FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Construção e Integração de um Quadcopter numa Plataforma de Simulação de Missões Multi-Veículo

Leonardo Silva Ferreira

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Prof. Doutor Daniel Castro Silva

Co-Orientador: Mestre Álvaro Luiz Panarra das Neves Câmara

31 de Julho de 2018

Construção e Integração de um Quadcopter numa Plataforma de Simulação de Missões Multi-Veículo

Leonardo Silva Ferreira

Mestrado Integrado em Engenharia Informática e Computação

31 de Julho de 2018

Resumo

Nos últimos anos, os veículos autónomos, nomeadamente os Veículos Aéreos Não Tripulados (VANTs ou UAVs, do inglês *Unmanned Aerial Vehicle*), têm crescido a um ritmo acelerado devido ao enorme leque de funcionalidades que podem desempenhar no terreno. Esta procura por novos caminhos e soluções para este tipo de aeronave tem gerado grandes esperanças em setores como aplicações militares, aplicações agrícolas, missões de salvamento, telecomunicações ou combate a desastres, entre outras. São várias as áreas em que o uso de VANTs é benéfico e é esperado que surjam ainda mais no futuro próximo. Todavia, nem todas as soluções presentes no mercado são fáceis de gerir, sendo algumas bastante dispendiosas, não só em termos de custo monetário como também na complexidade que acarretam.

Neste sentido, o objetivo deste trabalho por apresentar uma alternativa mais simples, completa e principalmente mais barata comparativamente com as disponíveis no mercado, através da criação e integração de um quadcopter (um dos tipos de VANTs) numa plataforma de simulação de missões multi-veículo, ou seja, numa primeira fase monta-se o quadcopter de forma a poder voar e depois representa-se, de uma forma realista, a movimentação do mesmo na plataforma de simulação. Para tal, procurou-se montar um quadcopter com base nas especificações pretendidas pelo projeto e com o material disponível e configurar o seu controlador de voo onde se tem em conta as funcionalidades que o mesmo deve desempenhar. Após esta fase de montagem e configuração do quadcopter, fez-se uma análise ao voo do quadcopter. Uma vez finalizada esta etapa, foi necessária a integração desta componente numa plataforma de simulação já existente. Esta plataforma proporciona uma configuração flexível de missões e cenários de teste, que permite observar o seu desempenho em diversas áreas de investigação e com diferentes tipos de veículos (terrestres, aquáticos e aéreos). Com esta integração na plataforma de simulação, observou-se que o *digital twin* (modelo que representa o quadcopter real no ambiente simulado da plataforma), foi capaz de realizar todo o tipo de movimentos aplicados no quadcopter real (*yaw*, *pitch* e *roll*). Para além disso, a plataforma de simulação foi equipada com a possibilidade de enviar comandos ao quadcopter real. Para tal, foram respeitadas as normas redigidas pela NATO (*North Atlantic Treaty Organization*) descritas no documento *STANAG 4586*.

Com a montagem e integração do quadcopter na plataforma de simulação terminadas recorreu-se a dois tipos de testes: individuais e coletivos. Os testes individuais têm como propósito estudar e analisar cada um dos sensores utilizados (sonar, GPS e giroscópio) enquanto que os coletivos têm a missão de observar o comportamento (interno e externo à plataforma de simulação) que os sensores podem demonstrar quando emparelhados uns com os outros e o voo do quadcopter. A avaliação dos resultados obtidos nos testes tem em conta várias métricas como rapidez e distância de comunicação, tempo de calibragem, desempenho em cenários abertos e fechados, entre outras. Em termos individuais, grande parte componentes utilizados revelaram resultados satisfatórios. O tempo de calibragem do GPS (mede latitude e longitude) em cenários externos revelou-se inferior aos cenários externos. Os sinais do giroscópio (mede a orientação) variaram como expectável com as diferentes movimentações (*pitch*, *roll* e *yaw*) e o sonar (mede a altitude) apresentou algumas li-

mitações quando perante inclinações superiores a 45° o que demonstra não ser viável para terrenos com relevo acidentado. Quanto aos testes coletivos, apenas foi possível manter o quadcopter no ar durante um curto período de tempo devido à presença de material defeituoso. Já na integração o objetivo passou por avaliar-se o tempo entre o fim da movimentação real e o fim da movimentação virtual sendo que os resultados poderiam ter sido melhores.
Espera-se que, com todo este processo, seja possível estudar e analisar a interação entre VANTs reais e simulados através de missões cooperativas de diversos tipos. Este projeto pode ajudar o futuro com novas aplicações e ideias que possam ajudar a sociedade a progredir a diferentes níveis. Áreas onde se procura a total substituição do piloto durante missões de risco, são um bom exemplo do potencial destes veículos.

Palavras Chave: *Unmanned Aerial Vehicles* (UAVs), *Unmanned Aerial System* (UAS), simulação, quadcopter, aplicações, comunicação, controlador de voo, plataforma, veículos autónomos

Abstract

In recent years, autonomous vehicles, in particular Unmanned Aerial Vehicles (UAVs), have grown at a rapid pace due to the huge range of applications they can perform on the ground. This search for new routes and solutions for this type of aircraft has created big hopes in sectors such as military applications, agricultural applications, rescue missions, telecommunications or disaster fighting, among others. There are several areas where UAVs are beneficial and are expected to emerge even more in the near future. However, not all the solutions on the market are easy to manage, some of which are quite costly, not only in terms of money costs but also in the complexity that they entail.

In this regard, the objective of this work is to provide a simpler, more complete and mainly cheaper alternative to those available in the market, by assembling and integrating a quadcopter (one of the types of UAVs) into a multi-vehicle mission simulation platform, that is, in a first phase the quadcopter is assembled in order to be able to fly and then it is represented, in a realistic way, the movement of the same in the platform of simulation. In order to accomplish this, the assembly of the quadcopter was done based on the specifications required by the project and the material available and configuration of his flight controller where we take into account the features that the same should perform. After the assembly and configuration of the quadcopter, the flight analysis of the quadcopter was performed. Once this step was completed, it was necessary to integrate this component into an existing simulation platform. This platform provides a flexible configuration of missions and test scenarios, allowing you to observe the performance in various research areas and with different types of vehicles (land, sea and air). With this integration, the simulation platform, namely *digital twin* (model representing the actual quadcopter in the simulated platform environment), is able to perform all type of movements applied in the real quadcopter (yaw, pitch e roll). In addition, the simulation platform was equipped with the possibility of sending commands to the real quadcopter. To do this, the standards written by NATO (North Atlantic Treaty Organization) described in STANAG 4586 were respected.

With the assembly and integration of quadcopter in the simulation platform, two types of tests were used: individual and collective. The individual tests are designed to study and analyze each of the sensors used (sonar, GPS and gyroscope), while the collective ones have the mission of observing the behavior (internal and external to the simulation platform) that the sensors can demonstrate when paired with the others and the quadcopter flight. The evaluation of the results obtained in the tests takes into account several metrics such as speed and distance of communication, calibration time, performance in open and closed scenarios, among others. In individual terms, much of the components used showed satisfactory results. The GPS calibration time (measures latitude and longitude) in external scenarios is lower than external scenarios. The signals of the gyroscope (measure orientation) varied as expected with the different movements (*pitch*, *roll* and *yaw*) and the sonar (measures altitude) presented some limitations when facing inclinations higher than 45°, which proves that is not a viable component for irregular terrain. As for the collective tests, it was only possible to keep the quadcopter in the air for a short period of time (3 to 4 seconds) due to the

presence of defective material. In the integration, the objective was to evaluate the time between the end of the actual movement and the end of the virtual movement and the results could have been better.
It is hoped that with all this process, it is be possible to study and analyze the interaction between real and simulated UAVs through cooperative missions of various types. This project can help the future with new applications and ideas that can help society move forward at different levels. Areas where full pilot replacement is sought during risk missions are a good example of the potential of these vehicles.

Keywords: Unmanned Aerial Vehicles (UAVs), Unmanned Aerial System (UAS), simulation, quadcopter, applications, communication, flight controller, platform, autonomous vehicles

Agradecimentos

Reservo este espaço para todos aqueles que me ajudaram e apoiaram enquanto estudante da Faculdade de Engenharia da Universidade do Porto (FEUP). A realização desta dissertação contou com vários apoios aos quais estou eternamente agradecido.

Começo por agradecer ao Professor Doutor Daniel Castro Silva e ao Mestre Álvaro Câmara pela orientação, opiniões e críticas, palavras de incentivo, calma transmitida e sobretudo pelo total apoio e disponibilidade que me dedicaram ao longo do desenvolvimento deste trabalho.

Gostaria também de agradecer aos meus amigos mais próximos André, Luís, Edgar, Diana e Ana pelas experiências e grandes momentos que passamos nos últimos anos. A vossa presença e apoio, durante este meu percurso académico, foi essencial na realização deste sonho.

Por último e não menos importante, deixo um profundo agradecimento à minha família, em especial aos meus pais, irmã, avós e namorada por todos os sacrificios que tomaram. Agradeço todo o apoio incondicional, força de superação, paciência e incentivo dado. Sem estas premissas não teria conseguido terminar este projeto. Por tudo o que representam na minha vida, é a vós que dedico este trabalho!

Leonardo Ferreira

"And once the storm is over, you won't remember how you made it through, how you managed to survive. You won't even be sure, whether the storm is really over. But one thing is certain. When you come out of the storm, you won't be the same person who walked in. That's what this storm's all about."

Haruki Murakami

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

AAN	Autoridade Aeronáutica Nacional
ANAC	Autoridade Nacional da Aviação Civil
APM	*ArduPilot Mega*
ARM	*Advanced Risc (Reduced Instruction Set Computer) Machine*
AUVSI	*Association for Unmanned Vehicle Systems International*
CLP	Controlador Lógico Programável
EASA	Agência Europeia de Segurança da Aviação
ESC	*Electronic Speed Controller*
FPGA	*Field Programmable Gate Array*
FPV	*First Person Vision*
FSX	*Flight Simulator X*
GCS	*Ground Control Station*
GPIAA	Gabinete de Prevenção e Investigação de Acidentes com Aeronaves
GPIO	*General Purpose Input/Output*
GPS	*Global Position System*
GSM	*Global System for Mobile Communications*
HDL	*Hardware Description Language*
HIL	*Hardware in Loop*
IMU	*Inertial Measurement Unit*
ISA	*International Standard Atmosphere*
NATO	*North Atlantic Treaty Organization*
ODE	*Open Dynamics Engine*
PID	Proporcional Integral e Derivativo
PPM	*Parts per Million*
PWM	*Pulse-Width Modulation*
RC	Controlador Rádio
ROS	*Sistema Operacional de Robôs*
RPA	Aeronave Remotamente Pilotada
SMA	Sistema MultiAgente
SIL	*Software in Loop*
TCP	*Transmission Control Protocol*
UAS	*Unmanned Aerial System*
UAVs	*Unmanned Aerial Vehicle*
UDP	*User Datagram Protocol*
UVS	*Unmmaned Vehicle Systems*
VANT	Veículos Aéreos Não Tripulados
VLOS	*Visual line of sight*
VTOL	*Vertical Take-off and Landing*

Capítulo 1

Introdução

Desde o século XIX, altura que marcou o início do uso de VANTs (Veículos Aéreos Não Tripulados) especialmente em conflitos armados [Newcome, 2005], foram desenvolvidos novos métodos e tecnologias que permitiram projetar o futuro para novos horizontes. Com a evolução tecnológica que se tem sentido, os VANTs têm obtido cada vez mais preponderância e relevância no nosso quotidiano oferecendo vastas vantagens mas também riscos que colocam em causa o bem-estar da sociedade. Tem-se a intenção de explorar e estudar no capítulo atual tópicos como quais os objetivos, necessidades e problemas que podem surgir no presente e futuro no que toca ao paradigma dos VANTs, tendo especial foco no modo como estes nos podem beneficiar.
Dado a sua maior proliferação e reconhecimento, ao longo deste documento, será utilizada a sigla UAV (do inglês *Unmanned Aerial Vehicle*) em alternativa a VANT.

1.1 Contexto e Motivação

Graças aos avanços da Aeronáutica nos últimos anos, são várias as empresas e outras entidades que têm procurado investir e explorar os veículos aéreos não-tripulados (UAVs). Estes veículos são meios excepcionais que possuem características únicas (tamanho e funcionalidades) e que permitem executar um variado conjunto de tarefas. Contudo, levantam diversas críticas e questões legais, seja a nível civil ou militar [Newcome, 2005], o que acaba por muitas das vezes levar as pessoas a terem uma perceção errada ou pouco esclarecida sobre os UAVs (o regulamento destes veículos é referido na secção 2.2. Apesar destas limitações, os UAVs são bastante úteis na sociedade e desempenham um papel importante em vários países. Os Estados Unidos são um exemplo disso, desde a sua utilização no passado, na 2ª Guerra Mundial [Haulman, 2003], até à elaboração de planos para o melhoramento da defesa do país [United States Department of Defense , 2005], que procuram automatizar processos e aumentar a segurança. No entanto, não são apenas os Estados Unidos que olham para este setor e vêm um mundo interminável de oportunidades.
Independentemente destes fatores, uma das razões pelas quais existe tanto interesse nesta área é o

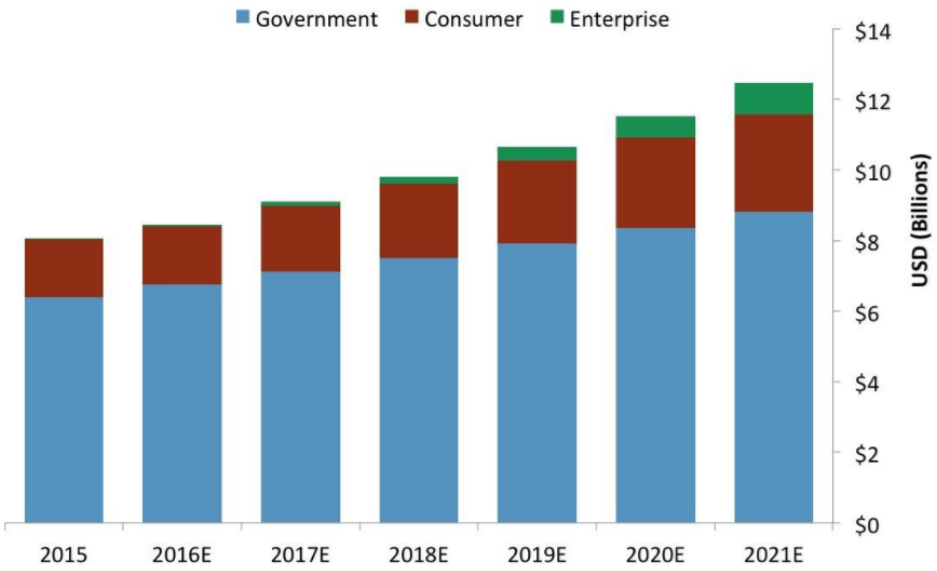


Figura 1.1: Investimento estimado para hardware de UAVs até 2021 [Joshi, 2017]

facto de esta se constituir como uma das alternativas para o auxílio do piloto durante o voo, como por exemplo na aterragem e descolagem, e pelo facto de poder, por exemplo, entrar em ambientes que um piloto em condições normais não conseguiria. Em alguns casos pretende-se mesmo que os UAVs substituam os pilotos de forma a reduzir o risco que lhes é apresentado durante diversos tipos de missões[Austin, 2010]. No capítulo 2, será feita uma análise mais detalhada sobre como estão distribuídas as aplicações que os UAVs podem desempenhar e em que níveis podemos tirar partido dos mesmos.

Quanto ao seu mercado, de acordo com a Stratistics [Stratistics Market Research Consulting, 2017], o mercado global dos UAVs está estimado em 2015 com 5,93 mil milhões de dólares americanos e é expectável que atinga os 22,15 mil milhões de dólares em 2022, apresentado um crescimento total de 20,7%. Apesar de o mercado dos UAVs ainda se encontrar nos primeiros estágios, a tendência é que este progrida a vários níveis com o decorrer dos anos (como se observa na Fig. 1.1). A crescente utilização de UAVs em aplicações militares para combater o terrorismo e automatização no processo de auxílio a pilotos são os principais fatores que impulsionam este mercado, enquanto que fatores como os regulamentos legais podem atrasar o mesmo. Outro fator desafiante a ter em conta são os acidentes dos UAVs, como por exemplo o caso da queda de quatro UAVs Predator[Observador, 2017] entre 2015 e 2016. Além de aplicações militares, a expansão dos UAVs também se tem sentido em outras áreas, nomeadamente Comercial, Pessoal e Tecnologia futura [Joshi, 2017].

Assim sendo, sejam os UAVs (a diferenciação dos UAVs é feita na secção 2.1.1) controlados remotamente (com rádios controldadores) ou acedidos por uma aplicação smartphone (qualquer plataforma), eles possuem a capacidade de chegar às áreas mais remotas com pouca ou nenhuma força de trabalho necessária e requerem o mínimo de esforço, tempo e energia. Esta é uma das

grandes razões que justificam estes resultados.

1.2 Objetivos

Este trabalho tem como principal missão a construção e adaptação de um quadcopter numa plataforma de simulação para missões multi-veículo. Como resultado, deve ser possível estudar a interação e o comportamento entre os UAVs reais e simulados. Este projeto faz parte de um trabalho de maior dimensão, cuja missão visa a utilização de veículos autónomos para a realização de missões de vigilância, de busca e salvamento, gestão do espaço aéreo, restauração de comunicações em situações de desastre entre outras. Para que tal seja possível e para que tenhamos uma melhor perceção sobre o mesmo, o projeto foi dividido em duas fases.
Construção do Quadcopter e Configuração de um controlador de voo para o mesmo:

- Construção do Quadcopter - Montar o hardware do quadcopter com base naquilo que será o seu papel e com especial atenção às especificações da plataforma a ser integrado. A montagem será feita com recurso a diversas peças de diferentes características e funções.

- Controlador de voo - Existem diversos controladores de voo para quadcopters disponíveis no mercado; no entanto, muitos deles acabam por ser bastante custosos e complexos para o público em geral. Assim sendo, será adaptado um controlador de voo que permite controlar o quadcopter durante o voo sem comprometer o mesmo, respeitando os objetivos e regras idealizadas. Serão tidos em conta vários pacotes de softwares para controlo de voo com o intuito de encontrar aquele que melhor se adeque aos objetivos do projeto.

- Funcionalidades do Quadcopter - O quadcopter deverá ser capaz de alterar a altitude durante o voo, aumentar/diminuir a velocidade do voo, entre outras. Opcionalmente, e dependendo da disponibilidade temporal para tal, podem ser implementadas funcionalidades adicionais, como seguir rotas pré-estabelecidas, ter em conta o input do utilizador ou até mesmo reconhecer rotas inteligentes (o quadcopter, dependendo da carga que possui ou de outros fatores, opta por abortar o voo ou seguir um caminho alternativo).

Adaptação do controlador de voo do Quadcopter à plataforma de simulação de missões multi-veículo:

- Protocolos de comunicação - São bastante importantes neste género de tecnologias porque fazem a ponte entre o quadcopter e a plataforma de simulação. Contudo, é preciso ter em conta quais as situações com que o quadcopter se deparará no futuro, visto que o desempenho dos protocolos de comunicação varia consoante as propriedades do ambiente em que se encontra (calor, humidade, precipitação, entre outros fatores). Serão analisadas tecnologias como GSM (Sistema Global para Comunicações Móveis), Rádio, Infravermelho, Bluetooth, comunicação via microondas, WI-FI entre outras.

- Configurar a plataforma para a integração de agentes UAVs - É necessário adaptar a plataforma de simulação de missões multi-veículo de modo a corresponder às especificações do

projeto e para que se possa testar a interação e comunicação entre a mesma e quadcopter. Uma vez cumprido este passo, a plataforma deve ser capaz de aceitar UAVs simulados.

- Preparar o quadcopter e a plataforma de simulação para a introdução de sensores - Com a introdução de vários sensores no quadcopter, este e a plataforma de simulação devem estar preparados para recolher e interpretar os dados recolhidos nos mesmos. Uma vez recebidos os dados pela plataforma de simulação, o quadcopter simulado deve saber interpretar estes dados e mover-se consoante os mesmos. O quadcopter simulado representa o quadcopter real na plataforma de simulação.

- Preparar o quadcopter, estação de controlo e a plataforma de simulação para a introdução de comandos - A plataforma de simulação deve ser capaz de enviar vários tipos de comandos ao quadcopter real. Estes representam manobras e ações que o quadcopter deve ser capaz de interpretar e executar e podem ter várias naturezas. Englobam aumentar/diminuir altitude, mover para frente/trás, rodar e ligar motores.

- Testar a interação entre os veículos simulados e reais - Após a ligação entre controlador de voo e plataforma de simulação, serão feitos testes para verificar se o sistema funciona corretamente e como esperado. Serão feitas as correções que possam ser necessárias e tiradas conclusões tendo em conta os resultados obtidos. Estes resultados advêm do estudo do comportamento dos UAVs em diferentes cenários.

1.3 Estrutura do Relatório

O presente capítulo atual, esclarece qual o contexto e motivação, objectivos e aplicações que os UAVs podem desempenhar no quotidiano. No segundo capítulo 2, é dada ênfase ao UAVs onde se explica uma série de conceitos, tipos e classificações relacionados com o tema. Para além de se estudar quais os protocolos de comunicação a utilizar nos UAVs, tendo em conta o seu propósito e condições de voo, é ainda estudado um conjunto de leis e máximas nacionais e europeias que sintetizam as normas que os UAVs devem cumprir no seu normal funcionamento. No capítulo 3, é feita uma análise sobre o estado da arte destes veículos autónomos ao nível de componentes de quadcopters, plataformas de simulação prontas a usar, software existente para controladores de voo e abordagens semelhantes à que se pretende implementar durante este processo, ou seja, conjugar o controlador de voo de um UAV numa plataforma de simulação, de forma a poderem ser retiradas conclusões sobre qual a melhor abordagem a seguir. No capítulo 4, dá-se destaque aos requisitos tecnológicos a respeitar no decorrer do desenvolvimento deste projeto, ao hardware e software a utilizar na montagem e configuração do quadcopter para o voo e integração na plataforma de simulação, abordagem ao problema e ainda o plano de trabalho seguido O capitulo 5 começa por analisar a arquitetura do sistema pré-existente acompanhada com uma nova arquitetura que permite a introdução de UAVs no sistema. De seguida é explicado tudo o que compõe o voo do quadcopter, ou seja, o material utilizado, o processo de montagem seguido assim como

a configuração do software do mesmo. No que toca à integração do quadcopter na plataforma de simulação são abordados os agentes UAV e fluxo que a informação segue desde o quadcopter até à plataforma de simulação. Explica-se qual o material, processo de montagem e configuração do software a seguir e implementar no quadcopter e na estação de controlo terrestre. Por último é referido como transformar os dados reais em dados virtuais, quais os comandos utilizados e quais os modos disponíveis na plataforma de simulação de forma a se poder testar as funcionalidades do quadcopter real e da mesma. No capítulo 6, são referidos os resultados obtidos durante os testes feitos aos componentes e ao voo do quadcopter. Também são mencionados que condicionantes podem surgir no voo do quadcopter. No capítulo 7, tira-se as devidas conclusões sobre o trabalho realizado no presente projeto. Sáo dadas algumas sugestões para trabalho futuro que podem permitir a introdução de novas funcionalidades no sistema assim como melhorias em algumas componentes desenvolvidas.

Capítulo 2

Conceitos Fundamentais

Nos últimos anos, a popularidade dos UAVs aumentou drasticamente, perspectivando-se que continue a espalhar-se pelo mercado e pelos diversos países. O facto de ser uma tecnologia bastante flexível e versátil em termos de aplicações e materiais, torna-a numa tecnologia a explorar no futuro. De acordo com o estudo feito pela Associação Internacional de Sistemas de Veículos Não Tripulados (AUVSI, do inglês *Association for Unmanned Vehicle Systems International*) [Association for Unmanned Vehicle Systems International, 2013], é estimado que em 2025, o mercado de UAVs tenha um impacto de 82 mil milhões de dólares americanos sobre a economia dos Estados Unidos, contribuindo ainda para a criação de 100 mil postos de trabalho.
Este capítulo será dedicado a dar algumas noções importantes sobre várias matérias relativas aos UAVs. Serão explorados características, tipos, classificações, movimentos, hardware (componentes físicos dos UAVs), legislação imposta aos UAVs (nacional e europeia) assim como possíveis meios de comunicação.

2.1 UAVs e principais características

Estes veículos também conhecidos por drones (termo mais vulgar), UAS (*Unmanned Air/Aircraft System*) ou RPAS (*Remotely Piloted Aircraft System*, termo profissional mais utilizado no mundo tecnológico) representam todo o tipo de aeronave que não necessita de piloto para ser coordenada. Possuem três eixos que permitem aos mesmos executar vários géneros de movimentos (explorado mais à frente) e são controlados à distância com recurso a equipamentos eletrónicos e computacionais, supervisionados pelo ser humano ou por CLPs (Controladores lógicos programáveis).
São normalmente equipamentos pequenos e leves compostos por fibra de carbono, metal (pouco) e materiais plásticos. Existem vários géneros de UAVs (mais informação abaixo), sendo que os mais comuns possuem quatro pequenos motores elétricos, cada um localizado em cada uma das extremidades do UAV. Estes motores podem possuir ou não hélices (depende do género de UAV),

que estão responsáveis por assegurar o voo do UAV. Quanto ao fornecimento de energia, este pode ser feito através de uma bateria ou combustível dependendo do tipo de UAV.
Estes veículos servem-se também de uma placa lógica que contém o sistema de navegação e controlo. Esta placa possui um sistema que recebe instruções do utilizador (na maioria das vezes, através de um controlador rádio), interpreta-as e transmite comandos aos motores, aumentando ou diminuindo a aceleração e altitude. Grande parte dos UAVs disponíveis no mercado estão aptos para a inclusão de câmara, receptor GPS, entre outros componentes.

2.1.1 Tipos de UAVs

São várias as opções e combinações que o mercado oferece aos utilizadores. Dependendo do objetivo do utilizador para com o UAV, podem ser encontrados vários tipos de equipamentos. Os UAVs podem ser classificados em diversas vertentes.

Em termos de Hélices:

(a) Exemplo de um Quadcopter (quatro motores)[1]

(b) Exemplo de um UAV com único motor[2]

(c) Exemplo de um UAV de asa fixa[3]

Figura 2.1: Exemplos de UAVs tendo em conta o número de hélices

UAVs Multi-Motor: Representado na Fig. 2.1a e está dotado com vários motores. Por providenciarem um voo mais estável, são muito usados em áreas como fotografia, vigilância, entre outras. Podem ainda ser sub-divididos tendo em conta o número de motores de que dispõem: tricopter (três motores), quadcopter (quatro motores), hexacopter (seis motores) e octacopter (oito motores). De entre os mencionados, aquele que se pretende montar para este projeto é o quadcopter.

[1]Fonte: http://www.rc-drones.com/Rc-Logger-EYE-One-Xtreme-Brushless-Quadcopter-Mode-2-RTF_p_1061.html

[2]Fonte: https://dronelife.com/2016/07/13/australian-rescuers-will-put-drones-shark-patrol/

[3]Fonte: https://skydrones.com.br/zangao-v/

UAVs de Motor Único: Apresentam uma estrutura semelhante aos helicópteros, como se observa na Fig. 2.1b embora com um tamanho bastante mais reduzido. São compostos por um único motor e uma cauda (não conta como motor pricipal). São mais eficientes que os anteriores, por poderem voar durante mais tempo, embora estejam impossibilitados de executar movimentos variados e rápidos como os anteriores.
UAVs de Asa Fixa: Utiliza asas semelhantes às de um avião, como se observa na Fig. 2.1c, e utiliza apenas dois motores. É energeticamente mais viável que os restantes modelos devido à sua aerodinâmica. Apresenta um tempo de voo bastante longo mas requer um custo elevado para a sua obtenção e manutenção. Para além disso, necessita que o seu utilizador tenha algum treino para que o UAV possa voar em boas condições.

Em termos de Tamanho:

(a) Exemplo de um Nano UAV[4]

(b) Exemplo de um Micro UAV[5]

(c) Exemplo de um Mini UAV[6]

(d) Exemplo de um UAV Medio[7]

(e) Exemplo de um UAV Grande[8]

Figura 2.2: Exemplos de UAVs tendo em conta o seu tamanho

Nano UAV: Utilizados maioritariamente em operações militares de infiltração ou vigilânica apesar da sua fragilidade. Apresenta uma estrutura semelhante à Fig. 2.2a com dimensões na ordem dos 10 cm.

Micro UAV: São bastante pequenos, com cerca de 50 cm, especializados em missões de espionagem. Têm a disposição mostrada na Fig. 2.2b. Geralmente, estes veículos não necessitam de licença para a sua utilização.

Mini UAV: Apresentam um tamanho entre 50 cm e 2 m e pouca potência devido ao seu tamanho. Apresentam-se como na Fig. 2.2c e podem ser construidos com asa fixa ou hélices. São bastantes parecidos com os micro UAVs. Apresenta-se como sendo melhor que o nano UAV em termos de motores e funcionalidades.

UAV Médios: Apesar de não ser percetivel na Fig. 2.2d são mais pesados que os anteriores, conseguindo transportar cargas elevadas (até 200 kg), mas apresentam um tempo de voo reduzido (tipicamente entre 5 a 10 minutos).

UAV Grandes: Geralmente são utilizados para fins militares, nomeadamente, em missões de bombardeamento. Em termos de tamanho são comparados a aviões de grande porte. Apresentam-se como na Fig. 2.2e sendo bastante utilizados pelos Estados Unidos.

Como é possível constatar, são vários os tipos de UAVs presentes no mercado tendo em conta o tamanho e número de hélices.
A Tabela 2.1 mostra informações sobre os vários tipos de UAVS tais como principais características e quais os cenários de aplicação.

2.1.2 Classificações dos UAVs

De acordo com a UVS International (*Unmmaned Vehicle Systems International*)[UVS International, 2012]. responsável pela coordenação e cooperação internacional no que toca a sistemas pilotados remotamente, os UAVs podem ser classificados em nove categorias.

[4]Fonte: https://www.suasnews.com/2014/10/prox-dynamics-introduces-night-capable-pd-100-black-hornet/
[5]Fonte:https://www.microdrones.com/en/
[6]Fonte:http://www.dronenerds.com/products/specials/topproducts/drones/dji-spark-mini-drone-sky-blue-cp-pt-000733-dji.html
[7]Fonte:https://alquilerdrones.eu/alquiler-phantom-4-pro/
[8]Fonte:https://www.cgtrader.com/3d-models/aircraft/military/predator

Categoria	Peso (kg)	Altitude máx (m)	Autonomia (h)	Alcance (km)
Micro	5	250	1	<10
Mini	<30	150/250	2-4	10
Medium Range (MR)	150-500	5000	6-10	70-200
MR Endurance (MRE)	500-1500	8000	10-18	>500
Low Altitude Deep Penetration (LADP)	250-2500	9000	0,5-1	>250
Medium Altitude Long Endurance (MALE)	1000-1500	8000	24-48	>500
High Altitude Long Endurance (HALE)	2500-5000	20000	24-48	>2000
Unmaned Combat AV (UCAC)	>1000	12000	+/-2	+/-1500
Decoys (DEC)	150-500	5000	<4	500

Tabela 2.1: Classificações dos UAVs tendo em conta peso, altitude máxima, autonomia e alcance[Al-Kaff et al., 2018]

Estas categorias estão inseridas noutras categorias de maior relevo como missões de combate ou vigilância, entre outras. Os UAVs do tipo médio encaixam na categoria *Medium Range* enquanto que os de tipo grande entram nos *Medium Altitude Long Endurance*.

2.2 Legislação

Nos últimos tempos, têm sido relatados vários incidentes com o uso de UAVs, especialmente entre estes e aviões nas imediações de aeródromos e aeroportos.
Em 2016, segundo a GPIAA (Gabinete de Prevenção e Investigação de Acidentes com Aeronaves)[Gabinete de Prevenção e Investigação de Acidentes com Aeronaves, 2012], foram registados cerca de 31 incidentes com UAVs nos aeroportos nacionais. Existiram casos em que um UAV levou ao cancelamento temporário da descolagem de um avião, tendo condicionado outras operações nas pistas do aeroporto, ou até mesmo outras situações que levaram aviões a desviarem-se dos UAVs quando se preparavam para aterrar no aeroporto.
Embora um UAV possa ser encarado como um brinquedo, pode trazer grandes problemas caso a sua utilização seja feita de forma incorreta, podendo mesmo colocar em perigo tanto o utilizador como aqueles que o rodeiam. Apesar de um UAV não ser suficiente para provocar um acidente de grande magnitude (despenhamento de um avião), existe sempre uma preocupação acrescida devido ao número de passageiros que um avião transporta.
São este género de incidentes que têm colocado em causa o potencial destes veículos autónomos, e como resposta a estes acidentes, novas regras e regulamentos têm sido propostos com o intuito de limitar e obter um maior controlo sobre aquilo que podem fazer.

