ros::spinOnce();
        rate.sleep();
    }

    ROS_INFO("Stopped");

    ros::spin();

    ROS_INFO("End");

    return 0;
}
```

Fonte: (RUN..., 2020)