2.2.1 Regulamento Português

A legislação e os regulamentos nacionais relativamente aos UAVs são coordenados por duas entidades: ANAC [9] (Autoridade Nacional da Aviação Civil) e, AAN [10] (Autoridade Aeronáutica Nacional). O regulamento [Agência Nacional de Aviação Civil, 2016] em vigor foi aprovado no dia 24 de Novembro de 2016 pela ANAC e pelo governo português.
No mesmo, é apresentada uma lista de regras que devem ser cumpridas pelos utilizadores. Assim sendo, os UAVs:

- Devem apenas efetuar voos diurnos, em operações VLOS (linha de visão com o UAV) com uma altitude máxima de 120 m acima da superfície, à exceção das aeronaves brinquedo, que não devem exceder os 30 m de altura;
- As operações destes devem ser feitas de modo a reduzir os riscos para as pessoas, bens e outras aeronaves;
- O piloto remoto deve manter-se afastado e dar prioridade às aeronaves tripuladas;
- Ser utilizados a uma distância segura para com as pessoas e bens de modo a evitar acidentes;
- Não devem ser utilizados quando o seu utilizador não se encontra na posse máxima das suas capacidades físicas e mentais e quando se encontra sob o efeito de substâncias psicoativas (substâncias químicas que afetam o sistema nervoso);
- Devem verificar se os mesmos se encontram em boas condições para realizarem o voo;
- Não podem ser pilotados em operações VLOS quando perante vários veículos autónomos (apenas com a autorização da ANAC); Devem voar com as luzes de identificação ligadas, independentemente de ser um voo diurno ou noturno;
- Caso exista mais de um observador a auxiliar o piloto remoto, estes devem poder conseguir estabelecer comunicações, por qualquer meio ao seu dispor;
- Não podem voar sobre multidões (menos de 12 pessoas podem), salvo se expressamente autorizado pela ANAC;
- Não podem voar em zonas de sinistro onde se encontrem a decorrer operações de proteção e socorro, salvo se expressamente autorizado pelo comandante das operações de socorro;
- Não podem voar, salvo se expressamente autorizado pela ANAC, numa área próxima (1 km) de heliportos (hospitalares, sob gestão, comando ou responsabilidade de entidades públicas e para missões de proteção civil).

[9] Fonte:http://www.anac.pt/
[10] Fonte:http://www.aan.pt/

Respeitar estas regras é bastante importante, mas não suficiente. Antes de planear um voo, é imperativo ter a consciência que existem dois tipos de licenças, uma ligada à ANAC e outra relativa à AAN, com propósitos diferentes.
Caso o plano de voo inclua realizar uma operação que não obedeça a algum dos pressupostos acima enunciados, é necessário pedir uma licença obrigatória à ANAC para proceder ao voo.
A licença, acessível no site Voa na Boa[11], contém um formulário com os seguintes campos: identificação do requerente da licença, identificação do(s) piloto(s), especificações, características e tipo de UAV, sistemas de estabilização redundante, tipo de operação, identificação da área de operação e análise de risco. Deve ser enviado com 12 dias de antecedência por carta ou email. A emissão desta declaração é gratuita.
Caso o plano de voo inclua realizar uma operação com um UAV equipado com uma câmara, é preciso uma licença obrigatória da AAN, mesmo que não se pretenda fazer a captura de imagens. Esta licença, também acessível em [11], contém um formulário bastante semelhante ao anterior, com a diferença de que esta necessita de mais detalhes sobre o voo e informação adicional relativa à captura de imagens: tipo de recolha, equipamentos utilizados, propósito, autorização de terceiros.

É possível identificar quais as zonas disponíveis e indisponíveis para a pilotagem de UAV. A ANAC disponibiliza uma aplicação denominada Voa na Boa, que permite detetar se uma zona está restrita ou disponível para uma operação de voo com UAVs.

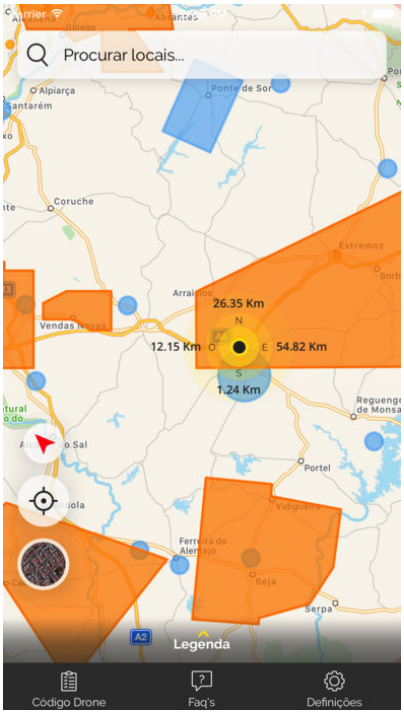


Figura 2.3: Interface da aplicação Voa na Boa com áreas proibidas à utilização de UAVs[12]

[11]Fonte: http://www.voanaboa.pt/formularios

Esta aplicação, disposta na Fig. 2.3, está disponível para IOS e Android e permite, para além de consultar o regulamento dos UAVs, observar um mapa com as áreas de voo limitado já marcadas, podendo o utilizador ainda definir automaticamente a sua localização de forma a poder saber se a zona onde se encontra é limitada ou não. Também é possível consultar as razões que levam a que uma zona seja considerada como restrita.

Apesar destas regras terem entrado em vigor em Novembro de 2016, perspetiva-se a entrada de novas regras no código dos UAVs devido ao elevado número de incidentes entre UAVs e aviões. Em Julho de 2017, a ANAC em conjunto com o governo português, formulou um novo decreto-lei para regular a atividade dos UAVs. Este documento está aprovado e encontra-se em fase de testes sem se saber quando entrará em vigor.
Neste documento será introduzida uma idade mínima para a realização de operações com UAVs. Assim, os menores de 16 anos não poderão utilizar UAVs com peso superior a 0,250 kg, a menos que sejam acompanhados por uma figura parental que cumpra os requisitos de registo e seguro obrigatórios.
Registo: os UAVs com massa superior a 0,250 kg estarão obrigados a um registo obrigatório na ANAC, juntamente com o representante legal. Estará sujeito a um registo temporário caso o UAV seja estrangeiro. Caso o proprietário se altere, a ANAC terá de ser notificada. Para o registo serão necessárias informações detalhadas sobre o proprietário e o UAV.
Após o registo, será fornecido ao proprietário, um código de identificação que deverá ser afixado na estrutura do UAV. O proprietário será obrigado a registar o UAV no prazo de 10 dias após a sua compra ou construção. Caso já tenha um, tem até 20 dias para proceder ao seu registo.
Seguro: será obrigatório os proprietários fazerem seguros de responsabilidade civil para os veículos remotos com massa superior a 0,250 kg.
A fiscalização dos UAVs pode ser feita sob duas formas: sanções e tecnologia antiUAV. Quanto à primeira, se o proprietário não cumprir as normas impostas no código dos UAVs, incorre numa contra-ordenação (leve, grave ou muito grave) que, dependendo da gravidade, pode levar a coimas com valores entre os 151 e os 4027 euros. Relativamente à segunda opção, esta monitoriza os UAVs e avisa as autoridades quando alguma das regras mencionadas anteriormente é quebrada.

2.2.2 Regulamento Europeu

Embora cada país tenha uma regulamentação diferente, a União Europeia tem reunido esforços no sentido de regulamentar o código dos UAVs dentro da Europa, de modo a unificar a legislação dos estados-membros.
Esta unificação, levada a cabo pela EASA (Agência Europeia de Segurança da Aviação)[European Aviation Safety Agency, 2012], consistirá num novo espaço europeu, chamado U-space, que irá até aos 150 metros de altitude. Este espaço será gerido por um sistema semelhante ao utilizado no tráfego aéreo e automatizado com recurso a ferramentas de identificação eletrónica e geo-facing

[12]Fonte:http://voanaboa.pt/voa-na-boa

(utiliza GPS para perceber a sua posição) de forma que os UAVs consigam aceder à informação do espaço aéreo. Espera-se que esta unificação esteja pronta e a funcionar em 2019. Uma vez completada esta unificação, é expectável que a EASA e a NATO se reunam tendo em vista a redação de novos documentos para legislar o voo dos UAVs. Estes documentos, como por exemplo o STANAG 4586 [North Atlantic Treaty Organization, 2015], representam um grande peso neste trabalho, nomeadamente nos comandos a enviar ao quadcopter (secção 5.3.7).

2.3 Controlo dos UAVs

Os UAVs podem executar diferentes tipos de movimentos, dependendo das características de que dispõem, mais propriamente do número de hélices e do seu tamanho.
A ideia deste projeto passa por utilizar apenas um género de UAV, o quadcopter, pelo que se dará apenas ênfase a este tipo de veículos.

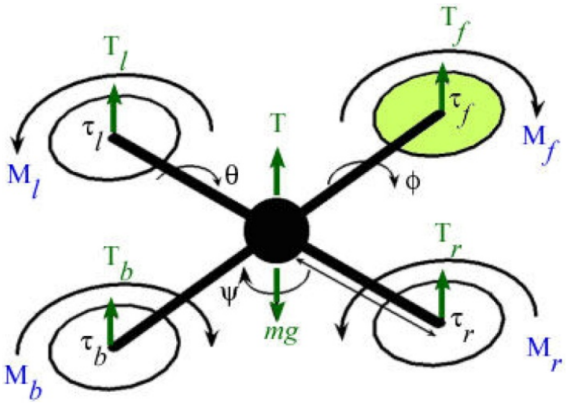
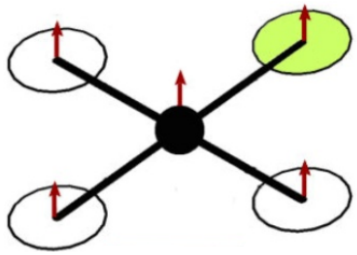


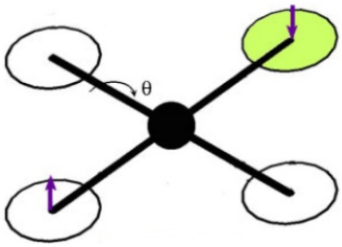
Figura 2.4: Diagrama conceptual de um quadcopter com sentido de rotação dos motores assim como as forças aplicadas [Raza and Gueaieb, 2010]

A Fig. 2.4 mostra as forças a que o quadcopter está sujeito quando se encontra a voar e como a altitude pode variar consoante o nível de rotação dos motores. Os motores apresentam um comportamento diferenciado, ou seja, o motor Mf da Fig. 2.4 (frente direito) e motor Mb (traseiro esquerdo) têm uma rotação no sentido horário enquanto que os restantes motores, Ml (frente esquerdo) e Mr (traseiro direito), possuem uma rotação no sentido anti-horário. Esta disposição serve essencialmente para contrabalançar as forças presentes no quadcopter de forma a que o mesmo consiga manter-se estável durante o voo. Estes veículos conseguem executar uma série de movimentos, gerados pelas diferentes combinações dos torques (energias geradas) dos motores. O quadcopter consegue executar várias manobras tendo em conta quatro atributos: *thrust*, *pitch*, *roll* e *yaw*. Estes atributos variam com a rotação sobre os eixos (x, y e z) e com alteração na aceleração:

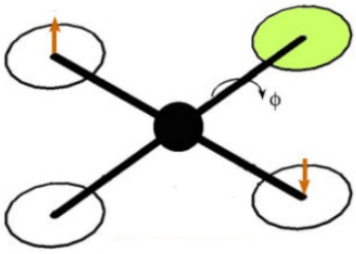
Thrust - Representado na Fig. 2.5a. Permite alterar a aceleração do quadcopter o que provoca variações na sua altitude através de uma rotação igual em todos os motores.



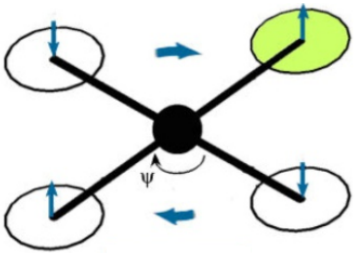
(a) Exemplo do *Thrust*[Raza and Gueaieb, 2010]



(b) Exemplo do *Pitch*[Raza and Gueaieb, 2010]



(c) Exemplo do *Roll*[Raza and Gueaieb, 2010]



(d) Exemplo do *Yaw*[Raza and Gueaieb, 2010]

Figura 2.5: Influência da variações dos eixos e da aceleração na movimentação do quadcopter[Raza and Gueaieb, 2010]

Pitch - Representado na Fig. 2.5b. Permite ao quadcopter rodar no eixo yy. Reproduzido através do aumento do impulso nos motores de trás (Mb e Mr) e redução do impulso nos motores da frente (Mf e Ml) ou vice-versa.

Roll - Representado na Fig. 2.5c. Permite ao quadcopter rodar no eixo xx. Reproduzido através do aumento do impulso nos motores Mf e Mr e redução do impulso nos motores Mb e Ml ou vice-versa.

Yaw - Representado na Fig. 2.5d. Permite ao quadcopter rodar no eixo dos zz. É gerado pela alteração do impulso dos quatro motores. Os motores Ml e Mr sofrem uma redução no seu impulso enquanto que há uma aceleração na rotação dos motores Mf e Mb (o quadcopter roda para a direita) ou vice-versa.

Outro movimento interessante e que importa destacar é o VTOL *(vertical take-off and landing)*. Este movimento é característico em veículos que fazem a descolagem e aterragem verticalmente. Um exemplo que incide bastante neste conceito, é o helicóptero. Pode ser feito com recurso a motores ou a energia elétrica.

2.4 Tipos de Comunicação

A inclusão de sistemas de comunicação nos diversos aparelhos tem-se tornado essencial para o utilizador poder comunicar com outras entidades, em diversos locais. Existem diversos tipos de comunicação sem fios; no entanto, apenas os seguintes ganham destaque:

- Infravermelhos - a troca de informação entre os dispositivos é feita por radiação infravermelha. É destinada a comunicações de curto alcance (até 1 metro) e em linha de vista, sendo utilizada por exemplo no controlo remoto de Tv, aparelhos de segurança, entre outros. Para que se proceda a este género de comunicação é necessário um transmissor de sinais, que irá enviar radiação infravermelha não visível, e um recetor que captura e interpreta os sinais transmitidos.

- Transmissão Rádio - a comunicação é feita por ondas de rádio com um determinada frequência sendo capaz de se desenrolar a grandes distâncias. Existem três elementos que permite a comunicação entre dois pontos: um transmissor (converte os sinais em ondas eletromagnéticas, enviando-as para o espaço para depois serem capturadas pelo recetor), o meio de transmissão e o recetor (descodifica o sinal recebido em sinais sonoros, digitais ou analógico). A distânica de comunicação está dependente do tipo de antena utilizada, local de emissão e receção, entre outras variáveis.

- *WIFI* - comunicação utilizado em diversos aparelhos tais como *Smartphones*, *laptops*, entre muitos outros dispositivos. Para os utilizadores poderem utilizar este tipo de rede, necessitam de se deslocar para perto de um ponto de acesso, ou seja, de um router ou outro aparelho. Este recebe e envia dados os dispositivos que se encontram ligados ao mesmo. Geralmente, este tipo de comunicação envolve a proteção da rede através uma password, caso contrário podem sofrer ataques de outros utilizadores (mesmo com password a segurança não é assegurada por completo). Existe um variado número de fatores que pode influenciar o desempenho e alcance da rede tais como o número de redes presentes no ambiente (um grande número de redes no mesmo meio pode reduzir a velocidade de comunicação), a existência de antena (aumenta a distância de comunicação), o meio onde se encontra a rede *WIFI* (a existência de muitos obstáculos reduz o alcance da rede), entre outros.

- Sistemas de Comunicação Móvel - também conhecido por GSM, é uma tecnologia utilizada pelos dispositivos móveis (telemóveis). É utilizado para serviços de voz e troca de dados. Apesar de ser considerada uma rede menos avançada, fornece novos serviços a baixo custo. O utilizador consegue enviar ou receber dados para todo o mundo, desde que o fornecedor do serviço tenha antenas ou torres de transmissão próximas do utilizador.

- Bluetooth - útil para transferir informação entre dois ou mais dispositivos próximos uns dos outros através de uma frequência de rádio de curto alcance. Destinada para comunicações a curtas distâncias (entre 4,5 a 15 metros). Antes da troca de informação entre os dispositivos é necessário que os mesmos passem por um processo de reconhecimento e emparelhamento.

De seguida é apresentada uma tabela comparativa com estas tecnlogias analisando as vantagens e desvantagens.

Tipos de Comunicação	Vantagens	Desvantagens
WIFI (Curto alcance)	Fácil de utilizar, instalar e de adicionar novos utilizadores; Presente em vários espaços (públicos e privados); Integração com vários dispositivos.	Sujeito a interferências que podem causar distúrbios na rede; A velocidade da transferência de dados reduz-se com o acréscimo de utilizadores na rede.
Rádio (Curto alcance)	Permite transferir grandes quantidades de informação; Presente em vários espaços (públicos e privados); Acarreta custos reduzidos; Presente em vários setores tais como saúde, lazer, entre outros.	Sujeito a interferências quando na mesma frequência; O poder de alcance de um sinal rádio pode ser afetado pela localização de receção do sinal, frequência do sinal, entre outros.
Infravermelhos (Curto alcance)	Exige requisitos de baixa potência; Baixo custo.	Precisa de uma linha direta entre o transmissor e o recetor; Alcance reduzido (limite máximo de 1 metro); A velocidade da informação depende do padrão utilizado e pode ir desde os 115.2 *Kbps* até 16 *Mbps*.
GSM (Grande alcance)	Não é afetado por se encontrar dentro de espaços; Fácil de integrar com outras tecnologias *wireless* (ex: LTE); Tecnologia madura e que está presente em vários países.	Pode causar interferência em vários aparelhos eletrónicos (razão pela qual se desliga o telemóvel no avião); O aumento de sinal requere *repeaters* (aumentam o sinal), ou seja, mais custos.
Bluetooth (Curto alcance)	O sinal consegue atravessar objetos físicos; Simples de usar.	Velocidade de transferência é mais baixa quando comparado com outras tecnologias; Tecnologia pouco segura e com alto gasto energético (mais tarde, surgiu o *Bluetooth Low Energy* que suavizou este fator).

Tabela 2.2: Vantagens e Desvantagens dos tipos de comunicação

Como podemos na Tabela 2.2, existem situações em que um tipo de comunicação supera outra. Por exemplo, a velocidade de transferência de dados é superior no *WIFI* do que no *Bluetooth* ou

o sinal Bluetooth é afetado pela existência objetos enquanto que os infravermelhos não (depende da natureza dos materiais). São vários as propriedades que nos ajudam a decidir quando escolher uma tecnologia em detrimento de outra.
Assim sendo e tendo em conta as propriedades do projeto, chegou-se à conclusão que a tecnologia rádio é a melhor opção visto tratar-se de uma tecnologia fiável e ao mesmo tempo acarretar custos reduzidos, algo que é bastante importante para a construção do quadcopter *low-cost*. A utilização de material *low-cost* não se trata de um requisito para este projeto mas foi explorado para se perceber se seria possível a montagem de um UAV de baixo custo.

2.5 Hardware para Plataformas UAVs

Relativamente às plataformas hardware open-source que são implementadas nos UAVs, são muitas as opções oferecidas pelo mercado. Cada uma das alternativas apresentadas de seguida contém características benéficas para determinados géneros de UAVs em ambientes específicos. Para além disso, serão explicadas algumas das funcionalidades.

Plataformas *FPGA* (*Field programmable gate array*)[Craighead et al., 2007] - plataformas em que o circuito integrado é configurado pelo cliente ou designer, depois de montado. A configuração é feita com recurso à linguagem *HDL* (*Hardware Description Language*). De entre as disponíveis, destacam-se as seguintes:

- *Phenix Pro*[RobSense, 2016]: construído num chip *Soc* (*System On Chip*) desenhado e fabricado pela RobSense Tech. O controlador de voo está equipado com um sistema em tempo real sendo este desenvolvido em Linux. Esta plataforma suporta mais de 20 interfaces, entre elas, sensores de bordo, radar *mmWave*, câmara térmica, entre outras. Permite a aceleração de hardware para visão computacional e redes neuronais convolucionais.

- OcPoC (Octagonal Pilot on Chip)[Aerotenna, 2016]: construído pela Aerotenna. Este chip estende as funcionalidades *input* e *output* para incluírem componentes *PWM* (*Pulse Width Modulation*), *PPM* (*Pulse Position Modulation*) e *GPIO* (*General Purpose Input/Output*). Estes sinais são bastante utilizados no controlo de vários tipos de sensores tais como ESC (*Electronic Speed Controller*) e motores. Permite o emparelhamento de vários sensores, processamento a bordo em tempo real e aprendizagem profunda.

Plataformas ARM (*Advanced RISC (Reduced Instruction Set Computing) Machine*)[Craighead et al., 2007] - plataformas que primam pelo desenho e acima de tudo pela arquitetura dos processadores. De entre as disponíveis, destacam-se as seguintes:

- PIXHAWK/PX4[Pixhawk, 2013]: O controlador de voo Pixhawk trata-se de uma versão melhorada do sistema de controlador de voo PX4. É composto por um controlador de gestão PX4-Flight (*FMU*) e um PX4-IO (módulo *input/output*) integrado numa placa com IO adicional, memória e outras funcionalidades. A placa PX4-Flight é uma unidade que gere todo o tipo de eventos relacionados com o voo enquanto que o PX4-IO registo os dados *output* e *input* e serve de suporte para o PX4-Flight.

- PIXHAWK 2/Cube[Pixhawk, 2016]: Versão melhorada do anterior modelo. É um pequeno cubo que possui três módulos GPS. Foi projetado para sistemas comerciais e fabricantes que têm a intenção de integrar o controlador de voo num sistema construido pelos mesmos. Todas as conexões por *input/output* para o cubo são feitas por um conector do tipo DF17 (conetores).

- Paparazzi[Paparazzi, 2003]: Provavelmente o mais antigo das plataformas hardware *open source*. Engloba sistemas *auto-pilot* e software de estação terrestre para veículos multi-copters/multi-rotors, helicópteros e aviões híbridos. Inclui várias versões do piloto automático, sensores IR (Infravermelhos), sensores inerciais, reguladores de tensão, receptores GPS, conversores, entre outras.

- CC3D and Atom[OpenPilot Manual, 2014]: Apesar de CC3D e Atom terem as mesmas funcionalidades, distinguem- se pelo seu tamanho. Trata-se de um tipo de hardware especializado na estabilização de voo. Funciona com base no *firmware* OpenPilot (secção 3.1), no entanto, pode ser configurado para voar qualquer UAV de asa fixa até um octocóptero através do software OpenPilot *Ground Control Station (GCS)* (secção 3.1).

Plataformas Atmel[Craighead et al., 2007] - Plataformas rápidas para fazer avaliação e prototipagem. Possuem kits fáceis de usar e apresentam um custo relativamente baixo[13]. De entre as disponíveis, destacam-se as seguintes:

- ArduPilot Mega (APM)[ArduPilot, 2012]: Arduino baseado num sistema *autopilot*. Trata-se de um piloto automático completo capaz de estabilização autónoma e navegação baseada em *waypoint* (rotas pré estabelecidas). É capaz de controlar vários UAVs tal como helicópteros, veículos rastreadores de antenas, entre outros.

- FlyMaple[ArduPilot FlyMaple, 2013]: Placa controladora para quadcopters, baseada no projeto Maple[14]. O design do FlyMaple é baseado num maple, que é um processador arduino ARM. É geralmente utilizada para plataformas mobile, helicópteros e em quadcopters que necessitem IMUs (*Inertial Measurement Unit*) e controladores em tempo real com alto desempenho. IMUs são dispositivos eletrónicos que medem e relatam a força específica de um corpo, a taxa angular e, por vezes, o campo magnético em torno do corpo.

Plataformas baseadas em Raspberry Pi[Craighead et al., 2007] - Plataforma alojadas em pequenos "computadores". Todo o hardware do Raspberry é integrado numa única placa. De entre as disponíveis, destacam-se as seguintes:

- Erle-Brain 3[Robotics, 2013]: Versão melhorada do Erle-Brain 2. Possui o software OpenPilot e foi desenvolvido pela Erle Robotics. Combina um computador Linux embutido

[13]Fonte:http://www.atmel.com/products/microcontrollers/avr/xplained.aspx
[14]Fonte: http://occsc.org/maple-project/

(Raspberry Pi) com uma placa filha (PXFmini). A placa PXFmini [15] é um escudo autopilot que permite combinar robos e UAVs com a família Raspberry Pi. Inclui sensores de gravidade, giroscópios, uma bússola digital, sensores de pressão, entre outros componentes.

Plataforma	Sensores	Consumo de energia (Watt)	Peso (g)
Phenix Pro	HUB, IMU, GPS, LED	2.6	64
OcPoC	IMU, Barometer, GPS, Bluetooth, WiFi	4	70
PIXHAWK/PX4	IMU, Barometer, LED	+/- 1.6	38
PIXHAWK 2	IMU, Barometer, LED	N/A	N/A
Paparazzi	IMU, Barometer	N/A	N/A
CC3D and Atom	gyro, accelerometer	N/A	8/4
ArduPilot Mega	IMU, Barometer, LED	N/A	31
FlyMaple	IMU, Barometer	N/A	15
Erle-Brain 3	IMU	N/A	15

Tabela 2.3: Hardware dos controladores de voo dos UAVs

A Tabela 2.3 permite comparar os controladores de voo em termos de hardware e ter uma melhor noção sobre os componentes disponíveis.

É possível analisar as diferentes plataformas em diferentes patamares, ou seja, em termos de sensores, consumo de energia e peso. Todos estes fatores são importantes quando contemplamos construir um determinado veículo autónomo, nomeadamente, um UAV.

Para a construção do quadcopter no presente trabalho, recorreu-se a uma placa Atmel do tipo ArduPilot Mega (APM), representada na Fig. 2.6 devido ao facto ser este hardware possuir as valências necessárias para a integração com a plataforma de simulação. Mais informação sobre o material utilizado é apresentado na secção 4.2.

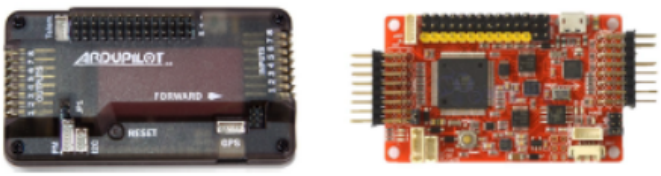


Figura 2.6: Placas ArduPilot Mega (APM) autopilot sendo a esquerda a versão 2.5.2 e a direita a versão 2.8[Craighead et al., 2007]

[15] Fonte: http://erlerobotics.com/blog/product/pxfmini/

2.6 Aplicações UAVs

Os últimos avanços tecnológicos que se têm sentido nas várias áreas mostram que os UAVs possuem um potencial enorme e que, se forem cumpridas as leis e tratados, podem muito bem desempenhar tarefas que outrora seriam inimagináveis. No passado, grande parte das aplicações dos UAVs estavam limitadas ao setor militar enquanto que hoje o caso muda de figura. Hoje é possível olhar para os UAVs e ver muito mais do que aplicações militares.
Assim sendo, podemos segmentar as aplicações dos UAVs em 3 panoramas diferentes[Austin, 2010]: militar, comercial e pessoal.

2.6.1 Aplicações militares

Destacam-se as operações navais, exército e força aérea como sendo os principais focos da utilização dos UAVs nas áreas militares [Austin, 2010].
Relativamente às operações navais, os UAVs podem voar a várias altitudes contudo, existem fatores que são necessários ter em conta tais como o vento, temperaturas, sal, turbulência, entre outras. Geralmente este género de UAVs não necessita de grande poder de fogo nem de sistemas de arrefecimento devido ao ambiente. Dentro das operações navais, os UAVs podem ajudar na deteção de navios (as baixas altitudes e reduzidas dimensões faciliatam a infiltração em frotas inimigas), confundir radares inimigos (através de sinais emitidos pelos UAVs seria possível "baralhar" o radar de um outro barco), "isco" para mísseis (UAVs equipados com materiais de fácil deteção, por parte dos inimigos, permitiria poupar recursos navais e salvar vidas apesar do custo dos UAVs), vigilância nos portos e zonas costeiras (salvaguardar os armazéns e evitar ataques surpresa) e deteção de importações ilegais (UAVs camuflados podem evitar que criminosos façam trocas ilegais. Em caso de infração, os UAVs podem enviar alertas à polícia marítima ou local).
Quanto ao exército, as missões estão sujeitas a uma miríade de terrenos e condições atmosféricas que acabam por influenciar a abordagem a proceder. Dentro das operações do exército, os UAVs podem facilitar o reconhecimento e vigilância (monitorizar o terreno, alertar as autoridades sobre potenciais inimigos e infiltrar em terrenos inimigos), estudo e planeamento do terreno de combate (estudar a geografia do ambiente assim como fatores atmosféricos pode ajudar as tropas a posicionarem-se melhor no terreno e a obterem um melhor desempenho) e deteção de minas e explosivos (o reconhecimento de minas e explosivos em terrenos pode evitar perdas humanas sem denunciar a posição aos inimigos).
Relativamente à força aérea, este trata-se do setor com menor número de soluções devido às restrições que os UAVs possuem em termos de altitude. Contudo, os UAVs podem ajudar na monitorização do espaço aéreo (constante observação por sinais de inimigos é um dos grandes propósitos dos UAVs no que toca ao controlo de tráfego).

2.6.2 Aplicações Comerciais

Destacam-se a fotografia e filmes, agricultura e pecuária, conservação do meio ambiente, companhias de eletricidade, combate aos incêndios, entrega de encomendas, agências de tráfego e combate ao crime como sendo os principais focos da utilização dos UAVs no setor comercial [Austin, 2010].

Quanto à fotografia e filmes, combinar altas altitudes (até 120m) com câmaras de alta definição pode ajudar a obter fotografias ou até gravar momentos que antes seriam difíceis de obter. Está a ser explorado por setores como jornalismo, fotografia, cinema, entre outros interessados. No entanto, apresenta limitações relativas ao limite de altitude que um UAV pode atingir.

Relativamente à agricultura e pecuária, a introdução de UAVs pode suavizar a monitorização de colheitas (a utilização de infravermelhos bem como outras técnicas pode servir para detetar doenças no seio das colheitas), proliferação de pesticidas (permite espalhar os pesticidas com maior rapidez e eficácia). A conservação do meio ambiente implica que os UAVs sejam capazes de detetar os efeitos dos gases poluentes na camada de ozono.

Quanto ao combate aos incêndios, os UAVs podem desempenhar vários papéis. A monitorização das matas (focar as zonas que necessitam de limpeza tal como enviar alertas quando há uma situação de incêndio) e a restauração de comunicações (UAVs serviriam como um ponto de restauro ou extensor de alcance para as comunicações quando existissem falhas) são alguns exemplos de aplicação.

A entrega de encomendas por parte dos UAVs é outro aspeto a ter em conta. Aliás, a Amazon [16] utiliza o Prime air (Serviço de entregas por UAV) para entregar as encomendas dos seus clientes. Podia ser aplicado pelos correios. Um exemplo que retrata este cenário foi aquele que se sucedeu na Australia em que UAVs entregaram burritos à população rural[17].

Relativamente ao combate ao crime, a deteção e prevenção do crime pode ajudar a Polícia a evitar conflitos, bem como a procurar fugitivos.

É expectável que, até 2021, o setor das infraestruturas e da agricultura seja o mercado com maior aplicação comercial de UAVs. Todavia, outras indústrias tais como os transportes, segurança e telecomunicações, estão atentas aos benefícios que os UAVs podem trazer para os seus negócios.

2.6.3 Pessoal

A comercialização de UAVs para o público em geral tem sido um dos grande pontos de viragem na venda de tecnologia dos últimos anos [Austin, 2010]. Pessoas de todas as idades têm aderido aos UAVs de todos os tamanhos e feitios para cumprir tarefas como pilotar um avião telecomandado, tirar fotografias ou gravar momentos, entre outras. As faixas etárias mais novas têm recorrido a esta tecnologia maioritariamente para divertimento e lazer enquando que as faixas etárias mais velhas vêm esta tecnologia como uma área de investigação, com potencial para se expandir para outras áreas.

[16] Fonte: https://www.amazon.com/Amazon-Prime-Air/b?node=8037720011

[17] Fonte: https://www.engadget.com/2017/10/17/alphabet-project-wing-australia-tests/?guccounter=1

Capítulo 3

Revisão da literatura

O estado de arte deste trabalho será analisado em várias perspetivas. Primeiro será feito um conjunto de observações sobre os pacotes de software para controlo de voo de UAVs que estão disponíveis para utilização. Depois de explorado cada um dos softwares relativos ao controlador de voo, será apresentado um quadro onde será possível comparará-los a vários níveis (processador e linguagem), de forma a facilitar a escolha dependendo do cenário de utilização. Uma vez finalizada a análise ao software existente, é estabelecida uma comparação entre plataformas de simulação prontas a usar, onde iremos procurar explicar algumas das suas componentes e quais as que se aproximam da nossa plataforma de simulação. Por fim, serão analisados outros possíveis projetos de plataformas de voo em que serão abordadas as metodologias e técnicas utilizadas. Estudos que procuram combinar UAVs reais com UAVs ou outros veículos autónomos simulados também serão tidos em conta.

3.1 Software existente

Quanto às plataformas de software, são bastantes as opções na área dos veículos autónomos. Estas plataformas, criadas por várias entidades, são controladores de voo que permitem aos UAVs voar. Para este projeto foram focadas apenas aqueles que se mostram como open-source e gratuitas, dado que se pretende apenas explorar soluções de baixo custo ou open-source à semelhança do projeto em curso. Cada uma das alternativas apresentadas de seguida contém características ideais para determinados géneros de UAVs em diferentes ambientes. Algumas estão disponíveis e atualizadas enquanto que outras estão indisponíveis por terem caído em desuso ou por terem sido ultrapassadas por novas tecnologias.

- ArduPilot[1] -> Trata-se de um software bastante utilizado no paradigma dos UAVs mostrando-se bastante fiável e capaz de cumprir as tarefas que lhe são destinadas tais como registo,

[1]Fonte: http://ardupilot.org/

análise e simulação de dados. Funciona em aeronaves convencionais, multirotors, helicópteros, diferentes tipos de barcos e submarinos, sendo utilizado não só por amadores como por profissionais. Utiliza a linguagem C++ e Python e suporta um processador 8-bit ARM mas com a release do ArduPilot Mega (APM) evoluiu para um processador 32-bit ARM. Caracteriza-se essencialmente pela sua versatilidade e facilidade com que é incorporado nos diferentes veículos;

- Multiwii[2] -> Identifica-se por ser um software para controlador de voo desenvolvido para plataformas Arduino e baseados em vários sensores (incluiu complexos). Utiliza linguagem C e foi desenvolvido para suportar sensores como o giroscópio e acelerómetro da consola Nintendo Wii no entanto, com os últimos desenvolvimentos, é capaz de lidar com outros sensores. A estabilidade alcançada é excelente para FPV (Visão de primeira pessoa) e permite executar qualquer tipo de acrobacia. Utiliza um processador 8-bit ATmega328. Desenvolvido com a linguagem C;

- AutoQuad[3] -> é um projeto de desenvolvimento de ESC baseado em hardware e controladores de voo gratuitos. Destina-se a fornecer um controlador de voo com estabilização, voo dinâmico e recursos do piloto automático. Utiliza um processador 32-bit ARM. Foi criado com base na linguagem C;

- LibrePilot[4] -> Projeto focado no estudo e desenvolvimento de software, para ser utilizado em vários níveis, incluindo controlo e estabilização de UAVs e de veículos robóticos. Ideal para controlar multicopters e outros modelos RC. Possui imensas funcionalidades que permitem configurar este software para vários cenários. Utiliza um processador 32-bit ARM. Foi desenvolvido com recurso à linguagem C++;

- Dronecode[5] -> Programa patrocinado pela Fundação Linux[6] onde o objetivo é desenvolver uma plataforma open-source para desenvolvimento de UAV de uma forma barata e eficaz. Esta plataforma é composta por: PX4 Flight Stack (sistema para controle de voo); MAVLink (kit de ferramentas para comunicação robótica); QGroundControl (Interfaces mobile e desktop para configuração do sistema de voo). Está equipado com um motor de simulação capaz de introduzir novos métodos de simulação. Foi desenvolvido com as linguagens C e Python;

- SMACCMPilot[7] -> Controlador de voo desenvolvido pela Galois[8], empresa que trabalha com sistemas críticos, destinado a pequenos quadcopters. Foi desenvolvido com recurso a linguagem Ivory (C mais seguro) para fornecer sistemas de programação seguros. Possui

[2]Fonte: http://www.multiwii.com/
[3]Fonte: http://autoquad.org/
[4]Fonte:https://www.librepilot.org/site/index.html
[5]Fonte:http://px4.io/
[6]Fonte:https://www.linuxfoundation.org/
[7]Fonte: https://smaccmpilot.org/
[8]Fonte:https://galois.com/

uma interface externa em C que permite reutilizar componentes do projeto open source ArduPilot.

Podemos comparar e observar quais as principais propriedades dos software de controlo disponíveis no mercado, na seguinte tabela.

Software	Características	Plataformas	Veículos	Linguagem
ArduPilot	Especializado em ferramentas avançadas de registo, análise e simulação de dados	Windows, Mac, Linux, IOS e Android	Multirotors, helicópteros e submarinos	C++ e Pyhton
LibrePilot	Especializado em controlo e estabilização de veículos	Windows, Linux, Mac e Android	Multirotors e helicópteros	C++
MultiWii	Especializado para UAVs que requerem sensores com alguma complexidade	Windows, Mac, Linux, IOS e Android	Quadcopters, tricopters e hexacopters	C
DroneCode	Possui um motor de simulação flexível que permite adicionar novos métodos de simulação	Windows, Mac, Linux, IOS e Android	Capaz de ser integrado em vários géneros de veículos (incluindo terrestre)	C e Python
AutoQuad	Especializado em estabilização, voos dinâmicos e funcionalidades auto-piloto	Windows, Mac, Linux e Android	Multicopters, veículos de asa fixa ou de um único rotor	C
Project YMFC-ALL	Simples e com excelente documentação. Inclui métodos de calibragem.	Window, Mac, Linux, IOS e Android	quadcopters	C

Tabela 3.1: Software disponível para controlo de UAVs

A Tabela 3.1 permite-nos constatar quais os softwares acessíveis, versáteis e documentados. Cada um possui características que os tornam ideais dependendo da aplicação e do cenário de atuação. Podem ser adaptados consoante as necessidades dos utilizadores. O software utilizado para o controlo do voo do quadcopter foi o Project YMFC-ALL.

3.2 Plataformas de Simulação

Esta secção é dedicada à apresentação de plataformas de simulação prontas a utilizar. O facto de este projeto estar inserido num trabalho de maior dimensão fez com que se continuasse com a mesma plataforma de simulação, ou seja, o FSX. Todavia, o surgimento de novas plataformas de simulação, novas funcionalidades e a necessidade de se perceber se existe alguma alternativa melhor ao FSX levou a uma nova investigação sobre as plataformas de simulação disponiveis. Características como funcionalidades e capacidades, cenários de utilização, vantagens, aplicações *third parties*, APIs, preços, entre outras informações, são exploradas. Esta análise é feita em várias vertentes que permitem discernir as vantagens e desvantagens de cada uma.

Flight Simulator X[9] : Considerado o melhor simulador de voo disponível no mercado, apesar de também ser considerado como um jogo para computadores. É bastante utilizado pelo facto de possuir uma API que permite adaptar a qualquer projeto o que acaba por facilitar o desenvolvimento de novas ferramentas. Também permite modelar falhas nos UAVs. Tem várias características tais como grande quantidade de cenários, equipado com um modo online onde podemos desempenhar várias tarefas (controlador de tráfego, piloto ou copiloto), modo corrida que suporta multiplayer, várias dificuldades no plano de corridas, possui um sistema de meteorologia completo e complexo, utiliza uma abordagem paramétrica para o modelo de voo através de vários parâmetros, permite testar falhas (automaticamente ou manualmente) e é compatível com windows. Inclui vários serviços aeroportuários, tráfego de veículos nas auto-estradas, tráfego marítimo (barcos e navios). Outro fator que destaca o FSX é a grande quantidade de add-ons que este software possui. Atualmente existe uma versão melhorada e que foi construida tendo conta os princípios do FSX, Prepar3D.
Acarreta um custo de 29,36 dólares sendo que este valor pode subir com a aquisição de expansões. A Fig. 3.1 representa um possível cenário de utilização.

Figura 3.1: Cenário de utilização da Plataforma FSX[9]

A plataforma de simulação em utilização no presente e em trabalhos passados é o FSX.

[9]Fonte: https://arielcreation.blogspot.com/2016/01/indonesia-traffic-x-fsx.html

X-Plane[10]: Provavelmente um dos melhores e mais poderosos simuladores no mercado. Trata-se de uma excelente ferramenta que pode ser utilizada para a previsão das capacidades de voo de UAVs de asa fixa ou rotativa com grande precisão. Ideal para os engenheiros que procuram prever como um drone voará e para os entusiastas que desejam explorar e treinar voos dinâmicos. Encontra-se bastante documentada, com bastantes instruções sobre como configurar os voos e cenários. Possui várias funcionalidades tais como customização do cenários (tempo, terreno e aeronave), simular falhas (manualmente e automaticamente), possui uma estação instrutora que simula o comportamento de um instrutor e que nos ajuda a progredir e é compativel com Windows, Mac e Linux. Utiliza uma abordagem geométrica para determinar a dinâmica de vôo. Apresenta duas formas de interação programática: *sockets* e add-ons UDP (*User Datagram Protocol*). Comparando as duas, a opção de add-ons UDP revela-se melhor devido ao nível de documentação. Enquanto o recurso a sockets providencie pouca documentação, permitindo tanto a recuperação quanto a publicação de dados, o segundo é bem documentado, apresentando uma gama muito mais ampla de possibilidades de interação. Estes add-ons são programas escritos em C.
Apresenta vários custos dependendo da versão de utilização. Apesar de estar na versão 11, a versão mais barata é a versão 9 que tem um custo de 39,99 doláres. A Fig. 3.2 representa um possível cenário de utilização.

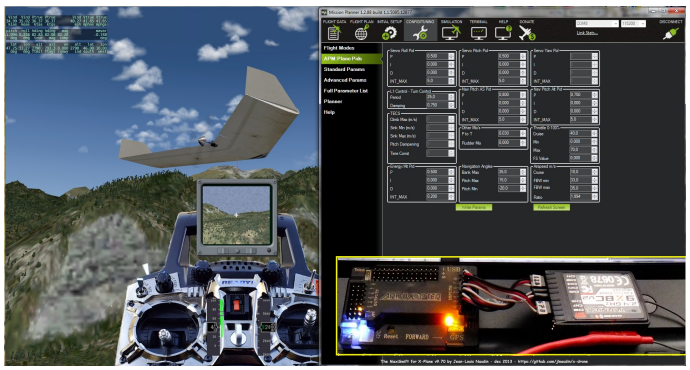


Figura 3.2: Cenário de utilização da Plataforma X Plane [11]

DroneSim Pro[12]: Simulador profissional, inovador e de fácil utilização que analisa a física envolvente no voo de um drone tendo sido desenvolvido com o objetivo de treinamento de voo de UAVs. Permite simular um voo com diferentes ambientes e condições meteorológicas. Possui várias características tais como modelos de voo precisos com Phantom 2 Vision +[13], visualização por drone ou piloto, os cenários são bastante realistas, avisa perante restrições de altitude e quando

[10]Fonte: http://www.x-plane.com/drones/
[11]Fonte:http://diydrones.com/group/apmusergroup/forum/topics/accuracy-of-xplane-hil-simulation
[12]Fonte: https://www.dronesimpro.com/
[13]Fonte: http://www.dji.com/phantom-2-vision

há perdas relativamente ao VLOS (*Visual Line of Sight* - se conseguimos ver o drone à distância), possui um motor de gráficos 3D com boa definição e é compatível com Windows e Mac. Tem um custo de cerca de 29,99 dólares para a utilização deste. A Fig. 3.3 representa um possível cenário de utilização.

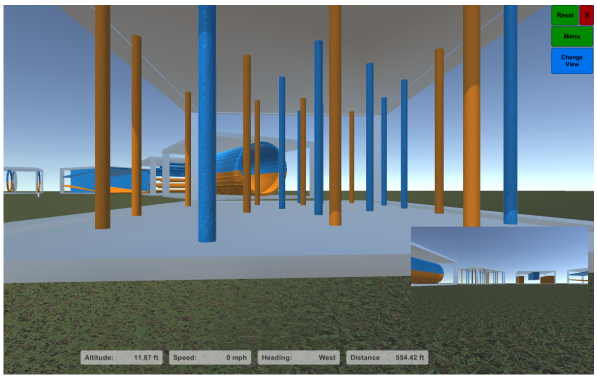


Figura 3.3: Cenário de utilização da Plataforma DroneSim Pro [12]

Real Flight[14]: Talvez um dos simuladores mais completos e abrangentes que existe no mercado. Está equipado com um comando de controlo, chamado *Interlink Elite*, que facilita a configuração da aeronave e dos cenários que pretendemos usar. As últimas versões vêm equipadas com suporte VR (Realidade Virtual). Está equipado com várias funcionalidades tais como uma interface simples e intuitiva, desafios com relativa complexidade, o que torna a experiência do utilizador bastante apreciável e divertida, simular o voo em diferentes cenários (mais de 40) com diferentes horários (dia e noite), pacotes de expansão que permitem adquirir alguns extras (mais configurações, cenários e aeronaves) e é compatível com windows.
Este equipamento traz um custo de 179,99 dólares (inclui controlador) sendo que este custo pode subir com a aquisição de expansões. A Fig. 3.4 representa um possível cenário de utilização.

Figura 3.4: Cenário de utilização da Plataforma Real Flight [14]

AeroFly RC7[15]: Simulador bastante realista, ideal para quem quiser praticar a visão em primeira

[14]Fonte: https://www.realflight.com
[15]Fonte: http://www.aeroflyrc.com/index.html

pessoa (FPV) ou praticar corridas de drones. Um dos pontos mais relevante que leva muitos utilizadores a escolherem este simulador em comparação com outros é o facto de facilitar a integração do controlador de voo do drone na plataforma. Possui várias características tais como bastantes modelos de aeronaves e cenários realistas (200 e 50 respetivamente), os modelos das aeronaves são escaláveis, suporte para dispositivos de input (*Joysticks*, *gamepads*, entre outros), vista pelo cockpit da camera do drone, efeitos de brilho com bastante detalhe, modo multiplayer com voice chat, possui um centro de gravação e de replay de cenas e é compatível com windows e mac.
Este equipamento tem um custo de 198,70 dólares. A aquisição de expansões, traz um acréscimo no valor indicado anteriormente. A Fig. 3.5 representa um possível cenário de utilização.

Figura 3.5: Cenário de utilização da Plataforma AeroFly [15]

LIFTOFF[16]: Simulador relativamente recente e imersivo (2016) e que está direcionado para os amantes de corridas de drone e para os interessados em visão de primeira pessoa (FPV). É talvez dos simuladores mais simples dentro das opções disponíveis. Possui como características a especialização em corridas, freestyle, entre outros modos, aeronaves com bastante detalhe, modificar quadcopter, construir e personalizar as pistas, suporta multiplayer e é compatível com windows e mac.
Naturalmente o facto de ser recente faz com que haja espaço para melhorias. Pelo feedback que este simulador tem recebido, áreas como o sistema de notificações (por exemplo, quando atinge a altura máxima ou excede o limite da zona de voo) e desafios são aspetos a melhorar num futuro próximo. Apesar de este simulador ainda estar em acesso antecipado, apresenta um custo de 19,99 dólares. A Fig. 3.6 representa um possível cenário de utilização.

[16]Fonte: https://www.immersionrc.com/

Figura 3.6: Cenário de utilização da Plataforma LIFTOFF [16]

Heli X[17]: Simulador de voo profissional especialmente dedicado aos helicópteros mas que também suporta multicopters (tricópteros, quadcopters, hexacopters, entre outros). Para utilizar este simulador é necessário um controle RC (pode ser adquirido em conjunto com o software). Equipado com várias funcionalidades tais como muitos cenários e vários tipos de treino, vários tipos de aeronaves, multiplayer e a sua configuração, sistema bastante flexível, configurar os cenários (vento, luminosidade, entre outros), atualizações e demo gratuitas e é compatível com windows, mac e linux.
Este equipamento tem um custo de 55 dólares. A Fig. 3.7 representa um possível cenário de utilização.

Figura 3.7: Cenário de utilização da Plataforma Heli X [17]

Podemos comparar e observar quais as principais propriedades das plataformas de simulação disponíveis no mercado, na seguinte tabela.

[17]Fonte: http://www.heli-x.info/cms/

Plataformas de Simulação	Motor	Gráficos	Documentação	Extras	Plataformas
FSX	modelo paramétrico para simulação	18 aviões, 28 cidades e +24,000 aeroportos. Utiliza DirectX	Boa	Sistema de Missões	Windows
X-Plane	modelo geométrico para simulação	Bastantes aeronaves, cenários e aeroportos. Dinâmico com mecânicas de voo fluídas	Boa	Plugin para a criação e partilha de aeroportos com outros utilizadores	Windows, Mac e Linux
DroneSim Pro	N/A	Cenários realistas (interiores, exteriores e ambiente de catastrofe)	Média	Vista de drone ou piloto	Windows e Mac
Real Flight	N/A	Cenários realistas (brilho, humidade, etc). Motor com UNIGINE [18]	Boa	Sistema de nível (melho-ramento de skills)	Windows
AeroFly	N/A	+200 aeronaves e 50 cenários	Boa	Suporta VR e replay de cenas	Windows e Mac
LIFTOFF	N/A	Bastantes cenários e aeronaves	Média	Modo Freestyle	Windows e Mac
Heli X	N/A	Bastantes cenários e aeronaves. Usa o coeficiente de Reynolds (usado em mecânica de fluidos)	Boa	Modo online com multiplayer	Windows, Mac e Linux

Tabela 3.2: Plataformas de Simulação disponíveis

Foram estudadas as plataformas tendo em conta o Motor, Gráficos, Documentação (medido pela dificuldade de obtenção de informação da plataforma em causa), Extras e Plataformas. Pela Tabela 3.2, observamos que as plataformas de simulação DroneSim Pro e LIFTOFF são plataformas onde se encontra informação com alguma dificuldade a respeito de alguns aspectos importantes como o motor de simulação, o que acaba por ser um grande inconveniente para quem procura alternativas no mercado. Todas estas plataformas podem ser adaptados consoante as necessidades dos utilizadores e apresentam vantagens umas em relação às outras.
De todas as plataformas referenciadas, aquela se destaca e que é utilizada neste projeto é o FSX, pelo facto de poder simular missões de vários tipos, ter um grande poder de customização do cenário de estudo (terreno, tempo, trânsito e efeitos visuais como fogo, explosões, fumo, etc), possuir um excelente documentação e de estar em constante evolução com novas funcionalidades. Plataformas como Aerofly o Heli X são opções também bastante interessantes pois permitem simular o voo de UAVs em cenários de desastre bastante realistas e podem funcionar com tecnologias emergentes como a realidade Virtual (VR). Contudo, o facto de não estarem preparadas para a simulação de missões de várias naturezas, faz com que estas opções caiam perante o FSX.

3.3 Abordagens Semelhantes

São apresentados vários estudos que seguiram abordagens semelhantes ao que se pretende implementar neste trabalho, ou seja, contruir e implementar um quadcopter numa plataforma de simulação para missões multi-veículo.

Em primeiro lugar, foi analisado o artigo Uma plataforma de simulação simbiótica para os Quadcopters baseados em agentes[Veloso et al., 2014b]. Este projeto procura fazer um desenvolvimento contínuo de uma arquitetura capaz de suportar a simulação simbiótica (através de módulos Xbee[19]) de quadcopters com o objetivo de avaliar técnicas desde o controle de baixo nível até as estratégias de coordenação e cooperação; em outras palavras, uma plataforma para o projeto, simulação e desenvolvimento de sistemas multiagentes Quadcopter.
O estudo da relação simbiótica feita ao sistema físico (quadcopter) é feita tendo em conta cinco parametros: sistemas simbióticos de apoio à decisão de simulação (SSDSS), sistemas simbióticos de controle de simulação (SSCS), sistemas simbióticos de previsão de simulação (SSFS), simbióticos sistemas de validação de modelos de simulação (SSMVS) e simbióticos sistemas de detecção de anomalias de simulação (SSADS). A unificação destas componentes numa arquitetura sólida permite a instanciação de múltiplos (quadcopters virtuais e / ou reais) enquanto que numa perspetiva fisica procura que os UAVs reais possam receber o estado dos sensores instalados no mesmo e receber comandos de alto nível. Para a utilização de UAVs no simulado foi utilizado um sistema SMA, ou seja um sistema Multi Agente.

[18]Fonte:https://unigine.com/
[19]Fonte: https://www.digi.com/xbee

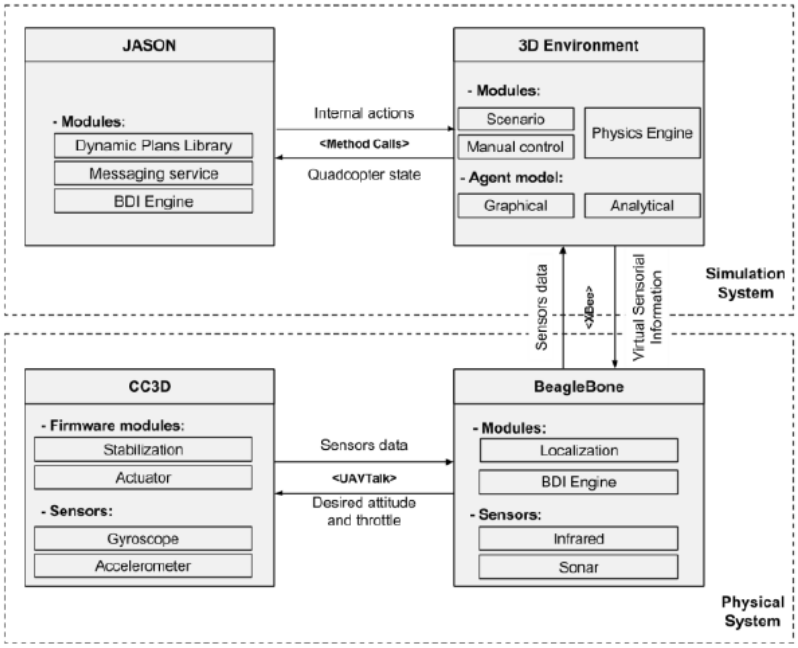


Figura 3.8: Arquitetura Proposta [Veloso et al., 2014b]

A arquitetura proposta na Fig. 3.8 está dividida em dois pilares: Sistema de Simulação e Sistema Físico.

Quanto ao sistema de simulação, numa primeira fase é feita a simulação da tomada de decisão do agente, enquanto que numa segunda fase procura-se o controle de baixo nível e a configuração do ambiente. Como um dos objetivos deste trabalho é aliviar a carga do projeto ao implementar os planos para os UAVs utilizou-se o *AgentSpeak* para desenvolver os comportamentos de alto nível. Um agente criado com a linguagem AgentSpeak, é definido como um conjunto de crenças, objetivos e sua biblioteca de planos, que adota o modelo de Crença, Desejos e Intenções (BDI). É utilizado um intepretador (Jason) na linguagem AgentSpeak que permite a adaptação na base do plano no tempo de execução para que os agentes tenham uma reação mais rápida e mais flexibilidade através de múltiplas simulações de determinados cenários. Estes agentes atuam sobre um ambiente 3D. Este ambiente usa jME32, um kit de desenvolvimento de software baseado em Java, que está dividido em três módulos: o cenário, o mecanismo físico e o controle manual. O primeiro suporta uma representação 3D do ambiente simulado com o objetivo de fornecer percepções para agentes virtuais e reais; um mecanismo físico confiável, essencial para testar o controle e a estabilidade de baixo nível e um modo para controle de teclado ou até mesmo um controle de rádio exclusivamente para testes e intervenções humanas.

Relativamente ao sistema físico, este possui duas camadas para descrever o agente-quadcopter; Ao fazer isso, é feita a distinção das atividades relacionadas às tarefas de estabilização e controle do quadcopter (camada fisica) das tarefas relacionadas aos processos de tomada de decisão e deliberação (camada estratégica que trata da ação, planejamento e assim por diante). Para implementar a camada física, optou-se por usar a plataforma de piloto automático OpenPilot CC3D4

para embarcações e helicópteros com múltiplos rotores; enquanto, para a implementação da camada estratégica, utilizou-se um ambiente baseado em Linux em uma plataforma BeagleBone. A plataforma OpenPilot CC3D trata-se de uma placa de circuito de baixo custo equipada com um conjunto de sensores que fornece feedback aos controles internos do motor; esses sensores englobam uma unidade de medição inercial (IMU), que mede acelerações e orientação em três direções diferentes. O BeagleBone trata dos principais algoritmos de tomada de decisões e os processos mais exigentes em termos de computação, pois o OpenPilot CC3D possui capacidades limitadas de processamento e memória. A interface interna do Sistema Físico conecta o OpenPilot CC3D ao BeagleBone através de um mecanismo no qual ambos compartilham um conjunto de objetos; quando há uma alteração nesses objetos, o sistema atualiza os campos de objeto para todas as plataformas envolvidas.
Para avaliar o desempenho do sistema, testou-se o sistema em dois cenários que procuram replicar o estado do quadcopter real através da leitura dos sensores giroscópio e infravermelhos.

TABLE I. INFRARED SENSOR - RESULTS

Distance - Real	43 cm	49 cm	73 cm	110 cm
Infrared Sensors (Real)				
Voltage - Average (V)	1.40	1.27	0.86	0.61
Distance - Average (cm)	43.43	48.21	73.77	106.50
Distance – Variance (cm)	0.24	0.22	1.12	3.59
Infrared Sensors (Virtual)				
Voltage - Average (V)	1.40	1.24	0.74	0.49
Distance - Average (cm)	43.40	49.59	73.49	110.13
Distance – Variance (cm)	0.006	0.006	0.004	0.003

Figura 3.9: Resultados do sensor infravermelhos [Veloso et al., 2014b]

Como é possível observar na Fig 3.9, a precisão do sensor virtual é semelhante ao do real, no entanto, sua precisão (representada pela variação) é melhor. Isso significa que é preciso injetar algum ruído gaussiano no modelo infravermelho para mapear corretamente o funcionamento do real.
O artigo Uma Plataforma para o Projeto, Simulação e Desenvolvimento de Sistemas Multi-Agente Quadcopter [Veloso et al., 2014a] (elaborado em conjunto com o anterior) fornece mais informações sobre a plataforma de simulação que possibilita uma nova perspectiva de implementação de UAVs para aplicações reais onde o projetista pode desenvolver os quadcopters e testá-los em ambientes virtuais sem prototipar o veículo aéreo real.

Em segundo lugar, estudou-se o artigo Simulação Online para Planeamento de Caminho Adaptativo de UAVs[Kamrani and Ayani, 2007]. Neste artigo, a simulação online no planeamento de caminhos de UAV é analisado e seu desempenho é comparado com dois métodos: um planeamento de caminho assistido por simulação offline e um método de busca exaustivo. O planeamento de

caminhos é um termo bem conhecido e discutido em muitos domínios e refere-se a problemas bastante diferentes em diferentes comunidades científicas. Planeamento do caminho significa determinar o caminho do UAV para que ele possa rastrear efetivamente um alvo quando alguma informação sobre o ambiente e o alvo está disponível.

Foram consideradas duas abordagens: pesquisa exaustiva e pesquisa auxiliada por simulação offline. Busca exaustiva é uma busca que procura todas as estradas para encontrar o alvo enquanto a simulação offline é um método de Monte Carlo que estima a probabilidade de existência do alvo quando o UAV atinge a área de responsabilidade e prioriza a busca em áreas onde essa probabilidade é maior. O método de pesquisa exaustiva é simples mas tem um desempenho mau. Já o método de simulação offline mostra um desempenho consideravelmente superior. Contudo, o sistema é pouco adaptável e não pode mudar o caminho escolhido quando novas informações ou observações do sensor estiverem disponíveis. Para superar esse problema, utilizou-se a simulação simbiótica (ou simulação online). O sistema simbiótico de simulação é definido como um sistema que executa um conjunto de simulações "*what-if*" em tempo real para controlar ou otimizar o desempenho do sistema.

A implementação deste método em UAVs traz alguns problemas que merecem destaque. Numa missão de vigilância, não há um quadro completo do estado do sistema. Neste caso, o objetivo passa por otimizar o processo de aquisição de informações. Os dados do sensor, antes de o alvo ser detectado, consistem principalmente em informação "negativa", isto é, falta de medição do sensor onde era esperada. Essas informações devem ser utilizadas para modificar a estimativa da localização da segmentação. O processo de tirar conclusões dos dados dos sensores é um problema estudado na comunidade de fusão de informações. Uma técnica bastante usada na fusão de informações é a Filtragem de Partículas (PF). Esta trata-se de um método sequencial de Monte Carlo baseado na representação de uma partícula de densidades probabilísticas. Em aplicações de rastreamento, o PF é um processo de simulação online, que corre em paralelo com o processo de coleta de dados e pode ser considerado como um exemplo de um sistema de simulação simbiótica. No planeamento do caminho online do UAV, usa-se o PF para estimar o estado do alvo. Esta estimativa que analisa um conjunto de partículas, é usada em simulações "*what-if*" para determinar como o UAV se deve mover para coletar novos dados da forma mais eficaz possível.

Para avaliar o desempenho da abordagem de simulação simbiótica, observaram-se vários cenários de teste com a ajuda da ferramente simulador S2. A Figura 3.10 ilustra o tempo de detecção do alvo como uma função da distância do UAV à origem, para os três métodos de busca e para três caminhos diferentes.

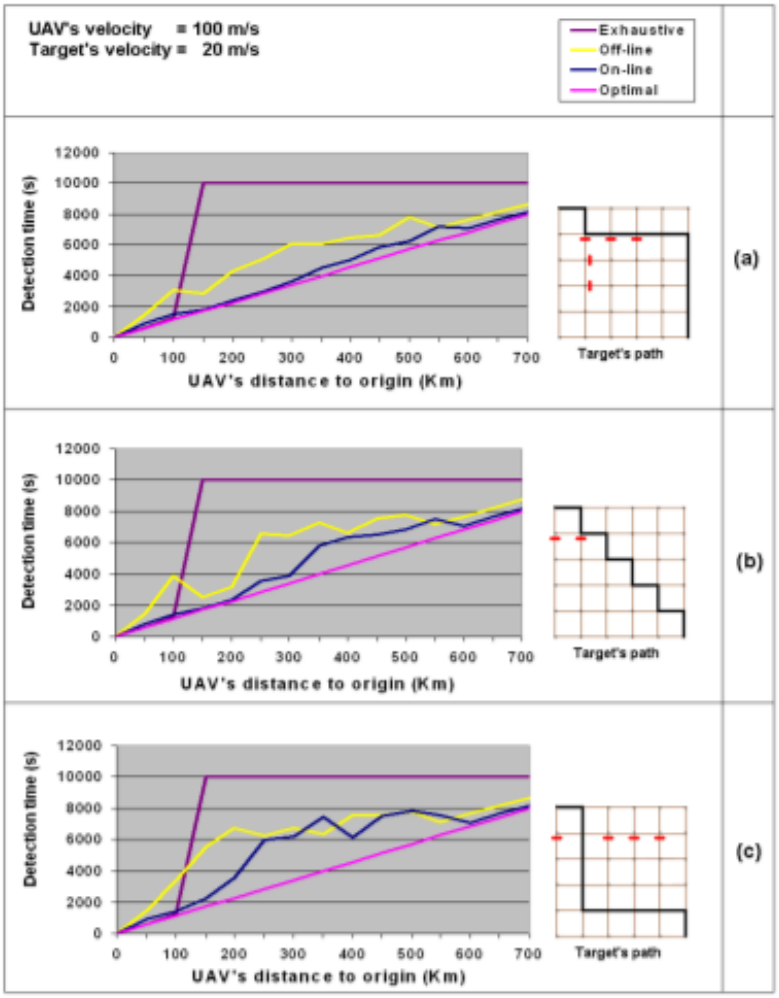


Figura 3.10: Tempo de detecção em função da distância do UAV à origem em simulação offline e online [Kamrani and Ayani, 2007]

Como esperado, independentemente do método de busca aplicado, o tempo de detecção aumenta com a distância do UAV. Como é possível observar na Fig. 3.10 o método de busca de simulação online em um ambiente dinâmico na maioria dos casos mostra um desempenho mais superior à simulação offline e o método de busca exaustivo. Somente quando a distância de origem do UAV é menor que 100Km, a busca exaustiva tem um tempo de detecção mais curto do que o método de simulação offline, o que pode ser explicado pelo fato de o método de simulação offline ter um atraso para simular toda a missão antes do início. Para distâncias maiores, o método de simulação offline é superior ao método de pesquisa exaustivo.

Em terceiro lugar, foi estudada a abordagem Plataforma de controle para Múltiplos Veículos Aéreos Não Tripulados (VANTs)[Leite, 2015]. Esta dissertação visa estender as funcionalidades de uma plataforma de controle de UAVs denominada por *DroidPlanner*[20], para o controle de múltiplos UAVs em simultâneo.

[20]Fonte:http://diydrones.com/group/tower-droidplanner

O *DroidPlanner*, estação de controle também conhecida por Ground Control Station (GCS), funciona com base num sistema de troca de mensagens entre um dispositivo móvel (preferencialmente Android) e um UAV, recorrendo ao protocolo MAVLINK. As mensagens enviadas pelo *DroidPlanner* para o UAV contêm comandos a executar, que depois de interpretadas por este, envia informação sobre o seu estado atual ao *DroidPlanner*. Uma vez recebidos esses comandos, o UAV executa-os e envia feedback sobre as suas ações para o dispositivo móvel com o fim de que este possa atualizar através da interface gráfica. Este software também permite criar missões, definindo pontos de interesse no mapa a serem visitados pelo veículo e a sua comunicação pode ser feita com base no protocolo UDP através da conexão WIFI, TCP com tecnologia 3G ou Bluetooth ou até mesmo por USB através do rádio.
No entanto, o *DroidPlanner* apenas funciona com um UAV. Para a inclusão de vários UAVs, mudanças foram feitas a vários níveis tais como comunicação, estrutura e interface.
Quanto à comunicação, este preocupa-se com a interação entre plataforma de controle e UAVs. Neste projeto, o protocolo de comunicação seguido foi o MAVLINK, por se tratar de um protocolo leve e por permitir a troca de informação entre plataforma de controle e UAV (MAVLINK e STANAG 4586 são dois conceitos diferentes, enquanto que MAVLink é um protocolo de comunicação usado por vários UAVs, STANAG 4485 é um padrão completo que foi criado pela NATO para ter interoperabilidade entre vários veículos autónomos). Relativamente à estrutura, esta é composta por componentes internos do sistema responsáveis pela estabilização do mesmo. Neste caso foi estendida para armazenar a informação dos diversos veículos. Na interface, esta foi adaptada para mostrar o desempenho do sistema assim como algumas das suas propriedades importantes para o normal desempenho dos UAVs. Suporta várias visualizações.
Foi também desenvolvida uma plataforma de simulação como forma de observar o desempenho dos UAVs e testar vários cenários com vários UAVs. O modelo de comunicação utilizado por este foi o Software em Loop (SIL) visto ser um protocolo semelhante ao utilizado pelos UAVs reais.
Este simulador permite que diferentes tipos de UAVs possam assumir um comportamento simulado de forma eficiente sem necessitar de hardware, visto que o SIL usa o software ArduPilot que permite simular como piloto automático dos veículos. Funciona em ambientes Windows e Linux.
Relativamente à componente da interação entre veículos autónomos reais e simulados, este projeto apenas funciona com UAVs simulados. Contudo, a ideia desta dissertação passa por aplicar este processo a UAVs reais. Assim sendo, foram contemplados possíveis veículos, sendo os escolhidos os multirotores/multicopteros.
Estes veículos são dinamicamente instáveis (devido à disposição dos seus rotores) e por isso necessitam de um controlador de voo integrado. Para além disso, o facto de se constituírem como veículos autónomos cria uma necessidade de se combinar informações provenientes do GPS, giroscópio, acelerómetro, entre outros.

Com base nestes requisitos, foi escolhido o dispositivo Multirotor IRIS+, presente na Fig. 3.11 com as características descritas na Tabela 3.3:

Figura 3.11: Dispositivo multirotor IRIS+ utilizado[21]

Características	
Peso(g)	1282
Capacidade de Carga(g)	400
GPS	3DR uBlox GPS com bússola
AutoPilot	Pixhawk v2.4.5
Firmware	ArduCopter 3.2
Bateria	polímero de lítio 3S 5.1 Ah 8C

Tabela 3.3: Características do dispositivo multirotor IRIS+

Utilizou-se os módulos XBee e ZigBee como link de comunicação entre os UAVs e a estação de controle.

A escolha destes componentes foi feita tendo em conta os requisitos impostos, ou seja, para controlar múltiplos VANTs em simultâneo, o canal de comunicação deveria ser capaz de transmitir informação a múltiplos destinatários. Além disso, pelo facto da aplicação ter como veículo alvo UAVs de pequeno porte (Fig. 3.11), a proposta deveria ser uma solução fisicamente leve e também pequena, a fim de não prejudicar o desempenho do veículo. Finalmente, o equipamento escolhido precisaria de ser comercialmente viável.

Todas as escolhas deste projeto foram tomadas no sentido de criar uma estação de controle que suportasse múltiplos UAVs em simultâneo. Comunicação entre os UAVs simulados (por XBee e ZigBee) e interface para visualização dos dados de cada UAV foram alguns pontos discutidos e tratados.

[21] Fonte:http://www.processio.com/3d-robotics-iris-plus-motors-and-propellers/

Figura 3.12: Controle simultâneo de quatro UAVs [Leite, 2015]

Apesar dos resultados terem tido sucesso, verifica-se que se se tiver mais do que um UAV a comunicar com a plataforma, esta acaba por ficar sobrecarregada devido ao alto volume de mensagens recebidas. Isto passa-se porque a plataforma apenas dispõe de um canal de comunicação, algo que limita o desempenho do sistema. Para além disso e como se verifica na Fig. 3.12 quando se utiliza vários UAVs a interface acaba também por ficar lotada tendo em conta o elevado fluxo de mensagens.

Também foi analisado o estudo Concepção e implementação de um sistema de simulação de hardware em loop para helicópteros de UAV de pequena escala [Cai et al., 2009]. Este artigo visa desenhar e implementar uma framework de simulação para hardware em loop (HIL), num sistema de UAV personalizado (UAVs de baixa escala). O HIL é uma técnica bastante utilizada para testar, verificar e validar o desempenho em tempo real de sistemas complexos. Para além de avaliar a performance, procura garantir que as normas de segurança dos UAVs são cumpridas, antes de se realizarem testes de voo.

A framework é composta por quatro grandes módulos, cada um responsável por desempenhar um papel específico na mesma, e que juntos permitem simular em tempo real o hardware em loop sendo estes hardware a bordo, controlador de voo, estação de controlo terrestre e outro que integra todos estes módulos.

Relativamente ao hardware a bordo, esta envolve toda a configuração em termos de hardware do sistema do UAV. Esta fase divide-se em três componentes: Sistema de processamento de computadores a bordo (onde o controlo do voo é feito); Sensores Guias (conduzir os UAVs para realizar movimentos *rool*, *pitch* e *yaw*); Routers Wireless (permite a comunicação e a troca de informação entre ambos). Quanto ao controlador de voo, este procura implementar algoritmos de controlo para os UAVs. Possui uma estrutura hierárquica relativamente a loops de controlo (unidade de sistemas de controle industrial), tais como Loops Internos (rastreia os sinais de referência das velocidades e ângulo de direção do UAV ao longo dos eixos do seu corpo), Loops externos (controla o UAV para uma posição específica baseada em coordenadas Px, Py e Pz) e bloco para agendamento de voos (permite agendar caminhos de voo para missões ou tarefas designadas para UAVs). A estação de controlo terrestre, está responsável por observar e monitorizar o desempenho do sistema no seu

todo. Possui 2 vistas de análise, uma direcionada mais para visualização de dados e outra mais vocacionada para gerar comandos, em tempo real, e tarefas para os UAVs. Por último, existe um módulo que agrega as três componentes anteriores num módulo compacto.
Na Fig. 3.13, observar-se como estão organizados os módulos e que relações estabelecem uns com os outros.

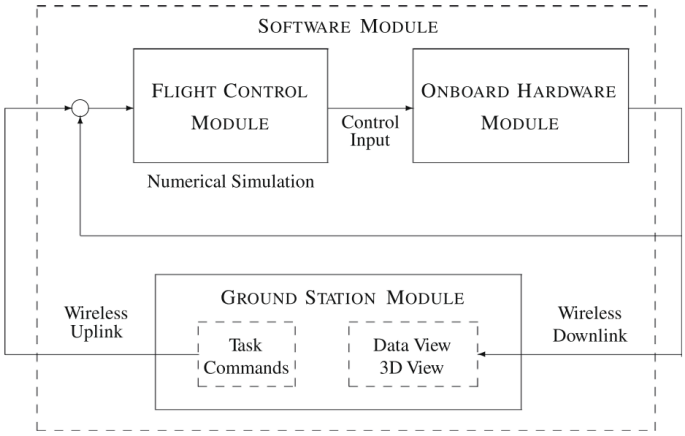


Figura 3.13: Simulação HIL em sistemas UAVs [Cai et al., 2009]

Para testar esta framework, foram construídos dois UAVs, presentes na Fig. 3.14:

Figura 3.14: Helicópteros HeLion e SheLion utilizados[Cai et al., 2009]

Testes conduzidos:

1. Movimentos de voo simples - foram conduzidos vários movimentos básicos como pairar ou seguir em frente. Estes testes visam analisar a estabilidade e o comportamento do controlo sobre o UAV de forma a poderem conduzir-se testes mais complexos;

2. Voo completo - O UAV conduz uma série de desafios com vista a obter o seu desempenho durante o voo. Movimentos como descolagem, deslizar, fazer piruetas ou espirais são alguns dos exemplos de testes conduzidos nesta fase;

3. Voo completo com outros UAVs - Bastante importante pois permite analisar os potenciais perigos que um UAV enfrenta num ambiente real. Nestes testes, a estação terrestre funciona

como o UAV líder virtual e tem como função comandar os UAVs (neste caso helicópteros) a seguir um determinado trajeto com múltiplos UAVs.

Através destes, provou-se que a framework de simulação com HIL é capaz de prever eficazmente várias situações de potencial perigo. Para uma melhor análise dos dados obtidos, foi feita uma comparação entre dados de situações reais com a informação obtida pela framework.

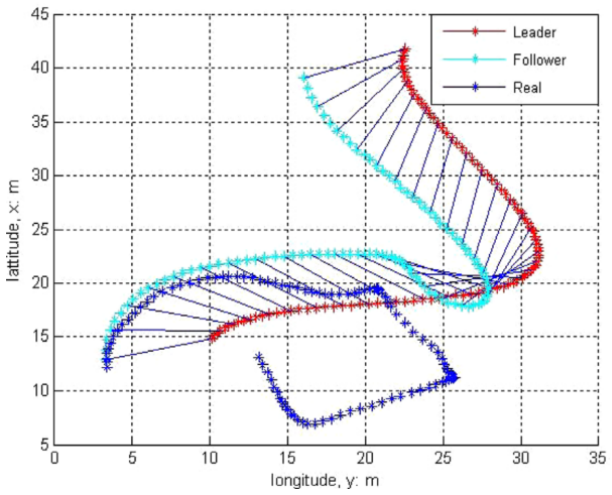


Figura 3.15: Trajetória seguida pelo veículo real, seguidor e lider [Cai et al., 2009]

Apesar de na coordenada (21, 20) da Fig. 3.15, o lider e o seguidor tomarem um comportamento estranho (teste foi conduzido em condições extremas) a análise acabou por ser satisfatória visto que dadas as condições conseguiu-se prever uma parte da trajetória.

Teve-se em conta outro projeto titulado Simulação abrangente de um Quadrotor através de um Sistema Operativo Robótico (ROS) e Gazebo [Meyer et al., 2012]. Este trabalho tem como missão fornecer uma plataforma de simulação destinada a UAVs, nomeadamente a quadrotors, capaz de cobrir voos dinâmicos e realistas com câmara e sensores e ainda suportar uma fácil integração de middleware (responsável por gerir a complexidade e heterogeneidade do hardware e das aplicações) robótico já existente.
A plataforma de simulação foi construída com base numa ferramenta para simulação de robôs, o Gazebo[22] e com recurso ao ROS, middleware que gere o hardware e as configurações do robô.
O modelo dinâmico de simulação idealizado para o quadrotor está assente em várias premissas sendo elas a geometria, o modelo dinâmico e nível sensorial.
Quanto à geometria, esta está Modelada através da ferramenta open source Blender[23]. A geometria visual foi pensada num formato COLLADA (formato com cores, texturas e material específico) e modelada como uma rede .stl. No modelo dinâmico, uma das grandes vantagens do quadrotor encontra-se na fácil gestão do sistema de propulsão e no sistema de direção. Possui quatro motores independentes fixos compostos por hélices, sendo que cada par de hélices roda numa direção para

[22]Fonte: http://gazebosim.org/
[23]Fonte: https://www.blender.org/

evitar "guinadas" durante movimentos de rotação ou de inclinação de voo. O estado e controle que não sendo específicos na simulação, desempenham um papel fundamental na estabilização e controlo do quadcopter. Conciliam a informação dada pelos sensores e pelos motores. Por fim existe o nível sensorial, que tendo em conta que num ambiente de simulação fatores como altitude, velocidade e posição são complicados de se obter, é necessária a inclusão de sensores para obtermos modelos precisos. Os sensores são independentes ao Gazebo e podem ter diferentes naturezas. Esta última premissa, engloba vários fatores tais como a unidade de medida inercial (um dos sensores mais importantes durante o voo do quadrotor. Ideal para obter respostas num curto espaço de tempo. A longo prazo não é aconselhável a sua utilização devido aos desvios que os sensores low-cost possuem), barométrico (modelo standard internacional da atmosfera (ISA) que descreve a pressão, temperatura e densidade a uma determinada altitude), ultrasónico (Bastante importante durante a descolagem e aterragem, e para ligar assim como desligar os motores), campo magnético (ajuda no cálculo da inclinação do quadrotor), GPS (Medidas de alcance, assim como posição e velocidades resultantes, são condicionadas por erros atmosféricos ou erros de recetor.
Apesar de não se testar a interação entre quadrotors reais e simulados, para análise de resultados e testes é feita a comparação entre informação de voos reais com os dados obtidos nas experiências de forma a tirar conclusões mais realistas, ou seja, não se testou esta interação em ambientes onde existem fatores externos capazes de influenciar os resultados.

Em último lugar olhou-se para o trabalho Simulação Realista de uma Plataforma Aérea [Carvalho, 2014]. Esta tese visa resolver alguns problemas associados aos UAVs, nomeadamente quadrotors, como por exemplo a sua reduzida autonomia e adaptação a diferentes cargas. Para tal, foram estudadas soluções com diferentes naturezas, antes de se implementar no mundo real.
Foram contempladas duas fases que abordam a simulação do quadrotor e o estudo das plataformas aéreas.
Relativamente à simulação do quadrotor, vários quadrotors (quadrotor-avião, quadrotor-dirigível e quadrotor-helicóptero) foram testados com recurso a diferentes plataformas de simulação, com vista a analisar a autonomia dos mesmos. Inclui também um módulo de sensorização que simula sensores que através da leitura dos mesmo, prevê o estado do veículo e uma parte de controlo que tem como missão a movimentação do veículo com uma determinada orientação e posição. As plataformas de simulação exploradas foram V-REP[24] e o SimTwo[25]:

- V-REP - trata-se de um simulador especializado em simulação de robôs 3D, baseado numa arquitetura de controle distribuído, que permite aos utilizadores criar sistemas robóticos o mais realistas possíveis. É utilizado quando se pretende fazer uma monitorização completa, controle de hardware, processamento de imagem, entre outras;

- SimTwo - é uma plataforma de simulação que permite uma visualização 3D, através da biblioteca GLScene (open-source), a implementação de um controle programável com recurso

[24]Fonte:http://www.coppeliarobotics.com/
[25]Fonte:https://paginas.fe.up.pt/~paco/wiki/index.php?n=Main.SimTwo

a Pascal Script e ainda simular corpos rígidos através de um ambiente de desenvolvimento aberto (ODE);

Depois da análise da informação recolhida em experiências feitas aos dois simuladores, chegou-se à conclusão que o SimTwo seria o melhor para resolver os problemas enunciados anteriormente visto que, para além de se ter em conta os muitos efeitos físicos que ajudam a obter um comportamento realista da plataforma a simular, trata-se uma ferramenta que, tendo sido desenvolvida dentro da mesma instituição que a deste trabaho, permitiu uma resposta rápida a dúvidas e problemas que foram surgindo.
Relativamente ao controlo da plataforma, foi utilizado um controlador PID (Proporcional Integral e Derivativo) devido aos bons resultados que apresentou. Relativamente à sensorização, foram simulados um giroscópio, uma bússola, um acelerómetro e um *Global Position System* (GPS), através dos quais foi possível recolher informação sobre o estado atual do veículo.
Para o estudo das plataformas aéreas, foram investigadas duas soluções híbridas, uma relativa ao quadrotor-helicoptero e outra direcionada ao quadrotor-dirigível. Em relação ao primeiro, a utilização de motores elétricos traz limitações energéticas por isso, ao passar a função de manter a plataforma no ar para as hélices, a gasolina introduzida fez com que se poupasse energia.
Relativamente ao quadrotor-dirigível, depois de se acrescentar o balão, que aplica uma força ascendente, a energia precisa para descolar o veículo e mantê-lo no ar é menor.
Em suma, foram criadas duas soluções que cumpriram os objetivos principais propostos. Este projeto permitiu ainda abrir portas para novos projetos de exploração como a simulação da plataforma quadrotor-avião e ainda estudos mais aprofundados sobre a sensorização. Apesar de entrar no âmbito dos UAVs não possui objetivos semelhantes aos que se pretende cumprir com este trabalho

3.3.1 Análise às Abordagens Semelhantes

Depois de alguns exemplos que conjugam os conceitos de UAV com plataforma de simulação, é importante lembrar quais os pontos que se assemelham aos objetivos que se pretendem alcançar com este trabalho.
Todos os estudos recolhidos estão organizados por ordem de importância o que permite inferir que a primeira abordagem apresenta bastantes semelhanças com o objetivo proposto deste trabalho, enquanto que a última apresenta poucas semelhanças, contudo, com alguma relevância.
A primeira abordagem (Uma plataforma de simulação simbiótica para os Quadcopters baseados em agentes) apresenta-se com pontos bastantes interessantes e é aquela que mais se aproxima do que se pretende implementar. Funciona com base em quadcopters, faz uso de uma plataforma de simulação para objetos 3D, procura integrar vários veículos autónomos num ambiente (configurável) através de um protocolo de comunicação que ajuda na cooperação entre os mesmos, utiliza um sistema SMA (Sistema Multi Agente) que permite manter o processamento mais pesado e as tomadas de decisão à parte do quadcopter e, acima de tudo, procura estudar o conceito de relação simbiótica, ou seja, a relação que o mundo real pode estabelecer com o ambiente virtual através de módulos XBee. Não se procurou utilizar esta abordagem no nosso projeto pelo simples facto

de se querer explorar o quadcopter e integrar o mesmo de raiz numa plataforma de simulação já em desenvolvimento.
A segunda abordagem (Simulação Online para Planeamento de Caminho Adaptativo de UAVs) tal como a abordagem anterior procurar analisar a relação simbiótica e comparar esta com outros métodos de planeamento de caminhos (offline e busca exaustiva) sendo que em alguns cenários a relação simbiótica revelou um melhor desempenho.
A terceira abordagem (Plataforma de controle para Múltiplos Veículos Aéreos Não Tripulados (VANTs)) foca-se na construção de um controlador para múltiplos veículos aéreos não tripulados através da extensão de uma plataforma já existente (DroidPlanner). Possui alguns detalhes interessantes como controlar um veículo com um dispositivo Android através de protocolo de comunicação MAVLINK, faz uso de um simulador que permite analisar e testar o desempenho dos UAVs e ainda dotar os mesmos com um comportamento preciso e sem hardware, através da arquitetura SIL (Software em Loop). Apesar de ser aplicável apenas a veículos simulados, pensou-se que género de UAV estaria preparado para incorporar a funcionalidade de interação com o mundo simulado.
Relativamente à quarta (Concepção e implementação de um sistema de simulação de hardware em loop para helicópteros de UAV de pequena escala), utiliza uma arquitetura Hardware em Loop (HIL) em tempo real, que permite medir o desempenho e a segurança dos UAVs. Possui como semelhanças o facto de se desenvolver uma framework para simulação capaz de ser testada com recurso a UAVs de pequenas dimensões (helicópteros HeLion e SheLion). Para além disso, alguns testes conduzidos apresentam algumas semelhanças com aqueles que pretendemos implementar (movimentos básicos, voo completo sozinho e com vários quadcopters (simulados e real)). Apesar destas semelhanças, grande parte dos objetivos deste trabalho são diferentes sendo o único semelhante inclui a utilização de UAVs de baixo porte. Outro fator de distinção é o facto de não explorar a relação entre UAV real e simulado.
Por fim surgem a quinta abordagem (Comprehensive Simulation of Quadrotor UAVs using ROS and Gazebo) e sexta abordagem (Simulação Realista de uma Plataforma Aérea) que tendo objetivos e planeamentos diferentes, recorrem a plataformas de simulação (Gazebo e SimTwo, respetivamente) para resolver problemas (por exemplo baixa autonomia) ou oferecer novas ferramentas de simulação capazes de analisar a fiabilidade e desempenho dos UAVs (por exemplo quadrotors). Embora não testem os seus instrumentos finais em UAVs reais, fazem uso de informações de voo reais de modo a obter modelos e resultados mais precisos.

Capítulo 4

Planeamento

Este capítulo destina-se essencialmente à organização e planeamento do trabalho realizado. Numa primeira instância, é abordada a tecnologia em utilização, onde é dado destaque a alguns pilares que sustentam este projeto. De seguida, é feita uma apresentação de todos os materiais (hardware e software) utilizados na montagem do drone e comunicação do mesmo com a plataforma de simulação, acompanhada com uma descrição do papel que cada um desempenha. É ainda demonstrado e justificado a abordagem pensada para a realização do objetivo proposto para este trabalho. Por fim, é apresentado o plano de trabalho e os sprints previstos antes da fase do desenvolvimento, acompanhado com uma análise de risco que nos permite discernir quais as fases que inspiraram mais cuidados. São ainda destacadas algumas alterações que foram feitas no plano de trabalho, durante a fase de desenvolvimento.

4.1 Escolhas e Necessidades Tecnológicas

Todos os projetos têm regras que devem ser seguidas e decisões que precisam de ser tomadas, e este trabalho não foge à regra. Pelo facto de este trabalho surgir no seguimento de outras propostas [Câmara, 2013] torna-se ideal a continuidade de várias tecnologias ainda em utilização. O motor de simulação *FSX (Flight Simulator X)*, *Agent Service* para a comunicação entre agentes, utilização da linguagem C# entre outras tecnologias são algumas das componentes exploradas e provenientes de trabalhos passados.

4.1.1 *Flight Simulator X (FSX)*

A escolha da plataforma de simulação a utilizar foi feita com base nos seguintes fatores:

- Motor de simulação - constitui-se como o elemento com maior importância uma vez que é aqui que estão centradas as funcionalidades do simulador. Num voo simulado, alguns motores apenas consideram fatores básicos (elevação, peso, entre outros), fatores esses que

em alguns casos não são suficientes visto que, durante um voo real, existem fatores externos (vento, turbulência, entre outros) que interferem com o voo. O tempo e a física envolvida num ambiente simulado assim como a cinemática são alguns pontos bastante importantes num motor de simulação;

- Gráficos - detalhes visuais não são muito importantes para a simulação científica mas têm sempre alguma influência quando um utilizador procura escolher uma plataforma de simulação. Aspetos como o realismo e detalhes do ambiente (terrenos, aeronaves e tempo) têm algum peso nessa escolha;

- Documentação - o facto de ter APIs open-source e de possuir uma documentação bem composta ajuda na compreensão e desenvolvimento de novas funcionalidades no simulador;

- Abertura e Versatilidade da plataforma de simulação - As plataformas para as quais estão disponíveis, tal como a versatilidade que apresentam permitem, ao utilizar experimentar as plataformas com diferentes configurações e cenários. Para além disso, a existência de extras ajuda a suprimir falhas no sistema e a fornecer novas experiências aos utilizadores.

Antes do desenvolvimento do presente trabalho, já tinha sido feita uma análise sobre qual a plataforma de simulação ideal a utilizar, tendo a escolha recaído no FSX [Silva, 2011] no entanto, decorridos alguns anos surgiram outras plataformas com pontos interessantes.
Foram analisadas várias plataformas de simulação (Tabela 3.2) como o *X-Plane*, *Flight Simulator X*, *DroneSim Pro*, *Real Flight*, *AeroFly*, *LIFTOFF* e *Heli X* e chegou-se à conclusão que o *FSX* continua a ser a melhor opção apesar de não se constituir como a melhor opção em alguns atributos analisados (características, plataformas destino e preço). Em termos de características, o *FSX* é o aquele que oferece melhores condições quando comparadas com outras soluções visto englobar várias funcionalidades importantes como o alto grau de realismo, modo online e multiplayer. Quanto às plataformas destino, o *FSX* é dos menos versáteis devido ao facto de apenas funcionar em Windows. Outras soluções como o X-Plane estão disponíveis em Windows, Mac e Linux. Relativamente a custos, o *FSX* é dos mais acessíveis apesar de a plataforma de simulação LIFFTOF ser mais barata. Outro aspecto para a escolha do *FSX*, centra-se no facto de possuir bastantes expansões e de estar em constante evolução.
Tendo em conta esta avaliação, chegou-se à conclusão que o **FSX** continua a ser a opção mais viável para o desenvolvimento deste projeto. Caso fosse necessário alterar a plataforma de simulação, esta acarretaria um grande esforço (tempo) por parte da equipa de desenvolvimento.
O *Flight Simulator* foi originalmente desenvolvido pela Microsoft em 1982 sendo atualizada em 2006 para a versão X (atual). Desde então que se tem mostrado como um dos melhores simuladores para aqueles que planeiam explorar a aviação.
Outro pormenor interessante que tem grande impacto neste projeto é a facilidade de comunicação que o FSX apresenta com o *SimConnect*. Esta interface permite a troca de variáveis entre o simulador e as *third parties* e possui uma API onde a troca de dados é feita com base em eventos pré-definidos. Este protocolo de comunicação entre servidor e cliente possui uma enormidade de

funções que permitem lidar com a meteorologia, missões, objetos AI, menus no jogo, manipulação e acesso a dados, entre outros recursos úteis. Também inclui extensa documentação, composta por vários exemplos detalhados [Microsoft, 2008]. Embora não ofereça muitas opções em relação aos efeitos visuais, a principal vantagem do *SimConnect* é o facto de poder ser executado em tempo real durante a simulação. Para além disso, permite ao utilizador utilizar todas estas funcionalidades em várias linguagens de programação. Estas são C/C++, Java e .NET com recurso a jSimConnect.

4.1.2 *Agent Service*

Um dos requisitos do projeto a desenvolver é a utilização de um sistema multi-agente para comunicação entre os diversos tipos de agentes presentes no sistema (aéreos, aquáticos e terrestres). Para tal estudaram-se vários modelos que seguem os princípios *FIPA (Foundation for Intelligent Physical Agents)* [Silva, 2011]. Inicialmente, pensou-se em utilizar um protocolo de comunicação baseado em JAVA, contudo, veio a provar-se que a utilização de uma das plataformas de comunicação mais conhecida e utilizada, como por exemplo o *JADE*, não se traduz num melhor desempenho quando comparado com outras opções [Hallenborg, 2008]. Chegou-se a considerar as frameworks *CAPNET* e o *ACENET*, contudo, o desenvolvimento lento verificado nos últimos anos tornou esta opção pouco viável. Assim sendo, escolheu-se a plataforma AgentService. Esta plataforma foi desenvolvida por Vecchiola [Boccalatte et al., 2004] e caracteriza-se por ser uma tecnologia simples, flexível e capaz de lidar com um grande fluxo de mensagens. Esta tecnologia funciona bem em ambientes .NET e permite fazer a integração de aplicações externas, ajudando a integrar softwares que não foram desenvolvidos especificamente para a plataforma. Através de uma das funcionalidades do *Agent Service*, *ExternalRuntime*, estas aplicações possuem a capacidade de aceder todos os serviços fornecidos pela plataforma. Outro aspeto do *Agent Service* centra-se no facto de poder funcionar num ambiente federado. Este fator permite que várias instâncias do AgentService sejam executadas em diferentes computadores dentro de uma rede, levando a um melhor balanceamento de carga e mobilidade do agente.

4.1.3 Linguagem C#

Outro aspeto a ter em conta e que tem alguma importância no projeto é a linguagem de programação a utilizar. Tendo em conta que a plataforma de simulação em utilização foi desenvolvida com recurso a C# também será esta a linguagem a utilizar na evolução da plataforma de simulação (integração do quadcopter com a mesma). Apesar de a ferramenta arduino IDE ter sido desenvolvida com recurso a JAVA, ela utiliza C/C++ para a programação dos arduinos em causa (UNO e MEGA2560).

4.1.4 Outros Fatores

Esta escolha do hardware traz repercussões para a programação do quadcopter e comunicação com a plataforma de simulação que podem ser benéficas ou prejudiciais. Tendo em conta que o

objetivo deste projeto passa por utilizar um veículo low cost e por isso material de baixo custo, é expectável que os objetivos sejam cumpridos (voo e integração) apesar da ocorrência de algumas falhas que podem acontecer e que são naturais quando se conjuga hardware de diferentes naturezas.
Relativamente à programação do drone, apesar de o código C# ser independente dos componentes do UAV, este é desenvolvido tendo em conta as características do material utilizado na construção do quadcopter (a eficácia do código pode ser influenciada pela qualidade do material escolhido). Quanto à comunicação, espera-se que os rádios utilizados consigam receber e enviar dados sem problemas. Existem alguns fatores externos que podem afetar o desempenho destes e que serão abordados mais à frente.

4.2 Hardware e Software a utilizar

Em termos de Hardware, teve-se em conta vários materiais tendo em conta o seu custo e peso (afeta o voo do quadcopter). Recorreu-se a vários materiais tais como: Kit quadro F450 Quadcopter, peças Eachine, 2 pares de hélices (mais 4 pares suplentes) para multicopter FPV RC quadcopter, Bateria ZOP Power 11.1V 2200MAH 3S 30C, 4 peças 30A SimonK ESC, 4 peças de motor 920kv Brushless, Módulo Sensor MPU6050, Módulo ultrasónico HC-SR04, GPS NEO 6M, 2 Rádios NRF24L01 + PA + LNA, Arduino MEGA2560, Arduino UNO e um Kit FlySky FS-i6. Foi ainda utilizado material básico que ajudou na montagem e na ligação de todos os componentes do quadcopter (voltímetro, material de soldadura, etc). Mais informações sobre as características e como utilizar estes materiais são dados na secção 4.1 do manual (Anexo A).

No que toca ao Software, serviu-se particularmente de três componentes:

- Visual Studio - ferramenta rápida, com formato *user-friendly* e equipada com bastantes ferramentas internas tais como a integração de um servidor TFS. Este servidor funciona como um repositório e permite a partilha de código entre a equipa de desenvolvimento;

- Arduino IDE - outra ferramenta bastante útil e importante para quem pretende fazer a programação de uma placa Hardware (Arduino UNO, MEGA2560, entre outros). Possui uma boa documentação e exemplos que ajudam a entender todo o tipo de conceito inerente aos Arduinos.

Investigou-se vários controladores de voos disponíveis no mercado tendo em conta o género e aplicação do quadcopter sendo o escolhido o seguinte:

- YMFC-ALL [Brooking, 2015] - Software responsável pelo controlador de voo do quadcopter. Possui três ficheiros que permitem ao quadcopter voar livremente.

4.3 Abordagem ao Problema

O problema central deste trabalho é o de montar e integrar um quadcopter numa plataforma de simulação para missões multi-veículo. Com este objetivo em mente foram analisadas abordagens com algumas similaridades (secção 3.3) e o material a utilizar, tendo-se retirado algumas conclusões importantes para o cumprimento do objetivo proposto (secção 4.2). A incompatibilidade entre os sensores disponíveis (por exemplo, GPS com o sonar) e a necessidade de alguns sensores utilizarem portas específicas foram alguns obstáculos encontrados. Os rádios NRF24 utilizam obrigatoriamente algumas portas do Arduino tal como o recetor de sinal do FlySky FS-i6 no entanto, não é possível utilizar-se os dois ao mesmo tempo pelo facto de não haverem duas portas com características semelhantes para os dois componentes. Outro problema que foi superado foi o reduzido número de portas (por exemplo, o Arduino UNO apenas possui 14 portas digitais tendo 6 delas propriedades *PWM*). Poderia criar-se uma rede com vários arduinos (cada um com alguns sensores) que iria ajudar a suavizar esses fatores, contudo, esta solução exigiria mais tempo de espera pela chegada de novas peças, algo que poderia perturbar o planeamento feito.
Perante estes inconvenientes optou-se por dividir o projeto em duas grandes fases: Voo do Quadcopter e Integração do Quadcopter na Plataforma de Simulação. Apesar desta divisão e dos problemas enumerados anteriormente, é possível conciliar as duas mas com algumas limitações (abordado na secção 5.3.8.3) sendo a segunda fase aquela que representa mais importância para este projeto. Esta secção tem assim como grande objetivo distinguir as duas fases, assim como relatar o principal objetivo de cada uma.

4.3.1 Voo do Quadcopter

Esta fase tem como intenção estudar, montar e testar os componentes de forma a permitir ao drone fazer um voo sustentável. Para cumprir com este requisito aspetos, como a disposição do material (FlySky FS-i6, ESCs, motores *brushless*, entre outros), ligação, configuração e calibragem de cada um dos componentes à placa Arduino (UNO ou MEGA2560) e mecânicas de voo são temas a ter em conta na montagem do quadcopter e que têm uma grande influência no voo do mesmo. Caso alguns dos aspetos anteriores seja negligenciado ou simplesmente mal feito, é provável que o quadcopter não consiga voar assumindo um comportamento com *flips* (quadcopter roda sobre si mesmo realizando um movimento inesperado que pode colocar em causa o mesmo e pessoas que se encontrem nas proximidades). Através de uma configuração geral correta do quadcopter seria possível executar os vários tipos de movimentos mencionados na secção 2.3.

4.3.2 Integração do Quadcopter na Plataforma de Simulação

Para além de se ter a intenção de fazer com que o quadcopter voe, pretende-se que seja possível integrar o quadcopter na plataforma de simulação já existente. Por forma a realizar este propósito, tenciona-se criar dois módulos que juntos permitem representar o quadcopter real na plataforma em causa. Numa primeira instância, tem-se como desejo implementar e desenvolver um módulo

capaz de recolher e enviar os dados dos sensores (GPS, Sonar, Bússola) ligados ao Arduino para a plataforma através de um middleware que terá como função fazer ponte entre o quadcopter e a mesma (estação de controle de terrestre). A segunda etapa centra-se em representar o quadcopter (adotou-se um modelo *FSX* para o quadcopter) na plataforma e interpretar os dados dos sensores na mesma. Sempre que há a variação dos fatores latitude, longitude, altitude e orientação a plataforma é notificada sobre essa variação, ou seja, se movermos o quadcopter e alguma leitura dos sensores se alterar, essa alteração é representada na plataforma (o quadcopter virtual "mexe-se"). Outra questão bastante pertinente e que também foi abordada nesta fase foi a realidade aumentada da plataforma. Esta funcionalidade permite que se crie uma associação entre o UAV real e virtual, ou seja, o quadcopter real é representado na plataforma de simulação através de um quadcopter virtual e qualquer movimento aplicado no veículo real é replicado no quadcopter virtual.
No próximo capítulo todas estas questões são abordadas. Serão ainda explicados as diferentes funcionalidades que permitem testar o voo do drone e recolha de informação dos sensores. Um deles (modo testar altitude e motores do quadcopter) permite conciliar o voo do quadcopter com a recolha de informação proveniente dos sensores.

4.4 Plano de Trabalho

E apresentado um cronograma com uma divisão de tarefas a desenvolver. É apresentada uma descrição sobre as principais tarefas que compõem o projeto assim como o risco que cada uma das fases possui neste trabalho.
De referir que a divisão de tarefas e a análise de risco foram pensadas antes de se iniciar o desenvolvimento do mesmo. São ainda explicadas algumas alterações levadas a cabo no planeamento inicial acompanhadas com a sua justificação.

4.4.1 Divisão de Tarefas

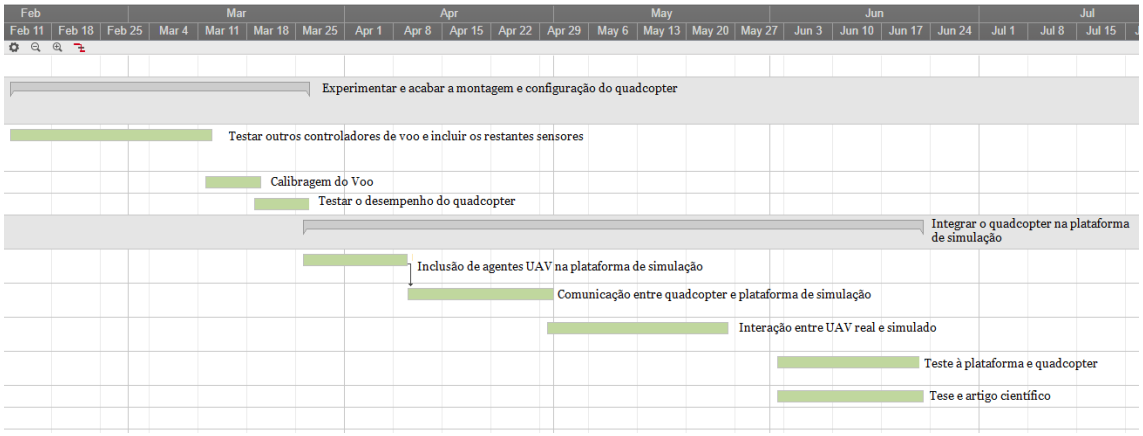


Figura 4.1: Diagrama de Gantt com o planeamento do trabalho desenvolvido

De acordo com a Fig. 4.1, optou-se por dividir o projeto em duas fases:

- Experimentação, configuração e finalização do quadcopter - ocupa uma duração de 6 semanas e envolve a experimentação, integração e avaliação de vários softwares de controlo de voo no quadcopter. Inclui a possibilidade de se proceder a alterações na disposição dos componentes do quadcopter de modo a promover um melhor desempenho (melhor balanceamento do peso) e um melhor aproveitamento do espaço. A afinação do hardware (motores e ESCs) e dos movimentos de voo também é de se ter em conta com vista a efetuar um voo sustentado, capaz de executar as diferentes manobras. A movimentação do quadcopter deve estar sujeita a testes para averiguar o seu desempenho;

- Integração do quadcopter na plataforma de simulação - ocupa uma duração de 10 semanas e contempla o desenvolvimento de várias funcionalidades. A plataforma já existente deve ser alvo de estudo e adaptada para a aceitação de agentes do tipo UAV. O estudo, escolha e implementação de um protocolo de comunicação é outro aspeto importante. Este deve permitir a comunicação entre a plataforma de simulação e quadcopter real estando responsável pelo envio e receção de dados e de instruções. A comunicação entre UAVs reais e os agentes UAVs simulados será feita em tempo real com a missão de observar os diferentes comportamentos que os mesmos podem manifestar. Todos os intervenientes aqui explorados devem ser alvo de teste tendo em vista a melhoria e introdução de novas funcionalidades. Numa fase final é redigido um relatório, a integrar na dissertação, acompanhada com um artigo científico onde são relatadas as experiências e resultados obtidos durante a fase de desenvolvimento deste projeto.

4.4.2 Análise de Risco

Esta secção visa averiguar os riscos e a probabilidade de os mesmos ocorrerem durante o desenvolvimento do projeto, assim como alguns métodos para mitigar os seus efeitos. Optou-se por dividir os riscos existentes em duas categorias: tecnológicos e humanos.
Os riscos tecnológicos incluem os riscos relacionados com o material utilizado na construção do quadcopter, programação do quadcopter, a plataforma de simulação e com a comunicação entre todos estes componentes. Já os riscos humanos têm em conta aqueles que são intrínsecos ao autor do projeto e que estão relacionados com a dificuldade e tempo despendido na realização do mesmo. A Tabela 4.1 apresenta os riscos existentes.

Tipo de Risco	Nº	Risco	Probabilidade	Impacto
Tecnológico	1	Falhas no protocolo de comunicação para a integração do quadcopter com a plataforma de simulação	Média	Alto
Tecnológico	2	Deficiências ou incapacidade dos materiais utilizados na construção do quadcopter	Média	Médio
Tecnológico	3	Informação pouco precisa e realista	Alta	Baixo
Tecnológico	4	Tecnologias pouco atualizadas e documentadas	Baixa	Baixo
Humano	5	Desenvolvimento de funcionalidades com deficiências ou diferentes das pedidas	Baixa	Médio
Humano	6	Atrasos da realização de tarefas	Média	Médio
Humano	7	Variação entre o tempo despendido no desenvolvimento de uma tarefa com o tempo previsto	Alta	Médio
Humano	8	Discrepância entre a dificuldade sentida no desenvolvimento de uma tarefa com a dificuldade prevista	Média	Médio

Tabela 4.1: Fatores de risco no desenvolvimento do projeto

Este projeto procura analisar dois grandes pilares: o quadcopter e a plataforma de simulação. Todavia, existe uma outra vertente a considerar e que será uma das grandes preocupações deste trabalho: o tipo e protocolo de comunicação a utilizar. Este cuidado adicional deve-se essencialmente ao facto de ser esta última a fazer a conexão e integração do quadcopter com a plataforma de simulação e a permitir a troca e recolha de dados e instruções. O protocolo escolhido deve ter em conta as características dos diferentes dispositivos utilizados durante as experiências e respeitar as regras impostas, de modo a facilitar a comunicação e troca de informação relevante.

Visto que a ideia do projeto passa por construir e integrar um quadcopter *low-cost* numa plataforma de simulação, é expectável a ocorrência de algumas falhas (atrasos nas respostas, perda de informação, pouca eficiência, entre outras). Estas podem ocorrer, ora porque se testou o sistema em condições que impossibilitam utilizar as capacidades de alguns componentes como os sensores (é esperado que seja complicado utilizar o GPS num ambiente fechado devido à impossibilidade de conexão ao satélite), ora porque não se respeitou os elementos utilizados na montagem do quadcopter (por exemplo, a incorreta ligação dos sensores ao arduino torna impossível a utilização das capacidades dos mesmo). A presença destas falhas pode afetar o normal funcionamento do sistema, provocar discrepâncias nos resultados obtidos (processamento da informação dos diversos sensores) e levar a conclusões irrealistas e incompreensíveis. De forma a combater estas possíveis falhas, será feita uma triagem sobre os componentes a utilizar para a montagem do quadcopter e uma maior calibragem dos mesmos de forma a maximizar o desempenho e resultados obtidos. Para acompanhar o processo de resolução das falhas serão estudadas e experimentadas diferentes

configurações na comunicação entre os elementos deste projeto com o intuito de se procurar uma melhor resposta entre os mesmos.
Relativamente aos riscos aliados ao autor deste trabalho, o desenvolvimento e implementação de funcionalidades com deficiências ou diferentes das pedidas e atrasos no desenvolvimento do projeto são erros que precisam de uma atenção redobrada. Tendo em conta a falta de conhecimento e experiência na área e, por sua vez, em estimar prazos de desenvolvimento e implementação por parte do autor, há a possibilidade de não se cumprir os prazos estabelecidos no planeamento do projeto. Outra agravante presente na Tabela 4.1 é a noção de dificuldade do trabalho. O desenvolvimento de uma funcionalidade pode revelar-se mais complicada e complexa do que se tinha previsto anteriormente, o que conduzirá a mais trabalho e a mais atrasos. Para abrandar ou solucionar os efeitos destes riscos, será conduzida uma pesquisa detalhada sobre os conceitos a utilizar no projeto, acompanhada com a sua discussão com o orientador e co-orientador, de forma a que se esclareçam incorreções ou dúvidas. Será necessário também uma atenção especial na recolha e leitura dos dados obtidos pelos sensores devido aos problemas já enunciados.

4.4.3 Plano de Trabalho vs Trabalho Realizado

Por vezes, quando surgem problemas ou imprevistos, há a necessidade de se reformular o plano idealizado. Neste projeto, a principal razão para modificação do plano prendeu-se com alguns imprevistos que apareceram tanto na fase do voo do quadcopter como na sua integração na plataforma de simulação.
No que toca ao voo do quadcopter, após a sua montagem e configuração, experimentou-se fazer o mesmo voar todavia verificou-se que este possuía alguns materiais com defeito. Estes materiais, nomeadamente os ESCs e motor, acabaram por inibir o desempenho e por sua vez alterar os resultados. Após o descobrimento deste problema e consequente tempo de espera para a chegada de novas peças (os tempos de espera variaram entre 1 a 2 meses), partiu-se para a análise da plataforma de simulação. Já era esperado que esta fase exigisse algum tempo devido à sua complexidade e inexperiência do autor deste trabalho contudo, esta fase veio a revelar-se mais complicada que o previsto. Perceber como o *SimConnect* atualiza as variáveis relativas ao UAV virtual (altitude, latitude, longitude, entre outras) assim como enviar eventos para o agente UAV foram algumas tarefas que se revelaram difíceis de entender e melhorar. Naturalmente, estes problemas acabaram por ser ultrapassados.
Com o aparecimento destes problemas acabou-se por alterar um pouco o plano de trabalho que se estabeleceu previamente. O novo plano continua a ser composto pela fase de experimentação, configuração e finalização do quadcopter e por outra relacionada com a integração do mesmo na plataforma de simulação. O tempo dedicado para a primeira fase foi alterado para 2 meses enquanto que a segunda foi também ocupada por 2 meses.
Esta alteração acabou por ser benéfica para a realização deste trabalho visto que se conseguiu cumprir com os prazos impostos sem comprometer os objetivos propostos.

Capítulo 5

Implementação

Neste capítulo serão apresentadas todas as funcionalidades desenvolvidas durante a fase de desenvolvimento do projeto em questão. Numa primeira fase, será abordada a arquitetura da plataforma, sendo explicadas todas as alterações necessárias para a inclusão do quadcopter na mesma. De seguida é dado destaque ao processo de montagem do quadcopter assim como à configuração da placa controladora para que o quadcopter consiga executar as manobras de voo. Uma vez terminada a descrição da etapa do voo do quadcopter, finaliza-se esta secção com a exposição da forma como foi feita a integração do UAV na plataforma de simulação, respeitando as normas impostas pela *NATO*.

5.1 Arquitetura do sistema

Uma vez finalizada a montagem e configuração do quadcopter, é importante estudar a forma como integrar na plataforma de simulação.
A plataforma de simulação, desenvolvida sob os princípios destacados em [Silva, 2011], tem sido desenvolvida em dissertações passadas como por exemplo [Neto, 2017] de forma a oferecer uma solução mais completa e a baixo custo em comparação com outras disponíveis no mercado. Esta plataforma de simulação é composta por vários módulos (cerca de 10) contudo a atenção foi concentrada nos módulos presentes na Fig. 5.1.

Os módulos comunicam uns com os outros estando também dependentes dos dados de alguns dos outros. Destas componentes, a componente crucial da plataforma é *Simulator* (*FSX*). Este módulo funciona como um motor de jogo capaz de simular e acompanhar os vários eventos que podem decorrer num ambiente simulado. Está interligado com os outros módulos através do *Simconnect* permitindo fornecer informação aos vários tipos de agentes e assim capacitar os mesmos de tomarem decisões.
A componente *ATC Agent* tem como objetivo simular um controlador aéreo tendo a capacidade de gerir aeroportos e espaços aéreos. Mais informações em [Neto, 2017].

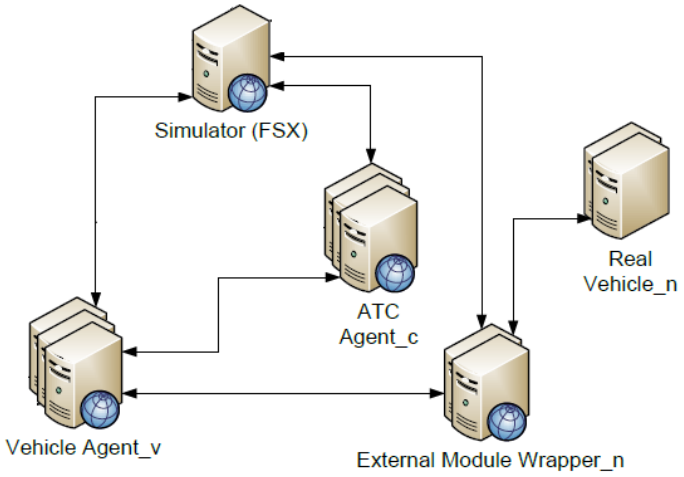


Figura 5.1: Parte da arquitetura global composta por alguns módulos da plataforma de simulação preexistente [Silva, 2011]

A componente *Vehicle Agent* permite controlar uma aeronave através da alteração de variáveis como altitude, velocidade e direção. Com este projeto, este componente sofreu algumas alterações tendo em vista a integração do quadcopter na plataforma de simulação.
Por fim, notem-se as componentes *External Module Wrapper* e *Real Vehicle* que serão aprofundadas de seguida e que, tal como *Vehicle Agent*, estarão em destaque visto serem as componentes com mais foco neste projeto.

5.1.1 Arquitetura Proposta

A arquitetura do sistema desenvolvida é semelhante à anterior sendo que as alterações feitas residem nos módulos *Vehicle Agent*, *External Module Wrapper* e *Real Vehicle*.

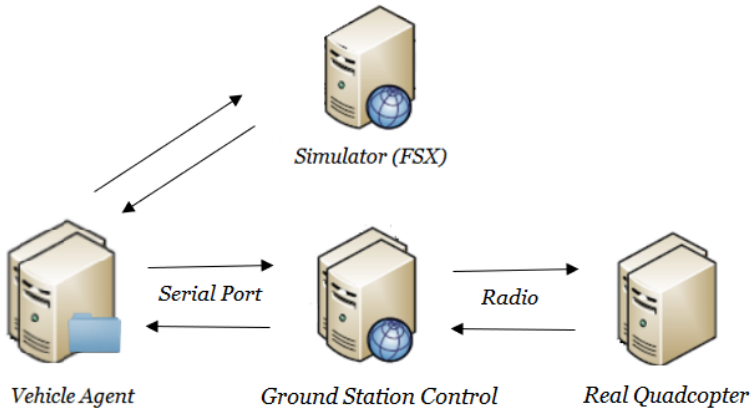


Figura 5.2: Módulos alterados para a integração de UAVs no sistema

Na Fig. 5.2, temos três grandes módulos, um que diz respeito ao *Real Quadcopter* e que envolve a montagem e configuração do quadcopter ao nível do voo e de sensores, outro que está relacionado com a Estação de Controlo e que está responsável por fazer a ponte entre o quadcopter real e a plataforma de simulação e, por último, o *Vehicle Agent* que está encarregue da gestão dos vários tipos de veículos (incluindo UAVs) e que possui um submódulo *Wrapper* (razão pela qual existe uma pasta no *Vehicle Agent* da Fig. 5.2) que trata da comunicação entre UAVs e dos dados provenientes do quadcopter real (projetando os dados para o avatar do mesmo na plataforma de simulação) entre outros aspetos.

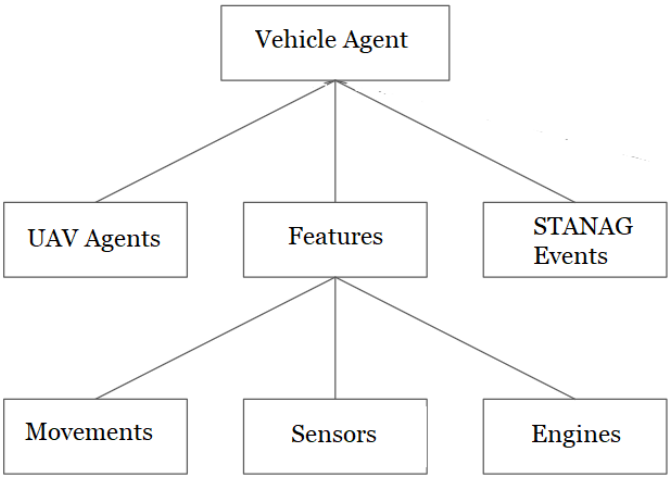


Figura 5.3: Estrutura do módulo *Vehicle Agent*

Como se observa na Fig. 5.3, o módulo *Vehicle Agent* apresenta um grande peso na integração do quadcopter na plataforma de simulação uma vez que está responsável pelo tratamento dos agentes UAV, modos presentes no sistema (permitem testar diferentes funcionalidades) e eventos *STANAG*. Todos estes tópicos são abordados e detalhados mais à frente. A montagem e configuração do quadcopter e da estação de controlo para o voo e para a integração na plataforma não estão incluidos neste módulo mas sim no módulo *Real Quadcopter*.
Em termos de comunicação entre o quadcopter real e a estação de controlo, esta é feita com recurso a rádios NRF24, enquanto que entre a estação de controlo e a plataforma de simulação é feita com recurso à porta de série (USB). A informação flui da estação para a plataforma e vice versa com uma velocidade que depende da velocidade máxima da porta de série disponível.

5.2 Voo do Quadcopter

Nesta fase é descrito todo o progresso alcançado relativamente ao voo do quadcopter. É explorado o material utilizado, o processo de montagem do quadcopter e a configuração do controlador rádio e da placa controladora.

5.2.1 Material utilizado

Antes de explorar a montagem do quadcopter importa analisar que material estará envolvido na mesma. Na secção 4.2 foi apresentado todo o material que esteve envolvido neste projeto; todavia, aqui apenas são destacados aqueles que têm uma maior preponderância para o voo do quadcopter.

(a) Quadcopter desmontado [AliExpress, 2017e] (b) Quadcopter montado [AliExpress, 2017e]

Figura 5.4: Quadro do quadcopter

Mais informações sobre as características e como utilizar estes materiais são dados na secção 4.1 do manual (Anexo A). Os restantes materiais utilizados no voo encontram-se na próxima página.

5.2.2 Montagem do quadcopter

O processo de montagem é um procedimento simples e rápido de se realizar mas que necessita de ser tratado com bastante cuidado e atenção. É necessário seguir um esquema[1] de forma a incorporar os vários componentes e a possibilitar o voo do quadcopter.
Realizaram-se as seguintes etapas:

- Montar o frame do quadcopter com base nas peças da Fig. 5.4a de forma a que o mesmo tenha a disposição da Fig. 5.4b;
- Incorporar o arduino UNO (Fig 5.5a) no nível superior do quadcopter;
- Colocar o sensor MPU6050 (Fig. 5.10c) no nível inferior do quadcopter. Este sensor possui vários sinais sendo os mais importantes o GND, VCC, SDA e o SCL. A ligação deste componente ao arduino UNO é demonstrada de seguida;
- Colocar o recetor FS-IA6 (Fig. 5.5d) na frente no nível inferior do quadcopter. O recetor faz uso de quatro canais de comunicação e do VCC e GND. A ligação deste componente ao arduino UNO é demonstrada de seguida;

[1] Fonte:http://www.brokking.net/YMFC-3D/YMFC-3D_schematic.jpg

(a) Arduino UNO [Arduino, 2018b]

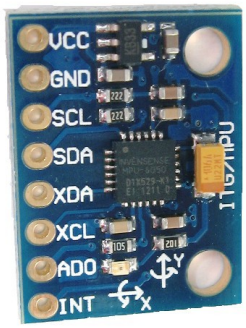
(b) *ESCs Simonk* [AliExpress, 2017a]

(c) Rádio transmissor *FlySky FS-i6* [AliExpress, 2017d]

(d) Recetor rádio *FlySky FS-IA6* [AliExpress, 2017d]

(e) Motores [AliExpress, 2017b]

(f) Hélices [AliExpress, 2017c]



(g) Sensor MPU6050 [Arduino, 2018d]

(h) Bateria [RCMoment, 2017]

Figura 5.5: Restantes Materiais na montagem do quadcopter

- Colocar a bateria (Fig. 5.5h) no nível inferior do quadcopter, os motores (Fig. 5.5e) nas extremidades dos braços do quadcopter e as hélices (Fig. 5.5f) em cima dos motores;
- Colocar os ESCs (Fig. 5.5b) nos braços do quadro do quadcopter um pouco afastados dos motores. A ligação deste componente ao arduino UNO é demonstrada de seguida;

Tanto as hélices como os motores necessitam de ter em conta o sentido de rotação. A posição dos ESCs também é relevante na disposição do quadcopter uma vez que as ligações que os mesmos perfazem com os motores têm influência no sentido giratório dos motores.

Posição	Porta Arduino	Escs	Sentido Motor	Sentido Hélice
Frente Direita	4	1	anti-horário	anti-horário
Traseira Direita	5	2	horário	horário
Traseira Esquerda	6	3	anti-horário	anti-horário
Frente Esquerda	7	4	horário	horário

Tabela 5.1: Noções importantes sobre os ESCs, motores e hélices

É importante que sejam seguidos os passos mencionados anteriormente e respeitada a informação contida na Tabela 5.1, caso contrário o quadcopter não conseguirá manter um voo sustentado e apresentará deficiências tanto no seu voo como nos dados recolhidos.

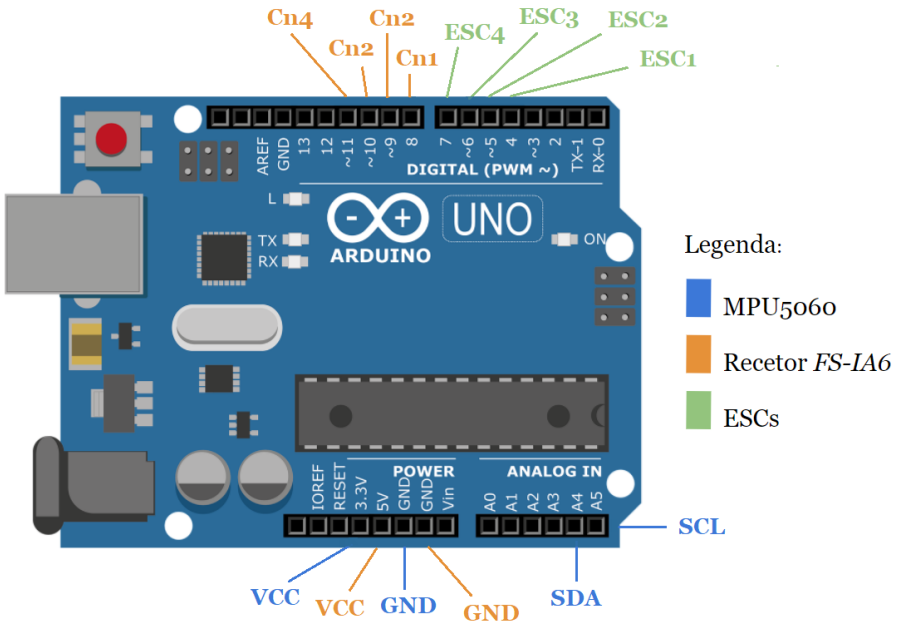


Figura 5.6: Ligação dos componentes ao arduino UNO

Ligaram-se os componentes MPU6050, recetor FS-IA6 e ESCs ao arduino UNO com base nas ligações dispostas na Fig. 5.6. A Fig. 5.7 apresenta o quadcopter depois da montagem com base

nos materiais referidos. Na secção 6.1.4 são detalhadas todas as condicionantes do voo entre as quais se destaca o calibragem dos componentes e peso do quadcopter.

Figura 5.7: Hardware do Quadcopter preparado para voo

Todo este processo de montagem está mais detalhado na secção 5.2 do Manual (Anexo A).

5.2.3 Configuração do Software para o Voo do Quadcopter

Uma vez completado o processo de montagem do quadcopter, é necessário pensar no próximo passo, ou seja, no software que permitirá ao mesmo voar. Nesta fase, é explicado qual o software escolhido e a razão para a sua escolha, assim como descrita a composição do mesmo.
A análise ao software para controlo de voo foi feita na secção 3.1 tendo em conta vários parâmetros (linguagem, custo de aquisição, entre outros). Visto não haver a necessidade de apresentar um voo muito elaborado (voo com acrobacias e movimentações em simultâneo) mas sim eficiente a executar as suas manobras e capaz de mostrar um voo sustentável, escolheu-se o software mais simples e com melhor documentação, ou seja, o Project YMFC-AL[Brooking, 2015]. Adicionalmente, o facto de ter sido desenvolvido para funcionar com o arduino UNO foi outra razão para a sua escolha. Para se configurar o arduino UNO do quadcopter é necessário incluir três ficheiros que compõem o Project YMFC-All, através do Arduino IDE.
Realizaram-se as seguintes etapas:

Incluir o ficheiro "YMFC-AL_setup.ino" - está responsável pela configuração do quadcopter com o rádio transmissor *FlySky FS-i6*. Nesta fase, os canais de comunicação com recetor *FS-IA6* são testados e configurados e os eixos do giroscópio do quadcopter são calibrados de forma correta com recurso aos manípulos do rádio transmissor (movimentos *thrust*,

pitch, *yaw* e *roll*). A informação é depois guardada na memória do arduino, ou seja, no *EEPROM* (*Electrically-Erasable Programmable Read-Only Memory*) para posterior utilização por parte do *Flight Controller*. Depois de carregado este ficheiro de configuração no arduino UNO é necessário cumprir um conjunto de tarefas. Estas encontram-se detalhadas na secção 5.3 do Manual (Anexo A);

Incluir o ficheiro "YMFC-AL_esc_calibrate.ino" - permite fazer a calibragem dos quatro ESCs dispostos no quadcopter de forma a que todos os motores funcionem em sintonia e à mesma velocidade. Para além disso, permite observar se o giroscópio do quadcopter está bem calibrado e se cada um dos motores está a funcionar corretamente e em sintonia. Depois de carregado este ficheiro de configuração no arduino UNO é necessário cumprir um conjunto de tarefas. Estas encontram-se detalhadas na secção 5.3 do Manual (Anexo A);

Incluir o ficheiro "YMFC-AL_Flight_controller.ino" - principal componente que relaciona os elementos anteriores de forma a que o quadcopter voe. É tida em conta informação proveniente dos motores, ESCs, bateria, giroscópio, entre outros. Apenas requer o carregamento deste ficheiro no arduino UNO.

Depois de carregados todos estes ficheiros no Arduino, o quadcopter estará pronto para receber comandos do rádio transmissor *FlySky FS-i6* e levantar voo. Tendo em conta as condições climatéricas, o mesmo deve perceber como e quando compensar os motores.
Caso o quadcopter manifeste comportamentos estranhos ou com *flips* deve-se repetir todo o processo de configuração. Caso o quadcopter consiga voar mas assumindo comportamentos agressivos então deve-se fazer aquilo que se chama de *PID (Proportional Integral Derivative) Tunning*. O *PID* constitui-se como um algoritmo de controlo bastante utilizado na indústria. Trata-se de um procedimento com funcionamento simples e direto. Quando os resultados não são satisfatórios ou apresentam problemas executa-se o *PID Tunning*. Esta técnica permite otimizar resultados e modelos possibilitando a afinação do quadcopter através da alteração de variáveis presentes do ficheiro "YMFC-AL_Flight_controller.ino".
Após o estudo e análise ao software para controlo do voo, procurou-se adaptar o mesmo ao arduino MEGA2560 de forma a que fosse possível fazer-se mais ligações a outros componentes contudo, verificou-se que esta tarefa seria difícil de se cumprir para aqueles que não possuem conhecimentos sobre as propriedades dos diferentes arduinos e bibliotecas disponiveis. Outro dificuldade que se enfrentou foi a sensibilidade do material disponível. Em alguns casos, os materiais obtidos no estrangeiro podem trazer defeitos, algo que pode ter implicações no voo do quadcopter. Neste projeto, o quadcopter não conseguiu voar de uma forma sustentada devido à presença de alguns problemas em alguns materiais (ESC e motor). Mais detalhes sobre os resultados obtidos no voo do quadcopter são dados no capítulo 6.

5.3 Integração do Quadcopter na Plataforma de Simulação

Nesta fase é descrito todo o desenvolvimento feito para integração do quadcopter na plataforma de simulação. Numa primeira instância, são explorados os agentes do tipo UAV, a forma como são representados na plataforma de simulação, assim como a comunicação que os mesmos podem estabelecer uns com os outros. De seguida, é dado destaque ao fluxo de informação, ou seja, como a informação flui desde o quadcopter real até à plataforma de simulação, e como o mesmo é alcançado através do material utilizado (software e hardware) acompanhado pelo seu processo de montagem. Será também explicado como os dados reais são "transformados" em dados virtuais de modo a que o quadcopter virtual possa interpretá-los e utilizá-los. No seguimento do ponto anterior, são abordadas as mensagens no formato *STANAG*, utilizadas no envio de comandos da plataforma para o quadcopter real. Por fim, são abordados os modos que a plataforma de simulação dispõe para testar os movimentos, a receção de informação externa, assim como envio de comandos para o quadcopter real.

5.3.1 Agentes UAV

A plataforma de simulação permite simular vários tipos de veículos, tais como aeronaves, carros, barcos e submarinos. Com vista a podermos integrar o quadcopter real na plataforma, dotou-se a mesma de um novo tipo de agente, o agente UAV.
Este agente é bastante semelhante ao agente aeronave sendo que a diferença reside no facto de o agente UAV poder receber dados externos e interpretá-los e enviar comandos para o UAV real.
O agente UAV é criado com base em dois parâmetros: uma *string* que representa o modelo a adotar no motor de jogo da plataforma de simulação e uma estrutura de dados própria do simconnect, o *SIMCONNECT_DATA_INITPOSITION*, que permite inicializar um agente UAV numa determinada posição tendo em conta métricas como latitude, longitude, altitude, orientação, velocidade entre outras.

5.3.1.1 Representação do Agente UAV na Plataforma de Simulação

Relativamente à forma de representar um agente UAV, a ideia passa por incluir um modelo *FSX* que permitisse representar, de uma forma aumentada, um quadcopter na plataforma. Existem várias empresas fornecedoras de modelos FSX, tais como: flyawaysimulation[2] e simviation[3]. Contudo, não se encontraram nestes modelos FSX compatíveis com este tipo de veículo. Devido a esta barreira, tomou-se a decisão de representar o agente UAV através de um modelo de helicóptero visto ser o veículo com morfologia e mecânicas mais semelhantes às de um UAV.
O modelo escolhido para representar um agente UAV foi o helicóptero *AgustaWestland EH101*[Força Aérea Portuguesa, 2018]. A Fig. 5.8 representa o modelo utilizado.

[2]Fonte: https://flyawaysimulation.com
[3]Fonte: https://simviation.com

Figura 5.8: Representação de um agente UAV na plataforma de simulação

Apesar da utilização de helicópteros para a representação de UAVs importa perceber que a utilização destes veículos não é tão linear, isto porque o *FSX* apresenta limitações quanto à inteligência artificial dos helicópteros. Para a criação de agentes UAV no ambiente simulado é utilizada uma função disponibilizada pelo Simconnect: *AICreateSimulatedObject*. Todavia esta função traz um grande inconveniente que é o de não possuir uma vista no objeto. Este aspeto será detalhado mais à frente. Outro aspecto que importa realçar é que o espaço onde o UAV virtual se desloca não é o mesmo do UAV real.

5.3.1.2 Comunicação entre Agentes UAVs virtuais

Com a introdução de agentes UAV na plataforma de simulação, desenvolveu-se uma funcionalidade que permite oferecer aos agentes UAV virtuais uma forma de poderem comunicar entre si, através do *AgentService* e partilhar informações como por exemplo o paradeiro de cada um. Esta funcionalidade, uma vez ativada, é utilizada por cada UAV virtual sendo que a informação é enviada constantemente.
Foi criada uma estrutura de dados denominada por *QuadMessage* que permite aos agentes UAV a troca de dados entre si. Esta estrutura é composta por quatro variáveis, sendo estas: *type* (texto para outros agentes UAV), latitude (informação sobre a sua latitude), longitude (informação sobre a sua longitude) e altitude (informação sobre a sua altitude).
As variáveis latitude, longitude e altitude são do tipo *double* enquanto que o *type* é do tipo *string*. Trata-se, por isso, de uma forma de os UAVs internos da plataforma (virtuais) poderem comunicar entre eles.

5.3.2 Fluxo da informação

Para além dos agentes UAV, outro aspeto bastante importante a reter neste projeto centra-se em perceber como a informação se "movimenta" do quadcopter para a plataforma de simulação e vice-versa.

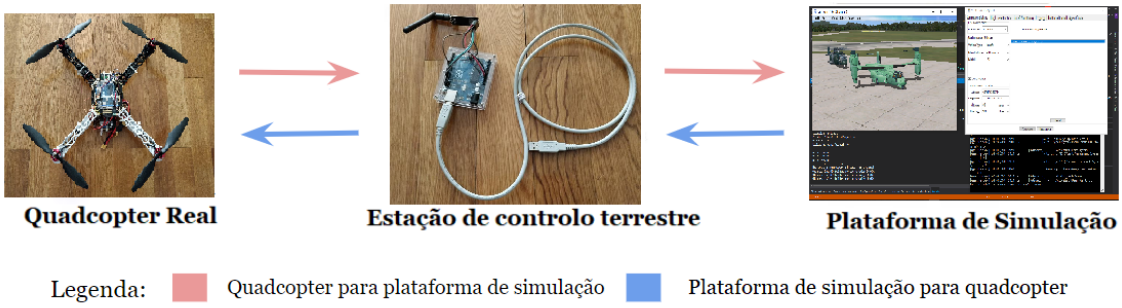


Figura 5.9: Fluxo que a informação segue na plataforma de simulação e quadcopter

Como é possível observar na Fig. 5.9, a informação segue dois sentidos: do quadcopter para a plataforma e desta para o quadcopter. Os dois sentidos distinguem-se pelo tipo de informação transmitida e pela função que cada um desempenha no sistema global.
No sentido do quadcopter para a plataforma de simulação, a informação engloba latitude e longitude (recolhidas pelo sensor GPS NEO 6M), altitude relativamente ao solo (coletada pelo sensor ultrasónico HC-SR04) e orientação (reunida pelo sensor MPU6050). A informação é coletada no quadcopter real, nomeadamente nos sensores referidos, e é enviada para a estação de controle através de rádios NRF24. Uma vez recolhida a informação do quadcopter, esta é então enviada para a plataforma de simulação através da porta de série onde é tratada.
No sentido da plataforma de simulação para o quadcopter real, a informação é essencialmente composta por texto que representa comandos a serem executados pelo quadcopter (explicado na secção 5.3.7). É transmitida por porta de série para a estação de controle e depois desta para o quadcopter por rádio NR24. Outro aspeto importante é a perda de mensagens entre o quad e a plataforma de simulação. Sempre que o quadcopter deixa de enviar de enviar mensagens, seja por falta de bateria ou por outro motivo, a plataforma de simulação espera um curto periodo de tempo e depois interrompe a simulação caso o envio de mensagens não seja retomado.

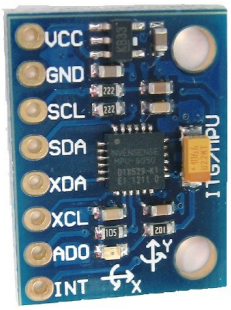
5.3.3 Material

Tal como no voo do quadcopter, é importante analisar que material estará envolvido nesta fase. Na secção 4.2 foi apresentado todo o material que esteve envolvido neste projeto, todavia, aqui apenas são destacados aqueles que têm uma maior preponderância para a integração do quadcopter na plataforma de simulação.

Para além destes materiais, utilizaram-se outros já enumerados na secção 5.2.1 tais como: o arduino UNO, quatro ESCS, quatro motores e a bateria. Estes componentes são os únicos que são comuns no voo e na integração.

(a) Arduino MEGA2560 [Arduino, 2018a]

(b) Sensor ultrasónico HC SR04 [Components101, 2017]



(c) Sensor MPU6050 [Arduino, 2018d]

(d) Sensor GPS NEO 6-M [Arduino, 2018c]

(e) Rádio NRF24 com antena PA [Elecrow, 2018]

Figura 5.10: Materiais para a integração no sistema

Mais informações sobre as características e como utilizar estes materiais são dados na secção 4.1 do manual (Anexo A).

5.3.4 Montagem do Quadcopter e da Estação de Controlo Terrestre

Esta fase constituiu-se como sendo mais exigente que a montagem para o voo do quadcopter e apresenta-se como sendo a etapa com maior complexidade e preponderância neste projeto não só por envolver vários componentes mas também por existirem várias condicionantes que os mesmos devem respeitar para que funcionem no modo pretendido. É descrito como os componentes devem

ser agrupados uns com os outros e que ligações devem ser feitas no quadcopter e na estação de controlo.

5.3.4.1 Montagem do Quadcopter

Antes de abordarmos a montagem do quadcopter com os sensores importa esclarecer que todos os componentes que tinham sido instalados para o voo do quadcopter foram desconectados, sendo que alguns foram mesmo retirados. Esta decisão será explicada no decorrer do seguinte guia de montagem.
Realizaram-se as seguintes etapas:

- Depois de desmontados os elementos do quadcopter com exceção do frame, dos ESCs e do sensor MPU6050, incorpora-se o arduino MEGA2560 (Fig. 5.10a) no nível superior do quadcopter;

- Colocar o rádio NRF24 (Fig. 5.10e) no nível superior do quadcopter. Este componente possui vários sinais, sinais esses que correspondem a pinos do componente. Os sinais envolvem o VCC (*power*), GND (*ground*), CE, CSN (este e o anterior controlada a transmissão dos dados), SCK (*Serial Clock*), MOSI (*Master Out Slave In*), MISO (*Master In Slave Out*) e IRQ (não será utilizado), sinais estes que devem ser ligados a portas com determinadas propriedades. A ligação deste componente ao arduino MEGA2560 é demonstrada de seguida;

- Colocar o sensor ultrasónico HC-SR04 (Fig. 5.10b), numa posição que dê para obter a distância a que o quadcopter se encontra do chão e prender o mesmo de forma a que o sensor permaneça seguro. Este sensor possui vários sinais tais como o GND, VCC, Trigger e o Echo. A ligação deste componente ao arduino MEGA2560 é demonstrada de seguida;

- Colocar o GPS NEO 6-M (Fig. 5.10d) na parte superior do quadcopter, em qualquer local desde que não interfira com outros sensores e deve estar preso para oferecer leituras mais precisas. Este possui vários sinais tais como o VCC, GND, RX e o TX. SCL. A ligação deste componente ao arduino MEGA2560 é demonstrada de seguida;

- Ligar os ESCs ao arduino MEGA2560. A ligação deste componente ao arduino MEGA2560 é demonstrada de seguida;

Grande parte dos componentes (GPS, sonar e MPU6050) utiliza uma fonte de alimentação VCC de 5V. Contudo, não é possível ligar os três VCCs dos três componentes ao arduino MEGA2560 uma vez que esta placa controladora apenas oferece uma porta 5V.
Face a este problema pensou-se em utilizar uma placa de ensaio ou *breadbord*. Este material permite fazer a partilha de portas sem comprometer o sistema. Outra solução possível seria soldar o VCC dos três componentes num só no entanto, esta solução não seria viável caso, no futuro, fosse necessário a alteração de componentes. Assim sendo, a melhor opção para a limitação de entradas VCC de 5V no arduino MEGA2560 foi a utilização de uma placa de ensaio.

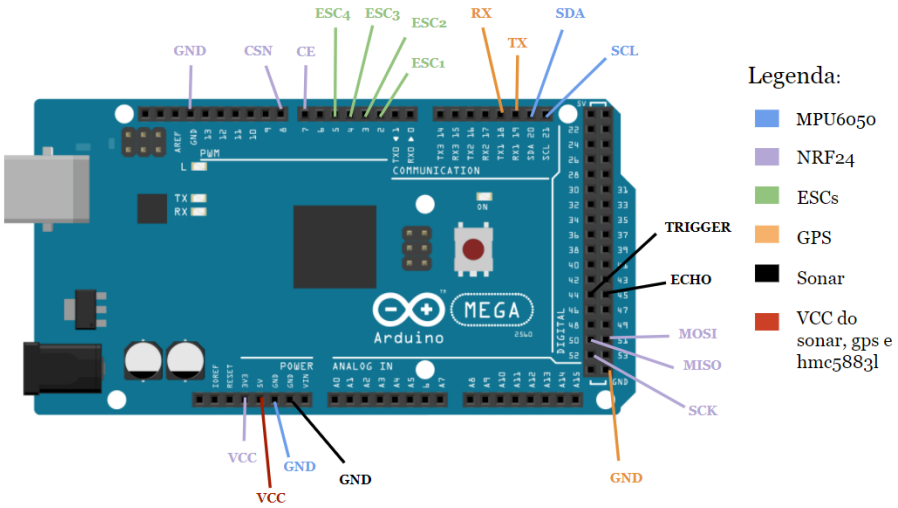


Figura 5.11: Ligação dos componentes ao arduino MEGA2560

Ligaram-se os componentes NRF24, sensor ultrasónico HC-SR04, sensor MPU6050, GPS e ESCs ao arduino MEGA2560 com as ligações dispostas na Fig. 5.11.
É importante que sejam seguidos os passos mencionados anteriormente, caso contrário não será possível fazer uma recolha eficiente dos dados dos sensores instalados no quadcopter, e posteriormente, observar a sua influência na plataforma de simulação. A Fig. 5.12 representa o quadcopter com os sensores instalados após a montagem. A introdução de outros sensores pode afetar o sistema por poder afetar o quadcopter em termos energéticos e por o arduino não ter capacidade para lidar com tantos sensores.

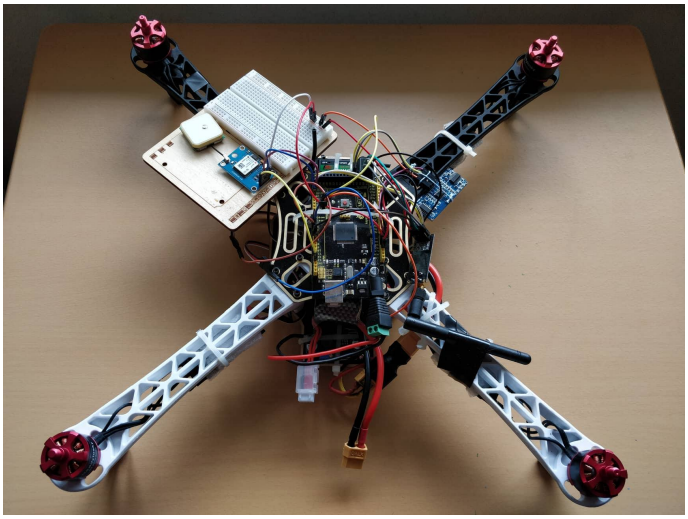


Figura 5.12: Hardware do Quadcopter preparado para a integração

Todo este processo de montagem encontra-se mais detalhado na secção 6.2 do Manual (Anexo A).

5.3.4.2 Montagem da Estação de Controle Terrestre

O processo de montagem da estação de controle terrestre é um processo simples e rápido de se realizar e que é independente do quadcopter montado anteriormente.
Envolve apenas a colocação do rádio NRF24 (Fig. 5.10e) no arduino UNO. Tal como na integração deste tipo de rádio no quadcopter, é importante voltar a perceber onde os sinais CE, CSN, SCK, MOSI e MISO devem ser colocados.

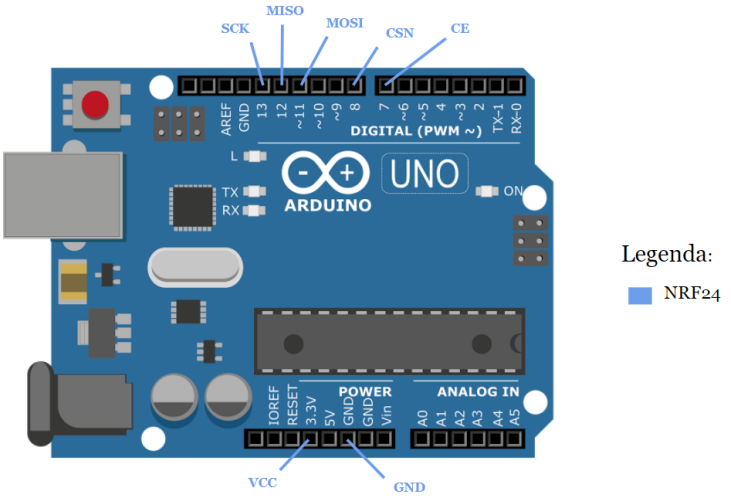


Figura 5.13: Ligação dos componentes ao arduino UNO (estação de controlo)

Como se verifica na Fig. 5.13, trata-se de um componente simples e que implica apenas a introdução de radios NRF24. A Fig. 5.14 representa a estação de controlo depois da montagem.

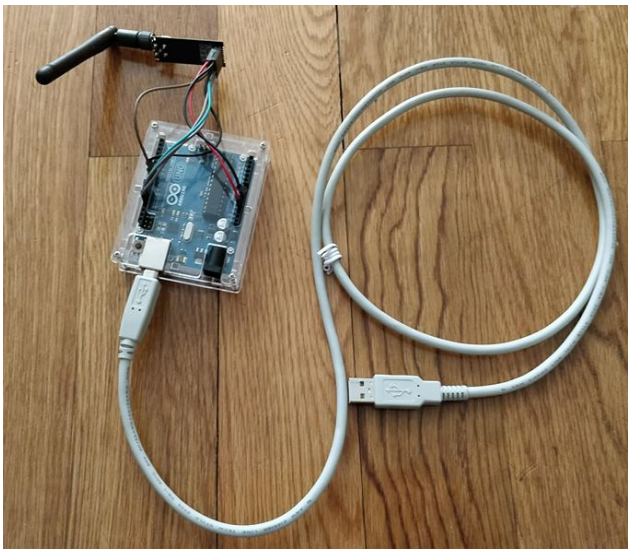


Figura 5.14: Hardware da estação de controlo terrestre

Todo este processo de montagem encontra-se mais detalhado na secção 6.4 do Manual (Anexo A).

5.3.5 Configuração do Software para a Integração do Quadcopter na Plataforma de Simulação

Uma vez terminada a montagem do quadcopter e da estação de controlo terrestre, importa esclarecer que software foi incluído nas placas controladoras MEGA2560 e UNO.
Nesta secção serão explicadas as principais funcionalidades do software utilizado, desenvolvido pelo autor deste projeto tendo como base de aprendizagem o *site* do Arduino[4] e outras fontes que estão sinalizadas nesta dissertação.

5.3.5.1 Software utilizado no quadcopter

O quadcopter foi montado para ser capaz de ler os dados do GPS, do sonar, do sensor MPU6050, enviar pulsos dos ESCs para os motores e ler comandos da plataforma de simulação. Em complemento com a montagem destes componentes no quadcopter foi desenvolvido software para cumprir com estas necessidades.
Primeiramente começou-se por estudar as bibliotecas necessárias para se aceder às funcionalidades de elementos como os rádios, GPS, sonar, MPU6050 e ESCs. Teve-se a preocupação de procurar bibliotecas otimizadas de forma a que a comunicação entre os elementos fosse a mais rápida e eficaz possível.
Bibliotecas utilizadas:

- *SPI*[Arduino, 2014b] e *RF24*[Arduino, 2015] - Bibliotecas responsáveis pela comunicação entre o quadcopter e a estação de controlo terrestre. A biblioteca *SPI* utiliza um protocolo de comunicação destinado para múltiplos dispositivos a curtas distâncias e a velocidade elevada (depende dos dispositivos destino). Geralmente há um dispositivo *master* que inicia a comunicação e controla a taxa de transferência de dados. Depois existem os *slaves* que recebem os dados e respondem ao *master*. A biblioteca *RF24* permite utilizar funções inerentes aos rádios NRF24;

- *NewPing*[Arduino, 2012] - permite que o sonar funcione em complemento com o arduino MEGA2560 com grande precisão e rapidez. Esta biblioteca concede a possibilidade de o sonar emitir e receber sinais de forma a saber a que distância se encontra um objeto;

- *Servo*[Arduino, 2014a] - concede a possibilidade de movimentar os motores do quadcopter através de pulsos *PMW*, enviados pelos ESCs. Esta biblioteca permite que a placa controladora controle servomotores (máquinas que recebem sinais de controlo) de uma forma simples. Os sinais recebidos pelos servomotores permitem que os mesmos girem numa posição específica;

- *NMEAGPS*[5] e *GPSport*[5] - permite que o GPS recolha informações acerca da posição do quadcopter (latitude, longitude, altitude, número de satélites, precisão, deslocamentos, entre outros). Inicialmente pensou-se em recorrer à biblioteca *TinyGPS* que também permite

[4]Fonte: https://www.arduino.cc/
[5]Fonte: https://github.com/SlashDevin/NeoGPS

recolher dados relativos à posição, todavia, a recolha é mais demorada, o que acaba por atrasar a comunicação entre dispositivos. Este comportamento deve-se ao facto de a biblioteca *TinyGPS* fazer o tratamento de toda a mensagem *NMEA* (tipo de mensagem fornecida pelo satélite) por isso optou-se por utilizar a biblioteca *NMEAGPS* e *GPSport*. Esta biblioteca faz "lock" ao satélite e faz o tratamento de uma parte da mensagem, obtend assim, apenas os dados necessários sem comprometer a comunicação;

- *Wire*[Arduino, 2014c] - concede a possibilidade de se obter a orientação do quadcopter. Trata-se de uma biblioteca bastante simples e prática que permite obter vários atributos com as informações do giroscópio nos eixos x,y e z. Também fornece dados sobre a aceleração em x,y e z.

Uma vez estudadas as bibliotecas, elaborou-se um ficheiro, Quadcopter.ino que conciliou os elementos rádio, GPS, sonar, sensor MPU6050 e ESCs no arduino MEGA2560.
Inicialmente foram incluídas as bibliotecas enunciadas, criou-se um objeto do tipo *NMEAGPS*, um objeto do tipo sonar com uma distância máxima de 2m com as portas declaradas na montagem do quadcopter, e um outro objeto do tipo rádio com as portas mencionadas na montagem do UAV; Desenhou-se duas estruturas de dados que permitem guardar a informação dos sensores (latitude, longitude, altitude e orientação) e armazenar os comandos enviados pela plataforma; De seguida, fez-se a configuração do rádio, nomeadamente em que pipes a informação deve ser lida e escrita. Esta configuração permite o intercâmbio de informação entre o quadcopter e estação de controlo sem misturar dados. Também se armaram (termo bastante utilizado na gíria dos UAVs e simboliza o momento em que os ESCs estão prontos a funcionar) os ESCs de modo a que os mesmos consigam girar quando são recebidos sinais de controlo da plataforma de simulação.

5.3.5.2 Software utilizado na estação de controlo terrestre

A estação de controlo terrestre foi montada para estabelecer uma ponte entre o quadcopter e a plataforma de simulação. É capaz de receber dados e enviá-los para a plataforma de simulação e receber comandos e enviá-los para o quadcopter, tendo sido desenvolvido software para cumprir com estas necessidades.
Em termos de bibliotecas utilizadas, apenas se utilizaram as bibliotecas *SPI* e *RF24*, já explicadas acima. Criou-se o ficheiro EstacaoDeControlo.ino que utiliza apenas as propriedades do rádio NRF24.
Inicialmente criou-se um objeto do tipo rádio com as portas declaradas na montagem da estação de controlo terrestre e repetiu-se a criação das estruturas de dados *myData* e *myCommand*. Tal como na configuração do quadcopter, procedeu-se à configuração do rádio, nomeadamente em que pipes a informação deve ser lida e escrita.
Concluída a configuração, os dados provenientes do quadcopter são enviados para a plataforma de simulação através da porta de série. Quando a estação de controlo recebe os comandos da plataforma, envia-os ao quadcopter para executar.

5.3.6 Transformação dos dados do Quadcopter Real para o Quadcopter Virtual da Plataforma de Simulação

Outro aspeto a ter em conta é a forma como a plataforma de simulação trata os dados provenientes da estação de controlo terrestre para depois poder transpô-los para o quadcopter virtual.
A plataforma está equipada com duas *threads*, uma que extrai a informação dos dados provenientes da estação de controlo para quatro listas, cada uma relativa aos atributos latitude, longitude, altitude, orientação e outra que permite que o quadcopter virtual assuma uma nova posição/disposição com base nos dados presentes nessas quatro listas. Poderia ter-se criado apenas uma lista de um tipo de objeto com os quatro atributos em vez de ter quatro listas, no entanto, o autor deste trabalho preferiu a escolha das quatro listas, visto ser uma solução mais percetível apesar de pouco otimizado. Tratam-se de *threads* separadas para reduzir possíveis atrasos entre a obtenção e envio dos dados do quadcopter real e a sua manipulação por parte do quadcopter virtual.
Observa-se assim que se estabeleceu um padrão produtor-consumidor[Hughes, 2013]. Este padrão descreve dois processos, o produtor e o consumidor, que compartilham um *buffer* de tamanho fixo usado como uma fila. A função do produtor é gerar dados, colocá-los no *buffer* e começar de novo. Ao mesmo tempo, o consumidor "consome" os dados, um pedaço de cada vez. A Fig. 5.15 identifica o padrão seguido.

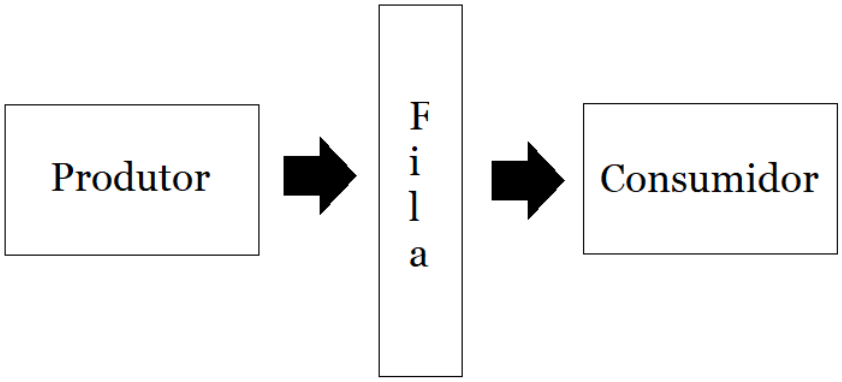


Figura 5.15: Padrão Produtor-Consumidor

Neste caso, a primeira *thread* (trata os dados enviados pela estação de controlo) é o produtor, enquanto que a segunda (uso dos dados presentes na lista no quadcopter virtual) é o consumidor.
Neste processo, a plataforma de simulação recebe mensagens com o seguinte formato: latitude | longitude | altitude | heading. Seria também possível obter estes dados através de comandos *STANAG*, nomeadamente através do comando *Inertial State* que permite fornecer dados sobre a sua posição e outros atributos no entanto, optou-se por criar uma nova estrutura de forma a simplicar e otimizar a transmissão dos dados do quadcopter para a plataforma.
Exemplo de mensagem utilizada: 41.224511 | -8.621419 | 30 | 120
Depois de receber a mensagem, esta é dividida com base nas divisórias presentes na mensagem,

sendo depois cada pedaço adicionado à lista respetiva. Paralelamente a este processo, os atributos latitude, longitude, altitude e orientação do quadcopter virtual são alterados com base nas listas que estão em constante preenchimento.
Quando o quadcopter virtual recebe dados novos (alteração em qualquer dos quatro atributos) a sua posição/disposição é atualizada através de uma função do *SimConnect*, *SetDataOnSimObject*. No entanto, apenas a utilização desta função para ajudar na rotação e deslocamento do quadcopter virtual não é suficiente, isto porque não garante um movimento fluido.
Neste sentido, para deslocar o quadcopter virtual para outra posição e assim suavizar o seu movimento, efetuaram-se algumas a alterações na sua translação. Sempre que o quadcopter virtual deteta alterações, seja na altitude, orientação, latitude ou longitude, o mesmo calcula a diferença entre a próxima posição e a posição atual e depois divide essa diferença em várias novas posições a assumir pelo UAV virtual. Esta divisão ajudou na criação do movimento fluido no veículo.

5.3.7 Eventos *STANAG* no Quadcopter

Para enviar comandos da plataforma de simulação para o quadcopter real, utilizou-se a 3ªedição do *STANAG 4586*[North Atlantic Treaty Organization, 2015], publicada em 2012.
O STANAG 4586 é um documento que descreve um padrão concebido pelos membros da NATO com a missão de definir as interfaces que devem ser implementadas de modo a atingir o nível de interoperabilidade requerido entre, por exemplo, UAVs e estação de controlo. Este define arquiteturas, interfaces, protocolos de comunicação, elementos de dados a trocar entre os veículos e a estação de controlo, assim como o formato das mensagens a trocar. Neste projeto damos foco ao formato e à estrutura que as mensagens devem possuir.

5.3.7.1 Estrutura das Mensagens com Formato *STANAG*

As mensagens que seguem o formato *STANAG 4586* são constituídas por conjuntos de blocos com a seguinte estrutura:
Como se observa na Fig. 5.16, cada mensagem possui como elementos principais um número de sequência, comprimento da mensagem, ID de origem, ID de destino, tipo de mensagem, propriedades da mensagem, dados e *checksum* opcional. Os restantes são parâmetros opcionais e não foram utilizados para este projeto.
O número de sequência permite segmentar os dados de uma mensagem em sequências de blocos de tamanho igual. O comprimento da mensagem diz respeito ao tamanho que a informação de uma mensagem deve possuir. O ID de origem ocupa 4 bytes e diz respeito ao componente que envia a mensagem, estando dividido em três partes: o código do país (8 bits), o código do componente (3 bits) e os restantes 21 bits estão reservados para o uso escolhido pelo país. O ID de destino é semelhante ao ID de origem com a diferença de que é relativo ao componente que recebe a mensagem. O tipo de mensagem ocupa 2 bytes e identifica o tipo de mensagem que está a ser enviada de modo a poder ser tratada de forma adequada quando rececionada pelos componentes. As propriedades da mensagem abrangem 2 bytes e dividem-se em quatro partes: *acknowledgment*

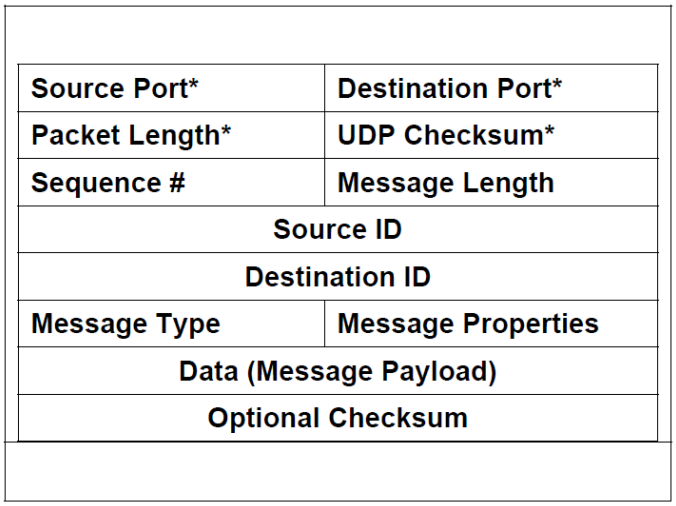


Figura 5.16: Estrutura de uma mensagem com formato STANAG 4586 [North Atlantic Treaty Organization, 2015]

(1 bit), versão de *IDD* (7 bits), comprimento *checksum* (2 bits) e utilização futura (6 bits). Estas propriedades permitem constituir dois tipos de mensagens: *Push* e *Pull*. As mensagens *Pull* são mensagens que são transmitidas em resposta a um pedido enquanto que as mensagens *Push* são mensagens transmitidas periodicamente tendo em conta um dado evento. Esta divisão é feita para otimizar a utilização da largura de banda necessária. O campo *data* permite identificar vários campos relativos à informação a ser transmitida e constitui-se como uns dos parâmetros mais importantes das mensagens com formato *STANAG*.

5.3.7.2 Comandos Principais

Os comandos enviados pela plataforma de simulação para o quadcopter real destinam-se a instruir o mesmo a executar um determinado movimento. No presente projeto, foram tidos em conta três tipos de comandos:

- Comando de direção do veículo ou *Vehicle Steering* - indica ao quadcopter real que deve alterar a sua direção com base num valor dado pela plataforma de simulação. Este comando também permite alterar a altitude do quadcopter;
- Comando para nova posição ou *AV Position Waypoint* - informa o quadcopter para assumir uma nova posição através do deslocamento (em xy) dado pela plataforma de simulação;
- Comando para motores ou *Engine* - informa o quadcopter para ligar os motores quando pretendido pela plataforma.

Cada um destes comandos segue a estrutura demonstrada na Fig. 5.16. É explicado de seguida como ocupar cada um dos campos de cada um dos comandos.

O comando direção ou *Vehicle Steering* é identificado pela sequência #2002 (obtida no documento STANAG 4586) e tem a seguinte estrutura:

1. Comprimento da mensagem - assume um comprimento de 25 unidades que em hexadecimal é 0x19. Ocupando 2 bytes na mensagem, aparece “00 19”;

2. ID de Origem - o código do país é 20 (codigo de Portugal) que convertido para hexadecimal apresenta-se como 0x14. O código do componente que envia o comando é o código que diz respeito à plataforma de simulação, ou seja, “001”, que conjugando com os restantes 5 bits do segundo byte corresponde a 0x20 em hexadecimal. Para os restantes 16 bits, ocupou-se da seguinte forma 00000000 00000001. Esta escolha foi arbitral, sendo que convertido para hexadecimal corresponde a 0x01. Ocupando 4 bytes na mensagem, o ID de origem é “14 20 00 01”;

3. ID de Destino - o código do país é o mesmo que o anterior. O código do componente que recebe o comando (quadcopter real) é “011”, que conjugando com os restantes 5 bits do segundo byte corresponde ao valor hexadecimal 0x60. Os restantes bits tomam o mesmo valor que os restantes bits calculados anteriormente no ID de origem. Ocupando 4 bytes na mensagem, o ID de destino será então “14 60 00 01”;

4. Tipo de Mensagem - corresponde à sequência #2002 que em hexadecimal é 07 D2. Ocupando 2 bytes, o tipo será “07 D2”;

5. Propriedades da Mensagem - trata-se de uma mensagem do tipo *Push*. Uma vez que este tipo de mensagens não necessita de *acknowlegment* o primeiro bit assume o valor 0. A versão *IDD* utiliza o valor 30 (valor fixo) que em hexadecimal é 0x1E. O *checksum* e a utilização futura não são utilizados por não desempenharem nenhum papel de relevância no projeto, e por isso assumem o valor 0. Ocupando 2 bytes, as propriedades serão “1E 00”;

6. Conteúdo da mensagem ou *data* - parte da mensagem onde é necessário ter bastante atenção, especialmente neste caso em que o tipo de mensagem pode servir para alterar a direção ou a altitude do quadcopter real. Em ambos os casos é necessário ter em conta as propriedades do comando de sequência #2002, presentes no documento *STANAG 4586*. O vetor de presença (atributo da *data*) está presente no início de cada mensagem e permite discernir quais os campos da *data* a utilizar, assim como o seu tamanho. Este vetor é um campo mapeado bit a bit que permite identificar quais os campos presentes no mesmo. Quando um bit toma valor “0” significa que o respetivo campo não está presente enquanto que se estiver a “1” está presente.

   (a) Na pretensão de alterar a altitude, é necessário colocar os campos *timestamp* (ocupa o índice 1 na data e indica quando a mensagem foi criada), *altitude command type* (ocupa índice 2) e *commanded altitude* (ocupa índice 3) a 1 e os restantes a 0. Tendo em conta que o vetor de presença indica que a *data* é constituída por 3 bytes, a sequência *data*

tem a seguinte disposição “00000000 00000000 00000111” que em hexadecimal é “00 00 07”;

(b) Caso a intenção seja alterar a orientação do quadcopter real então é necessário colocar os campos *timestamp* (ocupa o índice 1 na data), *heading command type* (ocupa índice 5) e *commanded heading* (ocupa índice 7) a 1 e os restantes a 0. Tendo em conta que o vetor de presença indica que a *data* é constituída por 3 bytes, a sequência *data* tem a seguinte disposição “00000000 00000000 01010001” que em hexadecimal é “00 00 51”.

O comando que permite assumir nova posição ou *AV Position Waypoint* é identificado pela sequência #13002 (obtida no documento STANAG 4586) e tem a seguinte estrutura:

1. Comprimento da mensagem - tal como no comando anterior, assume um comprimento de 25 unidades que em hexadecimal é 0x19. Ocupando 2 bytes na mensagem, esta fica “00 19”;
2. ID de Origem - igual ao comando anterior;
3. ID de Destino - igual ao comando anterior;
4. Tipo de Mensagem - corresponde à sequência #13002 que em hexadecimal é 32 CA. Ocupando 2 bytes, o tipo será “32 CA”;
5. Propriedades da mensagem - igual ao comando anterior;
6. Conteúdo da mensagem ou *data* - para alterar a posição do quadcopter, é necessário colocar os campos *timestamp*, *waypoint number* (ocupa índice 2), *waypoint to latitude or relative Y* (ocupa índice 3) e *waypoint to longitude or relative X* (ocupa índice 4) a 1 e os restantes a 0. Tendo em conta que o vetor de presença indica que a *data* é constituída por 3 bytes, a sequência *data* tem a seguinte disposição “00000000 00000000 00001111” que em hexadecimal é “00 00 0F”.

Por último, o comando que permite ligar os motores ou *Engine* é identificado pela sequência #2008 (obtida no documento STANAG 4586) e tem a seguinte estrutura:

1. Comprimento da mensagem - assume um comprimento de 9 unidades que em hexadecimal é 0x9. Ocupando 2 bytes na mensagem, aparece “00 09”;
2. ID de Origem - igual ao comando direção ou *Vehicle Steering*;
3. ID de Destino - igual ao comando direção ou *Vehicle Steering*;
4. Tipo de mensagem - corresponde à sequência #2008 que em hexadecimal é 07 D8. Ocupando 2 bytes, o tipo será “07 D8”;
5. Propriedades da mensagem - igual ao comando direção ou *Vehicle Steering*;

6. Conteúdo da mensagem ou *data* - para se ligar os motores do quadcopter, é necessário colocar os campos *timestamp*, *engine number* (ocupa índice 2), *engine command* (ocupa índice 3), *ignition switch power* (ocupa índice 6) e *ignition switch activation* (ocupa índice 7) a 1 e os restantes a 0. Tendo em conta que o vetor de presença indica que a *data* é constituída por 1 byte, a sequência data tem a disposição 01100111" que em hexadecimal é "67";

5.3.7.3 Personalização dos Comandos Utilizados

Utilizaram-se os comandos anteriores para informar o quadcopter real sobre quando deve executar uma determinada manobra ou evento.
Exemplo de comando Vehicle Steering (em hexadecimal) quando enviado ao UAV real: 00 19|14 20 00 2|14 60 00 2|07 D2|1E 00|00 00 07.
Cada parcela corresponde ao tamanho de mensagem, ID de origem, ID de destino, tipo de mensagem, propriedades da mensagem e *data*, respetivamente.
No entanto, houve a necessidade de se efetuar uma personalização sobre os mesmos devido ao facto de alguns parâmetros não serem relevantes para o projeto. Outra razão para as alterações nos comandos utilizados centra-se na identificação das manobras a utilizar. Atualmente o *STANAG 4586* permite ao utilizador assumir uma nova posição/disposição sem especificamente dizer que tipo ou género de manobras deve executar. Por exemplo, quando se pretende utilizar o comando para alterar altitude, não é indicado ao quadcopter se deve subir ou descer. Apenas é dado um valor que representa a altitude que deve ser assumida pelo quadcopter e que o informa se deve subir ou descer, dependendo se a nova altitude é superior ou inferior à atual. Trata-se de um comportamento natural na aviação que se deve ao facto de o quadcopter possuir mecanismos que o tornam *self-aware* (a utilização de mensagens *STANAG* do tipo *Inertial State* permitem ao quadcopter saber a que altura está, entre outras propriedades). Contudo para este projeto, precisa de ser mais detalhado. Melhorou-se este aspeto, adicionando um novo parâmetro que permite identificar exatamente o tipo de manobra. Assim sendo, tornou-se a estrutura dos comandos mais condensada e simples, composta apenas pela informação essencial. Esta inclui: o tipo da mensagem, o conteúdo da mensagem ou *data* e parâmetro extra para a identificação da manobra (*CMD_UP*, *CMD_DOWN*, *CMD_FORW*, *CMD_BACK*, *CMD_ROT* e *ENGINE*).

Exemplos de mensagens personalizadas enviadas ao quadcopter real:

- Subir - usa o comando de direção ou *Vehicle Steering* e tem a seguinte disposição "2002|7|CMD_UP-altitude";
- Descer - usa o comando de direção ou *Vehicle Steering* e tem a seguinte disposição "2002|7|CMD_DOWN-altitude";
- Andar em frente - usa o comando nova posição ou *AV Position Waypoint* e tem a seguinte disposição "13002|15|CMD_FORW-deslocamento";

- Andar para trás - usa o comando nova posição ou *AV Position Waypoint* e tem a seguinte disposição “13002|15|CMD_BACK-deslocamento”;
- Rodar - usa o comando de direção ou *Vehicle Steering* e tem a seguinte disposição "2002|51|CMD_ROT-rotação”;
- Motores - usa o comando de ligar os motores ou *Engine* e tem a seguinte disposição "2008|67|Engine-rotaçãomotores”. A rotação dos motores assume um valor fixo de 350, valor mínimo que permite a rotação dos quatro motores do quadcopter.

Os atributos altitude, deslocamento, rotação e rotação motores das mensagens anteriores são definidos na platforma de simulação.

5.3.8 Modos da plataforma de simulação

A plataforma de simulação possui três modos direcionados para UAVs. Os modos são distintos uns dos outros e permitem testar diferentes funcionalidades do sistema. Estes são:

- Movimentos do quadcopter;
- Sensores e envio de comandos;
- Motores.

5.3.8.1 Modo Movimentos Quadcopter

Modo desenvolvido inicialmente para observar o desempenho e os diferentes movimentos que o quadcopter virtual pode executar na plataforma de simulação. Ao contrário dos restantes modos, este modo não faz uso dos dados fornecidos pelos sensores do quadcopter real e por isso não necessita de ligação à estação de controle terrestre nem ao quadcopter. Assume-se assim como um modo bastante simples, construído com o propósito de mostrar os movimentos disponíveis e ajudar os restantes modos a instruir que tipos de movimentos o quadcopter deve desempenhar com base em fontes de dados externas (sensores utilizados).

Movimentos que o quadcopter virtual pode executar:

- *Vertical Take Off* (2);
- *Vertical Land* (3);
- *Move Forward* (4);
- *Move Backwards* (5);
- *Rotate* (6).

A numeração permite identificar os movimentos na figura seguinte. A numeração 1 corresponde à posição inicial.

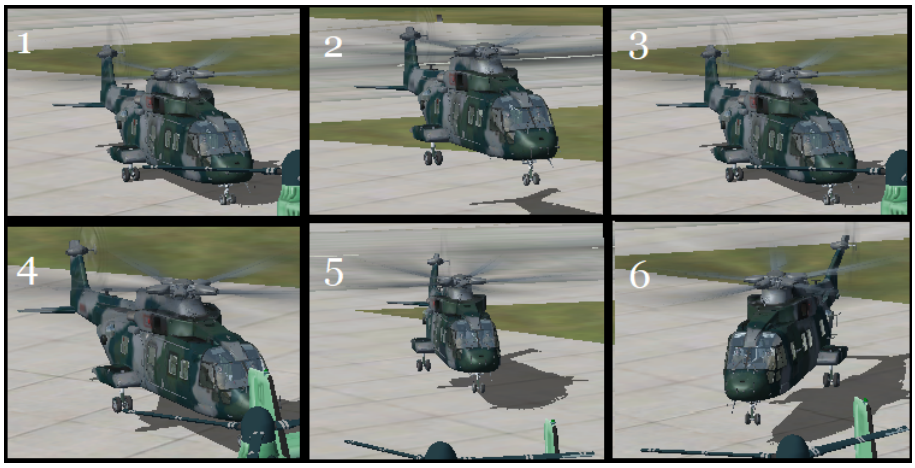


Figura 5.17: Movimentos possíveis do quadcopter

Na Fig. 5.17 também é apresentada uma hélice (canto da imagem a verde) que faz parte de uma aeronave, nomeadamente o Osprey V-22, de forma a termos uma câmara perto do quadcopter virtual. Poderia ter-se escolhido outro tipo de avião (mais pequeno), todavia procurou-se uniformizar o tipo de veículos utilizados. É verdade que o Osprey V-22 não é um quadcopter mas é das poucas aeronaves que mais se aproxima a estes veículos em termos de morfologia e mecânicas de voo. Não desempenha nenhum papel em específico na plataforma de simulação, apenas o de nos possibilitar a visualização do quadcopter virtual.

5.3.8.2 Modo Sensores e envio de comandos

Constitui-se como o modo com maior relevo, não só por permitir a utilização dos sensores, como também o envio de comandos ao UAV real. O recurso aos sensores permite que o quadcopter virtual se mova com base em dados reais extraídos pelos mesmos. Tal como explicado na secção do fluxo de informação, os dados, nomeadamente a latitude, longitude, altitude e orientação, são obtidos no GPS, sonar e MPU6050, respetivamente, sendo depois transferidos para a plataforma de simulação através da estação de controlo terrestre. Uma vez rececionados os dados, estes são tratados e transformados pela plataforma de simulação (secção 5.3.6). Quando algum dos atributos varia, o quadcopter virtual move-se com base na variação sentida. Os movimentos que o quadcopter executa são os mencionados no modo anterior.
Para além da movimentação do quadcopter virtual na plataforma de simulação com base em dados externos, este modo também contempla a possibilidade do envio de comandos da plataforma para o quadcopter real. Estes comandos (secção 5.3.7) permitem informar o UAV real sobre que manobra devem executar.
No presente modo, o utilizador apenas tem influência nos comandos que envia para o quadcopter realizar, ou seja, o utilizador não interfere com os movimentos do quadcopter virtual uma vez que

estes são provocados por elementos externos. Toda a informação que a plataforma recebe por parte da estação de controle terrestre é tratada independentemente do que o utilizador faça.
Antes da execução deste modo há a necessidade de se enviar para as placas controladoras do quadcopter real e da estação de controlo terrestre os ficheiros que correspondem ao modo dos sensores.
Outro aspeto a ter em conta é que é possível enviar todo o tipo de comandos, com a exceção do comando *ENGINE* que será abordado no modo seguinte.

5.3.8.3 Modo Motores Quadcopter

Depois de desenvolvido um modo que permitisse utilizar todos os sensores, criou-se um modo onde fosse possível conciliar a extração de dados externos e a sua transposição para a plataforma de simulação com o facto de o UAV real poder voar. No fundo, este modo tem como objetivo controlar o voo do quadcopter através de comandos enviados pela plataforma de simulação. Foi criado a possibilidade de se ligar os motores através do envio do comando *Engine* por parte da plataforma de simulação. A existência dos problemas relatados na secção 5.2.3 impediram o controlo total do voo do quadcopter todavia, espera-se que no futuro seja possível aplicar os vários movimentos no voo do mesmo (os movimentos seriam os relatados no modo anterior) através da adaptação do software para controlo do voo na placa controladora MEGA2560.

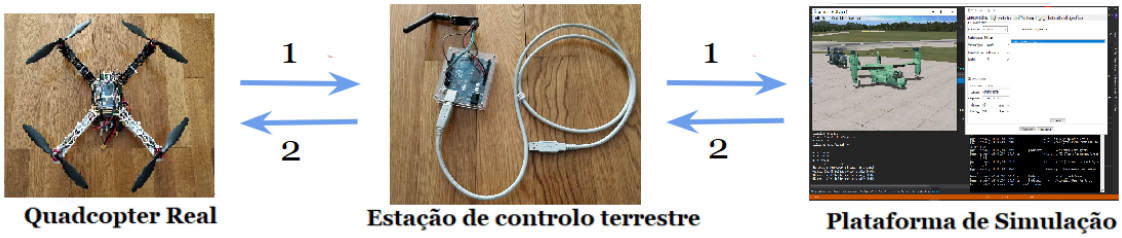


Figura 5.18: Dados (1) e Mensagens (2) dos Modos Sensores e Motores

A Fig. 5.18 permite identificar em que sentido são recolhidos os dados e enviadas as mensagens. Em termos de dados (1), enquanto que no modo anterior se utilizou os dados do sonar, GPS e MPU6050, neste apenas se recorreu à altitude fornecida pelo sonar. Esta escolha deveu-se ao facto de os dados provenientes dos sensores GPS e giroscópio já terem sido utilizados no modo anterior. Não havendo a necessidade de voltar a utilizar os mesmos componentes, utilizou-se apenas o sonar que permitiu ao quadcopter virtual executar movimentos como o *Vertical Take Off* ou *Vertical Land* no ambiente de simulação. Em termos de mensagens (2), este modo contempla a possibilidade de envio de mensagens do tipo V*ertical Take Off, Vertical Land* e *Engine* enquanto que no anterior é possível enviar todo o tipo de mensagens (*Vertical Take Off, Vertical Land, Move Forward, Move Backwards* e *Rotate*), com exceção do comando/mensagem *ENGINE*.
Também se distingue dos restantes modos por fazer uso dos ESCs e dos motores do quadcopter real. Apenas é permitido ao utilizador enviar comandos do tipo ENGINE. Quando este é enviado,

os ESCs enviam pulsos que permitem fazer com que os motores do quadcopter real façam uma rotação constante.
Tal como no modo anterior, é necessário enviar para as placas controladoras do quadcopter real e da estação de controlo terrestre os ficheiros que correspondem ao modo dos motores.
Os três modos habilitam a plataforma de simulação de testar diferentes componentes do sistema, seja a receção e tratamento de dados, como o envio e execução de comandos por parte do quadcopter real e movimentos do quadcopter.
É de extrema importância o carregamento dos ficheiros com os respetivos modos, nos arduinos UNO e MEGA2560 . Para se utilizar o modo dos movimentos do quadcopter não é necessário carregar qualquer ficheiro nas placas controladoras do quadcopter real e da estação de controlo terrestre uma vez que este modo não faz uso de dados externos.

5.3.9 Painel de Controlo para UAV

A plataforma de simulação possui vários painéis que permitem configurar uma série de parâmetros para a realização de voos e de missões. Estes encontram-se devidamente explicados e detalhados em [Silva, 2011].
Para este projeto, para além dos painéis já existentes, desenvolveu-se um novo denominado por *UAV Configuration*. O autor seguiu a disposição presente na Fig. 5.19.

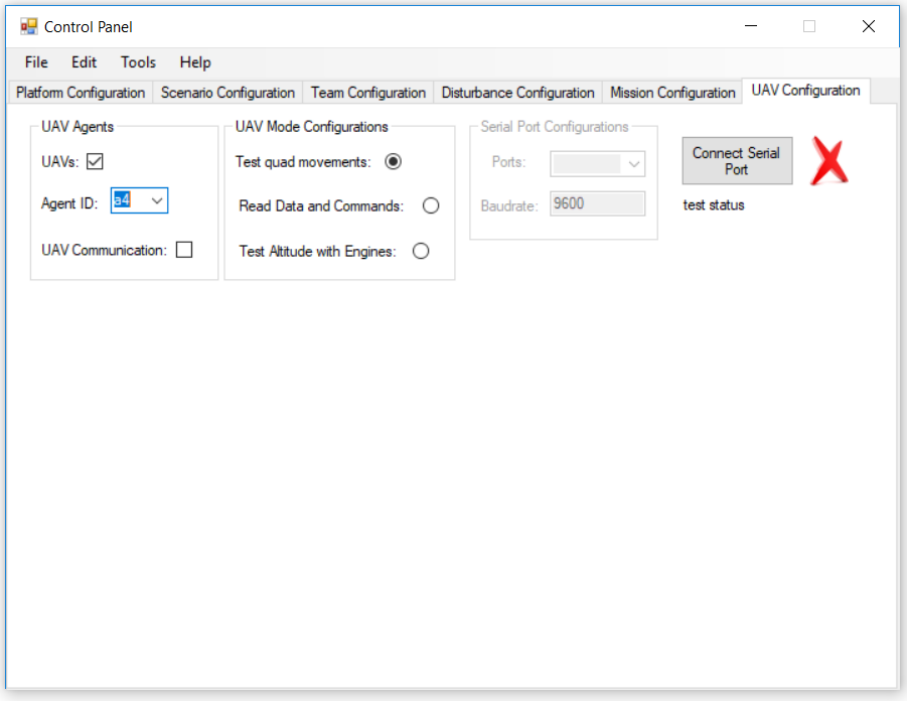


Figura 5.19: Painel para a configuração do UAV virtual na plataforma de simulação

Este painel permite testar várias funcionalidades da plataforma de simulação, relativamente aos UAVs. O bloco *UAV Agents* do painel é um dos mais importantes pois, para além de conceder

a possibilidade de trabalhar com UAVs no sistema, permite escolher qual o agente (*Agent ID*) que funcionará como *digital twin*, ou seja, selecionar o agente que será o quadcopter virtual e que poderá, consoante o modo escolhido, experimentar sensores, motores ou movimentos. Neste momento, apenas é possível ter no sistema um digital twin contudo, é possuir expandir o número de digital twins através de algumas alterações no sistema (tratamento da porta de acesso e sincronismo entre as threads dos UAVs virtuais). Inclui-se uma pequena *feature* que permite que os vários UAVs virtuais comuniquem uns com os outros através do *AgentService*. Esta comunicação foi explicada e abordada na secção 5.3.1.2.

No bloco *UAV Mode Configurations* é escolhido um dos três modos já abordados na secção anterior de modo a configurar o modo sobre o qual o sistema avaliará o desempenho dos UAVs.

Por fim, é mostrado o bloco *Serial Port Configurations* que permite selecionar a porta sobre a qual serão enviados comandos e receber informação do quadcopter real. É possível testar a conexão à porta selecionada. Para se saber qual a porta responsável pela receção de dados externos e envio de comandos convém utilizar a ferramenta arduino IDE.

Capítulo 6

Resultados

Neste capítulo são detalhados todos os testes realizados e resultados obtidos durante o desenvolvimento do presente trabalho. São abordados os testes realizados em dois níveis: voo do quadcopter e integração do mesmo na plataforma de simulação. Procurou-se também referir alguns aspetos externos que podem afetar o desempenho do sistema.

6.1 Testes ao voo do Quadcopter

Na presente secção serão discutidos os testes efetuados a vários componentes que permitem realizar o voo do quadcopter. Para além disso, serão debatidas algumas condicionantes que podem influenciar negativamente o voo do mesmo, acompanhadas por algumas soluções que podem ajudar a suavizar ou anular alguns comportamentos.
Vários testes foram realizados no âmbito do voo do quadcopter:

- *Inputs* do rádio transmissor *FlySky FS-i6*;
- ESCs e sentido dos motores;
- Giroscópio;

6.1.0.1 Testes ao rádio transmissor *FlySky FS-i6*

Nesta fase procurou-se verificar se os comandos enviados pelo rádio transmissor *FlySky FS-i6* eram captados pelo recetor *FlySky FS-IA6* do quadcopter, e se de facto estes comandos correspondiam aos movimentos certos, ou seja, *thrust*, *yaw*, *roll* e *pitch*. Para esta verificação, recorreu-se ao software criado por Joop Brokking [Brooking, 2015].
Antes de testar o rádio transmissor, houve a necessidade de se carregar no arduino UNO do quadcopter, o ficheiro **"YMFC-3D_receiver.ino"**. Existe uma outra alternativa para testar os comandos enviados pelo rádio transmissor através do carregamento do ficheiro **"YMFC-AL_esc_calibrate.ino"**.

Tanto a utilização de um como de outro permite testar a ligação do rádio transmissor com o recetor de sinal do quadcopter; contudo, a diferença entre os dois reside na forma como se procede o teste do mesmo. Enquanto que no **"YMFC-3D_receiver.ino"** basta testar os comandos com o rádio transmissor e com monitor de série do arduino IDE, com o **"YMFC-AL_esc_calibrate.ino"** é necessário enviar o caracter "r" no monitor de série do arduino IDE para se aceder ao painel de testes dos sinais enviados pelo rádio transmissor.

Figura 6.1: Estado inicial do rádio transmissor *FlySky FS-i6*

Quando se inicia o teste para verificar se o recetor *FlySky FS-IA6* capta os sinais, o rádio transmissor apresenta a disposição da Fig. 6.1, ou seja, com os dois manípulos centrados (objetos que estão rodeados com um círculo branco).
Os sinais testados foram o *thrust*, *yaw*, *pitch* e *roll*. Todos estes sinais apresentam valores entre os 1000 e 2000 (valores escolhidos pelo autor do software e que estão relacionados com o alcance do recetor e transmissor rádio) e influenciam a intensidade do sinal enviado.

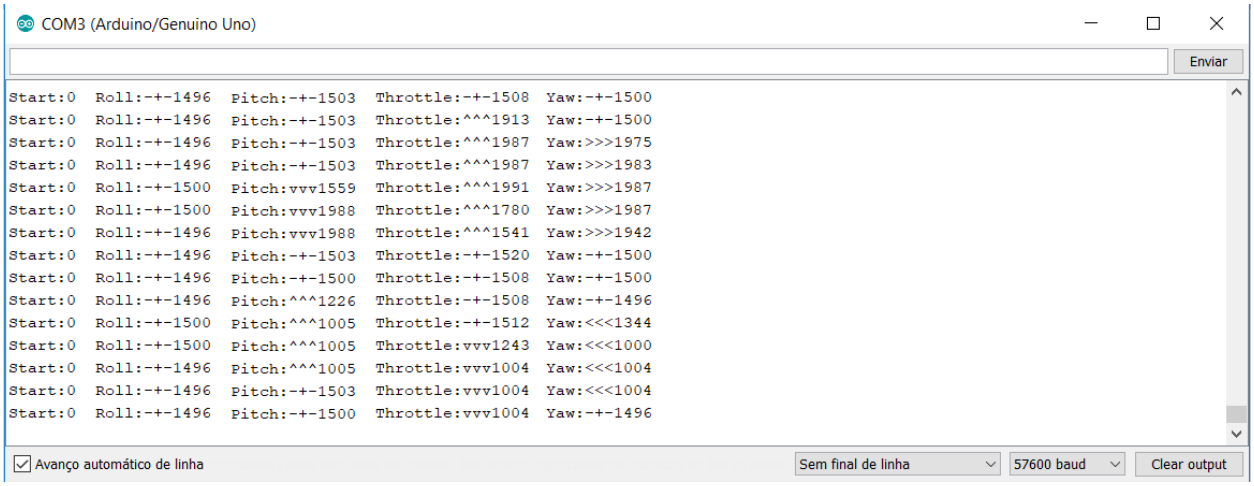


Figura 6.2: Monitor de série com os sinais rececionados pelo quadcopter, quando todos os sinais são alterados

Através do arduino IDE obtiveram-se os resultados da Fig. 6.2. Nesta, verificamos que o intervalo de valores que os sinais do rádio assumem é de facto entre os 1000 e 2000. Observa-se ainda que houve oscilações nos diferentes sinais quando aplicados diferentes movimentos em diferentes manípulos do rádio transmissor.
A tabela seguinte ajuda a perceber que manípulo corresponde a um determinado sinal e qual o seu significado, e como alcançar as diferentes variações nos diferentes sinais.

Sinal Rádio	**Manípulo**	**Sinal > 1500**	**Sinal +/-1500**	**Sinal < 1500**
Thrust	Esquerdo (1)	Manípulo para cima (aceleração)	Manípulo centrado	Manípulo para baixo (desaceleração)
Yaw	Esquerdo (1)	Manípulo para direita (rotação para a direita)	Manípulo centrado	Manípulo para esquerda (rotação para a esquerda)
Pitch	Direito (2)	Manípulo para baixo (subida)	Manípulo centrado	Manípulo para cima (descida)
Roll	Direito (2)	Manípulo para a direita (subir a asa esquerda)	Manípulo centrado	Manípulo para a esquerda (subir a asa direita)

Tabela 6.1: Como testar o rádio transmissor do quadcopter

Caso algum dos sinais referidos na Tabela 6.1 apresente valores fora do intervalo mencionado ou não sofra alteração quando era suposto, então recomenda-se repetir o processo.

6.1.1 Testes aos ESCs e sentido dos Motores

Nesta etapa, a intenção passa por testar se todos os ESCs funcionam corretamente e em sintonia. Para além disso, procurou-se verificar se o sentido de giro dos motores correspondia ao sentido de giro detalhado na secção 5.1.1 do Manual (Anexo A).
Para se recorrer a este género de testes, houve a necessidade de se carregar no arduino UNO do quadcopter o ficheiro **"YMFC-AL_esc_calibrate.ino"**. Uma vez enviado e calibrados os ESCs do quadcopter, é necessário enviar caracteres específicos de forma a poder testar se os ESCs, e por sua vez os motores, se encontram devidamente calibrados:

- "1" - rotação do motor 1;
- "2" - rotação do motor 2;
- "3" - rotação do motor 3;
- "4" - rotação do motor 4;

- “5” - rotação de todos os motores.

A verificação do sentido de giro de cada motor é feito individualmente através dos caracteres mencionados anteriormente. Para se avaliar se a rotação se encontra correta recomenda-se que se coloque, em vez das hélices, um pedaço de fita cola. Este processo não só ajuda a perceber o sentido de giro do motor como previne que ocorram acidentes quando os testes são feitos com as hélices.
Caso os ESCs não estejam sincronizados, recomenda-se a repetição do processo. Se os motores apresentarem sentido de giro incorreto ao pretendido é aconselhável trocar um dos três fios que servem de ligação entre o ESC e o motor. Esta troca pode ser feita a qualquer um dos fios.
Depois de carregado o ficheiro acima e escolhido o motor a testar, é avaliado o desempenho deste tendo em conta a sua vibração.

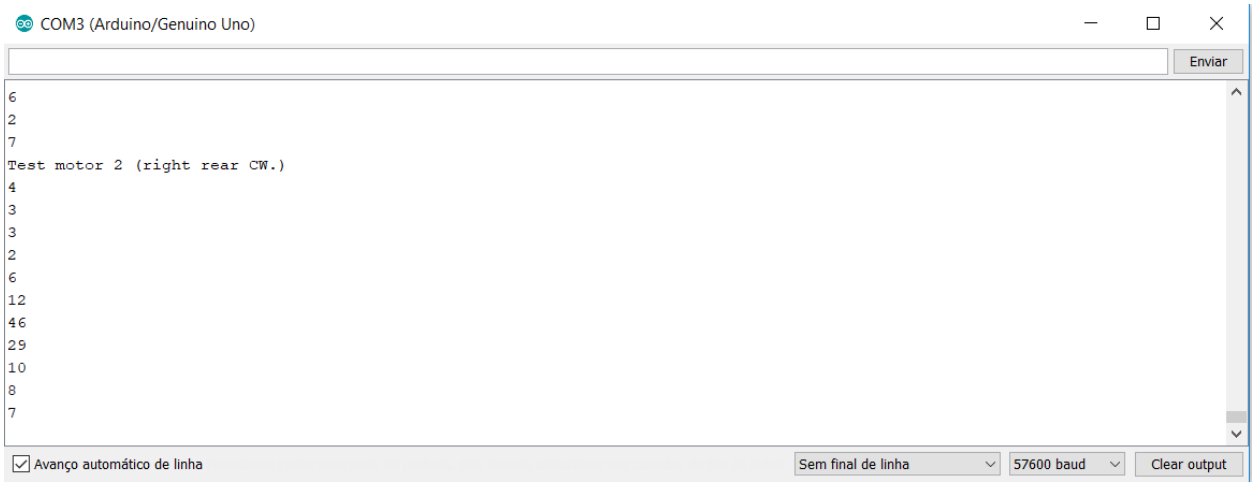


Figura 6.3: Monitor de série quando enviado o caractere “2”

Os valores presentes no monitor de série da Fig. 6.3, representam a vibração dos motores. Esta vibração é variável e depende se o motor está bem colocado, do material que o compõe e se foi montado corretamente (ausência de defeitos). Depois de vários testes, observamos que quanto maior for a aceleração, maior é a vibração e vice-versa, ou seja, a aceleração e a vibração são proporcionais uma com a outra. Mais uma vez a vibração depende do motor (não existem dois motores iguais), quando há aceleração os valores no monitor de série aumentam enquanto que quando há uma desaceleração os valores baixam. Quando os motores estão desligados não há qualquer vibração.

6.1.2 Testes ao Giroscópio

Nesta fase, procura-se verificar se o giroscópio, ou seja, o MPU6050 se encontra bem calibrado. É imperativo que esteja o melhor calibrado possível. visto ser este componente que indica ao quadcopter quando compensar e executar um determinado movimento.
Para se recorrer a este género de testes houve necessidade de se carregar mais uma vez no arduino

UNO do quadcopter o ficheiro “YMFC-AL_esc_calibrate.ino”. Uma vez enviado e calibrado o giroscópio do quadcopter, é necessário enviar o caracter “a” no monitor de série do arduino IDE para se aceder ao painel de teste ao giroscópio.

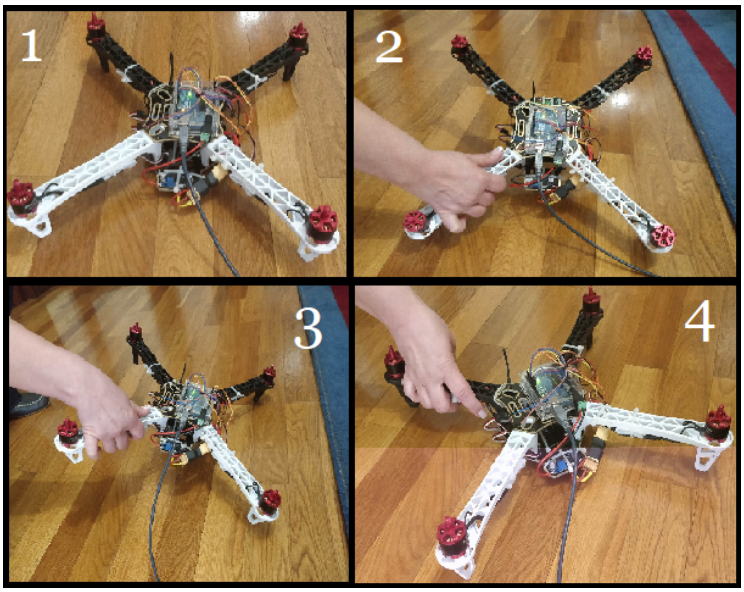


Figura 6.4: Testes ao giroscópio

A Fig. 6.4 representa os sucessivos testes que foram executados tendo em vista a calibragem do giroscópio. A imagem 1 da Fig. 6.4 representa a posição inicial, a 2 o *pitch*, a 3 o *roll* e a 4 o *yaw*.

É também apresentado o estado do giroscópio através do monitor de série do arduino IDE. Este tem em conta o sinais *pitch*, *roll* e *yaw*. Embora o ideal seja obter valores muito próximos de 0, isso é todavia complicado, visto, em grande parte dos casos, haver alguma inclinação e superfícies irregulares.

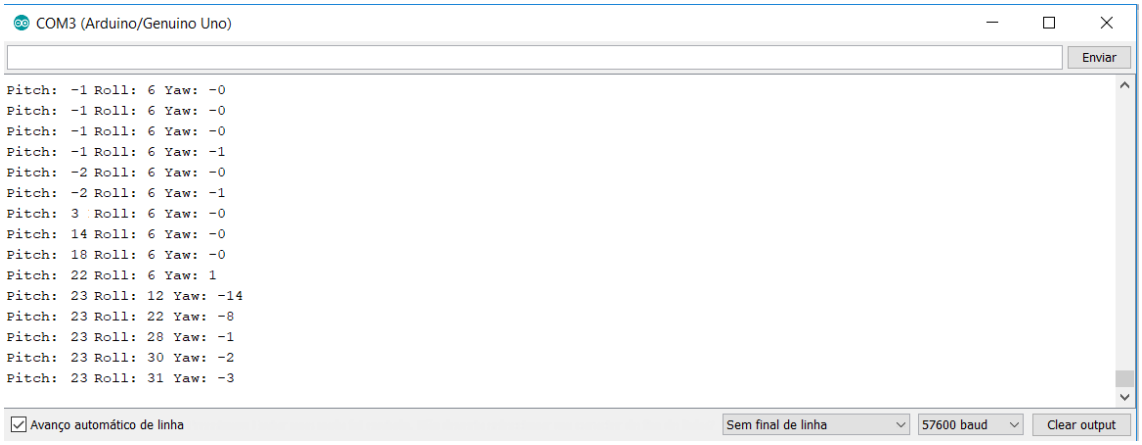


Figura 6.5: Monitor de série com os sinais do giroscópio quando se encontra calibrado

Através do arduino IDE obtiveram-se os resultados da Fig. 6.5. Nesta figura, verificou-se que o intervalo de valores assumidos pelos os sinais do rádio oscila bastante e que depende do movimento aplicado no quadcopter. As variações demonstradas nas três componentes da Fig. 6.5 devem-se à aplicação dos movimentos da Fig. 6.4, ou seja, as alterações nos valores *pitch*, *roll* e *yaw* causadas por 2, 3 e 4 respetivamente.

Sinal Rádio	**Sinal > 0**	**Sinal +/-0**	**Sinal < 0**
Pitch	levantar a frente	posição inicial	baixar a frente
Roll	levantar asa esquerda	posição inicial	levantar asa direita
Yaw	rodar à esquerda	posição inicial	rodar à direita

Tabela 6.2: Como testar o giroscópio do quadcopter

Caso algum dos sinais na Tabela 6.2 permaneça inalterado após aplicados os *pitch*, *roll* e *yaw*, então recomenda-se repetir o processo pois significa que o giroscópio ou está mal calibrado ou com problemas. Foram feitos vários testes e constatou-se que os sinais apresentaram resultados consistentes. No teste do *pitch*, empurrou-se o manípulo direito ligeiramente para baixo apresentando valores na ordem dos 150 o que representa uma inclinação de aproximadamente 12° no eixo do yy. No teste do *roll* empurrou-se o manípulo direito ligeiramente para direita apresentando valores na ordem dos 150 o que representa uma inclinação de aproximadamente 12° no eixo dos xx. No caso do *yaw* empurrou-se o manípulo esquerdo ligeiramente para esquerda e constatou-se valores na ordem dos 150 o que representa uma inclinação de aproximadamente 12° no eixo dos zz. Todos os dados podem ser guardados num ficheiro *log* através das funcionalidades do Arduino IDE.

6.1.3 Resultado do voo do quadcopter

Depois de todo o processo de recolha de informação e de montagem obteve-se o quadcopter presente na Fig. 5.7. Este quadcopter esteve sujeito aos vários testes mencionados anteriormente com vista a obter um voo equilibrado e sustentado. De seguida serão mostradas algumas imagens que representam diferentes planos de voo.

Figura 6.6: Estado inicial do quadcopter quando estabilizado

Como se observa na Fig. 6.6 colocou-se o quadcopter num plano o mais nivelado possível (a placa ajudou nesse aspeto), de modo a ajudar a descolagem. A manta tem como propósito impedir que alguma hélice sofra danos caso o quadcopter vire ao contrário. Para além disso, permite evitar que pedras entrem nos motores e danifiquem os mesmos. A placa serve essencialmente para ajudar a nivelar a superfície. Devido à enorme sensibilidade do giroscópio, sem uma superfície plana, o quadcopter não consegue levantar adequadamente.

Figura 6.7: Estado intermédio do quadcopter depois da descolagem

Depois de ligados os motores e calibrado o giroscópio, efetuou-se o *thrust*, ou seja, acelerou-se de modo a permitir que o quadcopter suba. Como é possível observar na Fig. 6.7, o quadcopter subiu e deslocou-se ligeiramente para a frente. Com o aumento da aceleração, é expectável que o mesmo continue a aumentar a sua altitude.

Figura 6.8: Estado final do quadcopter quando estabilizado

Como é possível observar na Fig. 6.8 o quadcopter conseguiu continuar a sua subida, tendo alcançado uma altitude de aproximadamente 50cm.
Depois de todos os testes realizados, alcançou-se o objetivo pretendido de fazer o quadcopter voar. Contudo, não se conseguiu estabelecer um voo sustentado (o voo foi curto e não possui *pitch*, *roll* e *yaw*). Depois de algum tempo no ar, os motores do quadcopter deixaram de responder, o que

levou à sua queda.
Na secção seguinte são abordados vários fatores que podem condicionar o voo do quadcopter e que se constituíram como possíveis causas para o sucedido.

6.1.4 Condicionantes no voo

Sendo composto por vários materiais e estando num ambiente onde existem múltiplos elementos externos que podem afetar o desempenho do voo do drone, é natural que, por vezes, seja difícil colocar o mesmo a voar na perfeição. São vários os fatores que podem impedir o voo sustentado, tais como:

- Giroscópio mal calibrado - umas das principais razões para a não descolagem do quadcopter. Deve-se sempre procurar calibrar o melhor possível com valores de *pitch*, *yaw* e *roll* iguais a 0. Nesta dissertação não se conseguiu obter valores de 0. Para contrabalançar este inconveniente, adicionou-se uma placa por baixo do quadcopter para nivelar a superfície e ajustou-se os valores iniciais dos sinais *pitch*, *roll* e *yaw* para aqueles que são obtidos inicialmente antes do voo. Depois desta alteração conseguiu-se descolar o quadcopter;

- ESCs mal calibrados ou defeituosos - Os ESCs devem permanecer sempre sincronizados caso contrário, o quadcopter não conseguirá manter-se no ar. Deve-se por isso repetir a calibragem dos ESCs. Outro motivo relativamente aos ESCs e que pode impedir o voo centra-se com o facto de estes possuírem defeitos. Este é um dos principais motivos por não se conseguir um voo sustentado neste projeto. Visto que os ESCs são fabricados em grandes quantidades existe sempre a probabilidade de existência de defeitos. No presente trabalho descobriu-se que existe um ESC defeituoso pelo facto de este provocar ruídos estranhos e de aquecer mais que os restantes. Procurou-se substituir o mesmo por um novo no entanto, a obtenção deste género de peças acarreta longos períodos de entrega (normalmente superiores a um mês) algo que atrasa o processo de desenvolvimento. Estes materiais são bastante sensíveis e propícios a estragarem-se com o tempo ou por má utilização;

- Sentido de giro dos motores - os motores devem ter o sentido de giro enunciado na secção 5.1.1 do Manual (Anexo A) para levantar voo, caso contrário, o quadcopter permanecerá no chão ou dará *flips*. No caso do sentido de giro estar diferente do planeado, deve-se trocar os fios que fazem a ligação entre os ESCs e motores (qualquer um serve);

- Colocação das hélices - existem hélices próprias para motores com sentido horário e sentido anti horário. Caso estejam colocadas incorretamente o quadcopter permanecerá no chão ou dará *flips*. Na secção 5.1.1 são demonstradas que hélices correspondem aos motores;

- Peso do quadcopter - O peso deve estar sempre bem distribuído pelo quadcopter e por esta razão no processo de montagem teve-se sempre em atenção o posicionamento do material. Caso o peso esteja mal distribuído, o quadcopter terá a tendência a cair para um dos lados;

- Condições atmosféricas - A presença de rajadas de, chuvas ou outros fatores podem dificultar o voo. Afeta as manobras de voo.

6.2 Testes à Integração do Quadcopter na Plataforma de simulação

Na presente secção, serão discutidos os testes efetuados aos vários componentes que permitem integrar o quadcopter na plataforma de simulação. Esta etapa sofreu um grande número de testes de forma a cumprir com o objetivo proposto.
Assim sendo, foram efetuados testes individuais e coletivos que permitem experimentar diferentes componentes no projeto. Os testes individuais estudam a influência individual de cada componente enquanto que os coletivos visam perceber a influência que um conjunto de componentes pode desempenhar sobre o sistema implementado.
Os testes individuais englobam a análise aos rádios NRF24, GPS e sonar. A MPU6050 não é testada aqui visto que já foi analisada na secção 6.1.2. Os testes coletivos procuram testar movimentos verticais do quadcopter virtual (inclui o sonar e os rádios), movimentos horizontais do quadcopter virtual (inclui o gps e os rádios), rotação do quadcopter virtual (inclui MPU6050 e rádios), todas as componentes anteriores (inclui o sonar, gps, MPU6050 e os rádios) e enviar comandos para o quadcopter real (inclui a possibilidade de ligar os motores do quadcopter real).

6.2.1 Testes Individuais

6.2.1.1 Rádios NRF24

Estiveram sujeitos a um grande número de testes, dado que desempenham um papel crucial na comunicação entre o quadcopter real e a estação de controlo terrestre. Os testes foram realizados tendo em conta dois parâmetros: presença de antena e distância.
Sem antena, a comunicação ficou limitada a uma distância máxima de 100 metros. Experimentou-se testar a comunicação entre os rádios a distâncias superiores a 100 metros e verificou-se que os mesmos não captavam qualquer sinal um do outro. Para além disso, observou-se que se os rádios estiverem a altitudes diferentes e próximos de redes WIFI, a comunicação sofre interferências.

Testes	Distância (m)	Tempo entre o envio e receção de dados (ms)
1	5	1767
2	20	1780
3	50	1784
4	> 100	Não capta sinal

Tabela 6.3: Testes aos rádios NRF24 sem antena tendo em conta o tempo e distância a que a comunicação é feita

A Tabela 6.3 demonstra os resultados obtidos nos testes aos rádios NRF24 sem antena. Foi analisado a distância de comunicação com o tempo entre o envio e receção dos dados.
Com a instalação de uma antena em cada um dos NRF24 a comunicação para além de melhorada ficou mais eficiente e sustentável. A introdução desta aumentou a distância de comunicação de 100 metros para 1000 metros e tornou a comunicação mais sustentável e mais rápida.

Testes	Distância (m)	Tempo entre o envio e receção de dados (ms)
1	1	1760
2	20	1784
3	50	1790
4	100	1804
5	500	1812
6	> 1000	Não capta sinal

Tabela 6.4: Testes aos rádios NRF24 com antena tendo em conta o tempo e distância a que a comunicação é feita

A Tabela 6.4 demonstra os resultados obtidos nos testes aos rádios NRF24 com antena. Foi analisado a distância de comunicação com o tempo entre o envio e receção dos dados. Difere dos testes sem antena devido ao facto de conseguir cobrir uma maior distância de comunicação.
Outro aspeto com relevo e que foi tido em conta foi a orientação em que é colocada a antena dos rádios NFR24. As antenas devem estar dispostas perpendicularmente uma com a outra, caso contrário não haverá comunicação ou existirá uma grande percentagem de perdas de pacotes de informação.

Testes	Orientação das antenas	Comunicação Rádio
1	Perpendicular	Capta Sinal
2	Paralelo	Não Capta Sinal

Tabela 6.5: Testes aos rádios NRF24 tendo em conta a orientação das antenas

Depois de analisados os resultados obtidos na Tabela 6.5, chegou-se à conclusão que seria melhor instalar-se a antena do quadcopter real numa posição vertical e colocar-se outra antena na estação de controlo terrestre numa posição horizontal. Esta colocação melhora o sinal de comunicação entre as duas antenas.
Descobriu-se também que a presença de obstáculos assim como de outras redes WIFI pode afetar o sinal chegando mesmo em alguns casos (bastantes obstáculos ou redes WIFI), a interromper o sinal por completo.

6.2.1.2 GPS

Componente com alguma importância no projeto e que permite ao quadcopter virtual deslocar-se horizontalmente na plataforma de simulação. Trata-se de um componente especial porque em alguns casos não é possível testar todas as suas funcionalidades (por exemplo a latitude e a longitude são atributos que nem sempre estão disponíveis). Assim sendo, estabeleceu-se dois cenários de teste: ambientes externos e internos.
Em ambientes internos é bastante complicado obter dados do GPS visto este não ter a capacidade de fazer *lock* (termo utilizado quando se pretende ligar um objeto a um satélite) a algum satélite. Sem o lock, não há dados do GPS e sem estes dados não há como verificar se há variação na latitude e longitude. Em ambientes externos, conseguiu-se fazer *lock* aos satélites e assim obter os dados latitude e longitude. No entanto, o tempo que demora a fazer *lock* varia consoante a posição em que o GPS se encontra. Existem outras condicionantes que podem afetar a eficácia do GPS, tais como o tempo decorrido desde a última utilização e o estado climatérico nublado (a presença de muitas nuvens afeta o *lock*). Foram feitos dez testes em cada um dos cenários com vista a demonstrar o impacto que o ambiente envolvente tem sobre o tempo de calibragem do GPS.

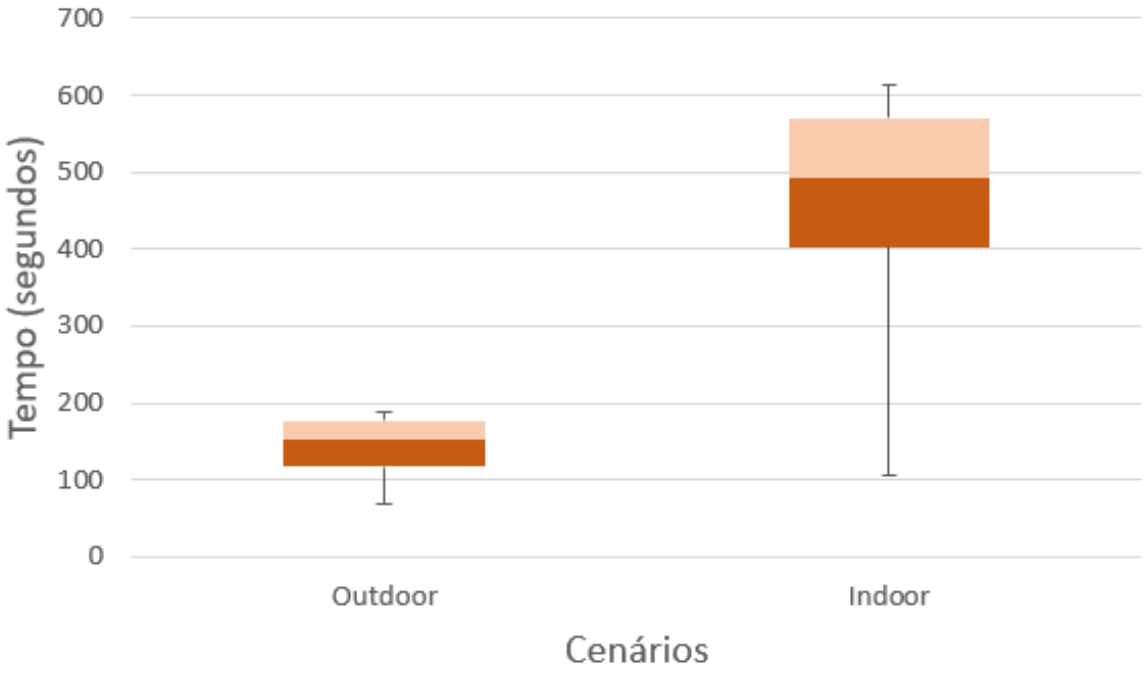


Figura 6.9: Testes feitos ao gps tendo em conta o meio que o rodeia

Como se observa na Fig. 6.9, os dois cenários influenciam bastante o tempo de calibragem do GPS. Verifica-se que em ambientes externos o tempo de calibragem varia entre os 80 e os 200 segundos enquanto que em ambientes internos estes valores sobem para os valores na ordem dos 400 e 560, ou seja, o tempo de calibragem em ambientes externos é inferior aos internos.
Tendo em conta os resultados obtidos, chegou-se à conclusão que se a intenção passa por utilizar o

GPS então a melhor opção passa por experimentá-lo em ambientes externos. Mesmo com o GPS instalado e em ambientes internos, a plataforma de simulação funciona normalmente apesar de não utilizar os dados latitude e longitude.

6.2.1.3 Sonar

Trata-se de um componente do quadcopter que fornece dados sobre a distância que este se encontra do chão. Estes dados permitem ao UAV virtual deslocar-se verticalmente no ambiente simulado da plataforma.
Fornece dados relativos à altitude do quadcopter real com grande precisão, apesar de só conseguir medir distâncias até 5 metros. Caso a distância para o chão seja superior a 5 metros ou menor que 3 cm, então os valores oferecidos pelo sonar apresentam o valor 0. Para se medir altitude superior a 5 metros poderia-se adquirir um GPS com maior potência de modo a recolher informação sobre a altitude (não se utilizou o GPS atual visto este não ser muito eficiente nesta função).
Fez-se dois tipos de testes para averiguar a eficácia do sonar utilizado. Estes foram realizados tendo em conta dois prismas: um em que se roda um plano para esquerda e outro que se roda para a direita. Em ambos, o objetivo passa por observar que distâncias o sonar recebe quando encontra superfícies com diferentes inclinações. A rotação à esquerda e à direita serve essencialmente para verificar se existe alguma diferença quando aplicadas diferentes rotações.
Em cada um dos testes foram feitas várias leituras às inclinações obtidas. Foram considerados três cenários: no primeiro cenário a distânica inicial foi de 20cm, no segundo foi de 30cm e no terceiro foi de 42cm. Os valores do eixo dos yy dizem respeito à distância para o objeto enquanto que os valores no eixo dos xx indicam a inclinação feita no objeto tendo em vista a verificação se a distância obtido pelo sonar é influenciada ou não.

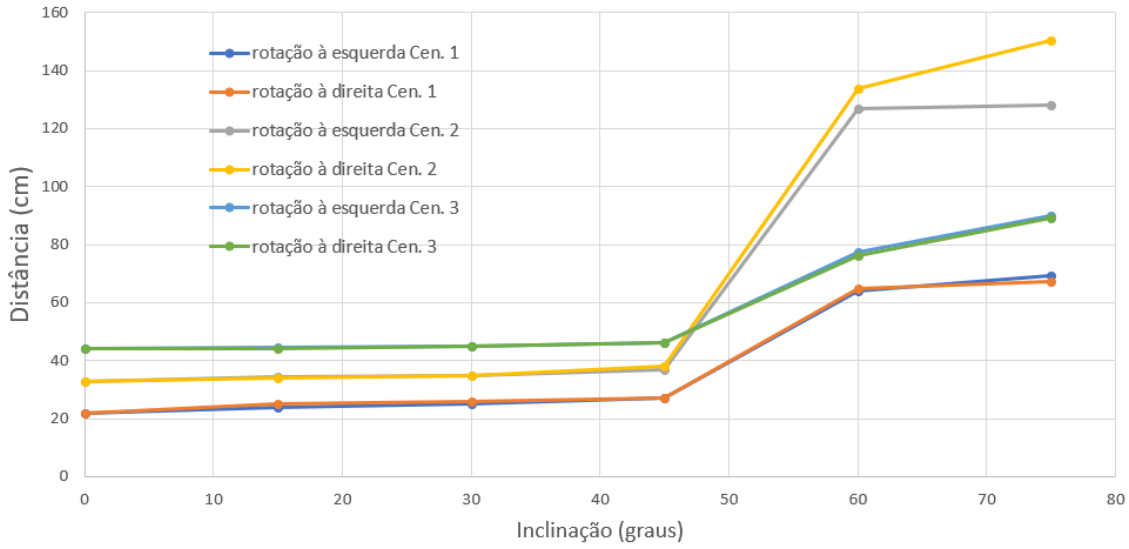


Figura 6.10: Testes feitos ao sonar tendo em conta a inclinação

Como se observa na Fig. 6.10, os cenários não apresentam grandes diferenças em termos de resultados o que permite inferir que se rodarmos a superfície para a esquerda ou direita, os resultados serão bastante semelhantes à exceção do cenário 2. Neste cenário, a partir dos 45°, verificou-se resultados piores quando comparados com o cenário 1 e 3. Este fenómeno deve-se à sensibilidade e limitações que o sonar apresenta quando perante inclinações elevadas.
Uma vez analisado o gráfico da Fig. 6.10 e como já foi constatado, verificou-se uma limitações no sonar. Em ambos os testes, o sonar conseguiu obter resultados precisos quando o plano tem uma inclinação máxima de 45 graus. Depois dessa inclinação, o sonar passa a receber dados muito pouco precisos. Este dado é relevante pois indica que o sonar utilizado não é apropriado para relevos bastante acidentados e com inclinações elevadas. Para superar este inconveniente, seria aconselhável seguir a solução enunciada anteriormente, ou seja, adquirir um GPS que permitisse medir altitudes.

6.2.2 Testes Coletivos

6.2.2.1 Movimentos verticais do quadcopter virtual

Este tipo de testes permite testar qual a influência que o movimento vertical do quadcopter real tem na altitude do quadcopter virtual na plataforma de simulação.
Para a realização deste, instalou-se um rádio e um sonar no quadcopter real e um outro rádio na estação de controlo terrestre. Uma vez preparados, ou seja, depois de carregados os ficheiros com o modo pretendido no quadcopter e na estação de controlo terrestre, procurou-se observar o comportamento do quadcopter virtual quando aplicadas alterações na altitude do quadcopter real.
Observou-se que tanto o rádio como o sonar funcionam sem qualquer problema. Os dados recebidos pela plataforma de simulação são tratados e depois aplicados no quadcopter virtual. Quando há a alteração dos dados recebidos o quadcopter virtual sobe ou desce consoante esses mesmos valores.
Procederam-se a dois tipos de testes (subir e descer) que permitiram analisar o movimento vertical do quadcopter virtual depois de aplicadas diferentes forças no quadcopter real. Em ambos o objetivo passa por analisar vários pontos, tais como: taxa de mensagens por segundo, intervalo colocado no movimento para aumento da sua fluidez (cada vez que são detetadas alterações na altitude, a diferença entre a nova e antiga altitudes são calculadas e depois divididas em várias posições para aumentar a fluidez) e o tempo entre fim do movimento real e fim do movimento virtual. Foram feitos seis testes que englobam subir e descer o quadcopter real.

Testes	Movimento	Taxa de mensagens/seg	Tempo entre fim do movimento real e fim do movimento virtual (ms)
1	Subir 20 cm	53	899
2	Subir 20 cm	54	1188
3	Subir 20 cm	54	1102
4	Descer 20 cm	54	887
5	Descer 20 cm	53	1008
6	Descer 20 cm	53	1198

Tabela 6.6: Medições feitas ao movimento virtual vertical

Como se observa na Tabela 6.6, os dois tipos testes não apresentam grande diferenças em termos de resultados. A única diferença de destaque é a diferença de tempo obtida entre o início do movimento real e movimento virtual que variaram entre os 890 e os 1200 microsegundos. Estas diferenças devem-se ao tempo colocado entre cada nova posição para o aumento da fluidez (15 microsegundos).

6.2.2.2 Movimentos horizontais do quadcopter virtual

Tem como foco averiguar qual o impacto que o movimento horizontal do quadcopter real tem no deslocamento horizontal do quadcopter virtual na plataforma de simulação.
Ao contrário da experiência anterior, substituiu-se o sonar pelo GPS e manteve-se os rádios no quadcopter real e na estação de controlo terrestre sendo que também foram carregados os ficheiros com o respetivo modo a testar.
Verificou-se que o rádio e GPS também funcionam corretamente sem causar qualquer interferência. A plataforma de simulação recebeu os dados e tratou-os, transpondo-os depois no quadcopter virtual. Tal como foi explicado na secção anterior, o GPS apenas fornece os dados em formato *NMEA* em ambientes externos, caso contrário a plataforma de simulação recebe o valor 0 para a latitude e longitude.
Depois de esperado pelo lock do GPS ao satélite, executou-se dois tipos de testes (andar em frente e para trás) que permitiram averiguar o movimento horizontal do quadcopter virtual depois de aplicadas diferentes forças no quadcopter real. Em ambos, o objetivo é o investigar vários prismas como a taxa de mensagens por segundo e o tempo entre a aplicação do movimento real para a movimentação virtual. Os atributos analisados foram estudados em ambiente externos visto apenas ser neste que se obtém os dados latitude e longitude. Também é possível testar esta componente em espaços internos contudo, não será possível ou será bastante demoroso o processo de calibragem do GPS. Foram feitos dois testes (não necessita de muitos testes visto tratar-se de um componente bastante simples em ambientes externos).

Testes	Movimento	Taxa de mensagens/seg	Tempo entre fim do movimento real e fim do movimento virtual (ms)
1	Frente 2 m	15	440
2	Atrás 2 m	15	440

Tabela 6.7: Medições feitas ao movimento virtual horizontal

Como se observa na Tabela 6.7, os dois testes apresentam resultados semelhantes mas com algumas ressalvas importantes. O facto de o tempo entre fim do movimento real e fim do movimento virtual ser tão reduzido (440 microsegundos) deve-se ao facto de não haver qualquer tempo para o aumento da fluidez entre os movimentos. Este dado é influenciado pelo problema enunciado na secção 5.3.1.1. Mesmo estando o quadcopter real parado, o GPS consegue detetar variações na latitude e longitude.
De realçar que a localização do quadcopter virtual no ambiente simulado não é a mesma do quadcopter real. O UAV virtual desloca-se com base no diferencial sentido nas movimentações reais.

6.2.2.3 Rotação do quadcopter virtual

Este teste visa estudar a influência que a rotação do quadcopter real tem na rotação do quadcopter virtual da plataforma de simulação. Para tal, retirou-se o GPS e colocou-se o sensor MPU6050 no seu lugar, mantendo os rádios no quadcopter real e na estação de controlo terrestre sendo que também foram carregados os ficheiros com o respetivo modo a testar tendo se observado que os sensores funcionam como pretendido.
Os dados recebidos pela plataforma de simulação são tratados e depois aplicados no quadcopter virtual quando há alteração da orientação do quadcopter real. O quadcopter virtual roda para a direita ou para a esquerda quando há variação dos valores oferecidos pela MPU6050.
Executaram-se vários testes que permitiram analisar a rotação do quadcopter virtual quando aplicadas diferentes rotações no quadcopter real. Tem como missão investigar a influência deste movimento em atributos como a taxa de mensagens por segundo, intervalo colocado no movimento para aumento da sua fluidez e tempo entre a aplicação do movimento real para a rotação virtual. Foram executados três testes onde se aplicou diferentes rotações.

Testes	Movimento	Taxa de mensagens/seg	Tempo entre fim do movimento real e fim do movimento virtual (ms)
1	Rotação 20º	53	814
2	Rotação 60º	53	900
3	Rotação 120º	54	1127

Tabela 6.8: Medições feitas à rotação do movimento virtual

Testes	Movimento	Taxa de mensagens/seg	Tempo entre fim do movimento real e fim do movimento virtual (ms)
1	Subir 20 cm; Andar para a frente 3 m e rodar 20º	13	2104
2	Descer 20 cm; Andar para trás 3 m e rodar 60º	14	2452
3	Subir 10 cm; Andar para trás 3 m e rodar 120º	13	2651

Tabela 6.9: Medições feitas ao movimento completo do UAV virtual

Como se observa na Tabela 6.8, os testes efetuados não apresentam grande diferenças em termos de resultados. A única diferença de destaque é o tempo obtido entre fim do movimento real e fim do movimento virtual. Os valores deste variaram entre os 814 e os 1127 microsegundos. Estas diferenças devem-se ao tempo colocado entre cada nova posição para o aumento da fluidez (16 microsegundos). Pode-se concluir que quanto maior a rotação, maior será a divisão do movimento para aumentar a fluidez. Quanto maior a fluidez maior é o tempo entre início do movimento real e fim do movimento virtual.

6.2.2.4 Todas as componentes anteriores

Esta experiência constitui-se como o teste de maior importância e tem com grande objetivo averiguar o comportamento do quadcopter virtual quando perante múltiplos dados com diferentes formatos. Estes dados envolvem informação providenciada pelo GPS, sonar e MPU6050.
Para a realização deste teste incluíram-se todos os elementos anteriores e observou-se que todos os componentes funcionam corretamente. Uma vez recebidos e tratados os dados pela plataforma de simulação, o quadcopter virtual desloca-se verticalmente e horizontalmente e roda quando há alteração dos respetivos atributos. Tal como no teste para os movimentos horizontais do quadcopter virtual, foi necessário ter-se em conta o ambiente em que este teste foi realizado.
Assim sendo efetuaram-se vários testes permitiram analisar o desempenho do quadcopter virtual depois de aplicadas forças verticais, horizontais e rotacionais no UAV real. Para tal, foram considerados os atributos: taxa de mensagens por segundo, intervalo colocado no movimento para aumento da sua fluidez e tempo entre a aplicação do movimento real para a rotação virtual. Os atributos analisados foram estudados em ambiente externos visto apenas ser neste que se obtém os dados latitude e longitude. Foram executados três testes com as componentes referidas e com diferentes tipos de movimentos. Os movimentos são aplicados sequencialmente.

Como se observa na Tabela 6.9, os testes efetuados não apresentam grande diferenças em termos de resultados assumindo valores entre os 2100 e os 2700 microsegundos. As diferenças de tempo entre fim do movimento real e fim do movimento virtual registadas devem-se ao facto de se aplicar uma rotação cada vez maior em cada um dos testes o que acarreta um maior tempo para o aumento da fluidez (15 microsegundos).

Modo	Leitura de dados (mensagens/seg)	Tempo entre o fim do movimento real e o fim do movimento virtual (seg)
Movimentos Quadcopter	Não faz uso nem recebe dados do quadcopter real	Inexistente
Sensores	14	1-3
Motores	54	0,5 - 1,5

Tabela 6.10: Modos presentes na plataforma de simulação

Assim sendo e tendo em conta os dados obtidos na Tabela 6.10, verificou-se que os diferentes modos possuem diferentes taxas de leitura de mensagens provenientes da estação de controlo e tempos para a sincronização entre o movimento do quadcopter real e movimento do quadcopter virtual.
Relativamente às taxas de leitura, o modo dos motores é o modo que apresenta uma maior taxa pelo facto de as mensagens que recebe da estação de controlo apenas possuírem informação relativa à altitude. O modo dos sensores apresenta uma taxa mais baixa visto que as mensagens que recebe contém mais detalhes (latitude, longitude, altitude e orientação) e necessitando assim de um maior tratamento. Outra razão para esta taxa deve-se à utilização do GPS. Este componente requer um maior tempo de espera para a receção de dados provenientes do satélite o que afeta como consequência a taxa de receção na plataforma de simulação. O modo dos movimentos não recebe nem trata dados provenientes do quadcopter real.
Quanto às diferenças temporais entre o aplicação de forças no quadcopter real e a sua visualização no quadcopter simulado, o modo sensores é aquele que mais tempo demora para a sincronização enquanto que o modo motores é que menos tempo demora. O modo movimentos como não utiliza dados externos não tem influência nestes estudo.
Estes resultados são perfeitamente normais nos UAVs. No modo sensores, o tempo obtido deve-se ao facto de o quadcopter virtual ter em atenção 4 atributos (latitude, longitude, altitude e orientação) enquanto que no modo motores apenas tem-se em consideração a altitude.

6.2.2.5 Enviar comandos para o quadcopter real

Este teste visa averiguar se o quadcopter real consegue receber os comandos mandatados pela plataforma de simulação. Os comandos experimentados encontram-se devidamente detalhados na secção 5.3.7.
Para a realização deste género de testes apenas é necessária a inclusão dos rádios no quadcopter real e na estação de controlo terrestre. Os sensores não desempenham qualquer papel de relevo neste ensaio. Neste caso, o utilizador escolhe qualquer um dos comandos disponíveis no painel *High Level Control*. Uma vez escolhido e enviado, o quadcopter recebe e executa o comando. É preciso ter em conta o modo em que se experimenta visto que existem comandos que apenas funcionam em determinados modos.
Assim sendo, experimentou-se enviar cada um dos comandos (cada um destes corresponde a um

movimento) ao quadcopter no modo sensores e esperou-se que o mesmo os recebesse. Observou-se que o quadcopter recebeu todos os comandos com um tempo de espera quase nulo.
Para além deste teste, experimentou-se um comando especial que permite ligar os motores. Foi necessário fazer um outro teste para este comando visto este apenas funcionar no modo dos motores. Sendo assim, experimentou-se enviar o comando *ENGINE* ao quadcopter real e esperou-se pela sua resposta. Verificou-se que após este envio, os motores do quadcopter real começaram a funcionar o que prova que o comando foi recebido com sucesso. O tempo de demora para a execução do comando foi quase nulo. Todavia também se observou que um dos motores e um dos ESCs não funcionava como pretendido. Por vezes, o ESC fez ruídos o que demonstrava que não estava inteiramente operacional e o motor não fazia a rotação com a mesma intensidade que os restantes motores o que também indiciou que estaria com alguns problemas. Neste momento, estes inconvenientes não têm grande impacto no modo dos motores, contudo, é aconselhável que no futuro se altere os componentes com deficiências.

Capítulo 7

Conclusões e Trabalho Futuro

Neste capítulo são abordadas todas as conclusões a que se chegaram com o presente trabalho assim como algumas melhorias a efetuar em trabalhos de futuro.

7.1 Conclusões

Com os constantes avanços tecnológicos que se têm sentido no presente século, a área da aviação autónoma, nomeadamente o setor dos UAVs, tem sentido uma maior procura por parte de várias entidades devido às inúmeras aplicações que possui. A possibilidade de se realizar missões onde o piloto é substituído, na tentativa de se salvar vidas, assim como por exemplo a automatização das entregas de encomendas, tornam o mercado destes veículos bastante apetecível. São bastantes as aplicações e vantagens que os UAVs podem oferecer à sociedade. Contudo, grande parte das soluções presentes no mercado são difíceis de gerir, sendo algumas bastante dispendiosas, não só na complexidade inerente a este género de veículos como em termos de custo monetário. Com base nestes inconvenientes torna-se fundamental o desenvolvimento de alternativas com custo mais baixo e mais simples, capazes de desempenhar várias funcionalidades básicas que são características dos UAVs como voar, entre outras.

De forma a resolver estes problemas montou-se um quadcopter e desenvolveram-se várias funcionalidades que permitiram a integração deste UAV numa plataforma de simulação pré-existente. O quadcopter foi construído com base em várias peças com o objetivo de se poder experimentar o voo do drone e de se poder extrair dados dos sensores previamente instalados. Foram tomadas várias medidas de maneira a cumprir com as regras impostas pela *NATO* e a obter resultados o mais realistas possíveis.

Em termos de voo do drone, procurou-se calibrar e testar o mais possível os componentes integrados no quadcopter tendo em vista um voo mais sustentável e prevenção a possíveis falhas que podem surgir e que foram enunciadas no decorrer deste trabalho. No que toca à interoperabilidade dos sensores e consequente integração do quadcopter na plataforma de simulação foram tomadas

providências na construção do quadcopter e da estação de controle terrestre para que o quadcopter virtual pudesse acompanhar os movimentos do quadcopter real sem qualquer problema e com o mínimo atraso possível. A plataforma de simulação foi melhorada através da implementação de um subsistema constituído por três modos, modos estes que permitem testar várias funcionalidades, tais como os movimentos que o quadcopter virtual pode executar no ambiente virtual, a leitura e transposição de dados externos para o quadcopter virtual de forma a que a disposição deste seja alterada com base nesses mesmos dados e, por fim, a ligação dos motores do quadcopter real quando sinalizado pela plataforma de simulação. De relembrar que os últimos dois modos permitem testar o envio de comandos por parte da plataforma, assim como a sua receção por parte do quadcopter real, com algumas condicionantes já referidas no presente trabalho.
Em termos de resultados, grande parte dos sensores utilizados apresentaram resultados satisfatórios, o que resultou no cumprimento de grande parte dos objetivos propostos. Por exemplo, o GPS demora uma média de 207 segundos a calibrar em ambientes abertos. O tempo para a comunicação entre os rádios NRF24 pode variar tendo em conta as variantes referidas na secção 6.2.1.1. Quanto aos resultados da integração, apesar do modo dos sensores permitir testar todos os componentes, aquele que se mostrou mais eficiente foi o modo dos motores. O facto de possuir uma taxa de receção de mensagens mais elevada (54 msg/seg) permitiu que neste modo houvesse uma maior sincronização entre o movimento real e virtual. Este aspeto já era esperado visto que o modo dos sensores tem em conta os quatro atributos (latitude, longitude, altitude e orientação) enquanto que o dos motores tem em conta apenas um (altitude). Quanto maior o número de atributos analisados, menor é a taxa de receção de mensagens (quanto mais componentes introduzidos no arduino MEGA2560 menor é esta taxa) e menor é a sincronização do UAV virtual e real.
Estes resultados mostraram-se satisfatórios ainda que alguns pudessem ser melhorados. Existem algumas matérias como o voo do drone que podiam ser mais eficientes, no entanto, a presença de material defeituoso acabou por limitar um pouco o espaço de manobra para o voo do mesmo.

7.2 Trabalho Futuro

No decorrer do desenvolvimento deste projeto, novos objetivos foram sendo estabelecidos tendo em vista o aperfeiçoamento de certas funcionalidades.
Estes objetivos contemplam, numa primeira instância, a instalação de novos ESCs e motores no quadcopter. É imperativo que novos ESCs e motores sejam colocados para se conseguir estabelecer um voo sustentado. Neste projeto conseguiu-se fazer com que o quadcopter voasse, tendo-se, no entanto, reparado que o mesmo não se encontrava muito estável devido à presença de um ESC e motor defeituosos.
Em segundo plano, temos a conjugação do voo do drone com a plataforma de simulação. Neste momento é possível ligar e desligar os motores do quadcopter com os sensores instalados através da plataforma de simulação; todavia, não é possível colocá-lo a voar enquanto a plataforma recebe dados externos por causa do facto de não haver portas suficientes na placa controladora (UNO) ou de não se conseguir adaptar o software para controlo de voo noutra placa controladora com maior

número de portas e capacidade (MEGA2560).

Em terceiro lugar, seria aconselhável o desenvolvimento de câmaras especializadas para objetos simulados no ambiente virtual da plataforma de simulação. Na presente dissertação, para se observar o quadcopter virtual foi necessário criar uma aeronave nas redondezas para se conseguir observar o mesmo. A introdução de uma nova câmara neste género de veículos ajudaria a visualizar movimentos com grande extensão e assim evitar criar aeronaves só para se observar o comportamento do UAV virtual.

Seria também interessante pensar-se como incluir múltiplos *digital twins* no sistema, ou seja, vários agentes virtuais que pudessem representar o quadcopter real. Neste momento, apenas é permitido haver um UAV virtual todavia, é possível expandir para mais de um *digital twin* através do tratamento da porta de acesso ao arduino. Esta funcionalidade seria bastante importante se o objetivo no futuro passasse por criar redes de comunicação entre vários agentes virtuais em diferentes computadores.

Por último, temos a melhoria das mecânicas de movimentação do agente virtual na plataforma de simulação, através da introdução dos movimentos *pitch*, *roll* e *yaw* e do comportamento da plataforma de simulação quando existem falhas no envio de informação por parte do quadcopter. No presente projeto, o veículo move-se horizontalmente e verticalmente e roda sobre si mesmo. Com a introdução dos movimentos mencionados em cima, a movimentação do quadcopter, para além de mais complexa, seria mais completa e real.

Referências

[Aerotenna, 2016] Aerotenna (2016). OcPoC. Disponível em https://aerotenna.com/ocpoc-cyclone/l (consultado a 27 de Abril de 2018).

[Agência Nacional de Aviação Civil, 2016] Agência Nacional de Aviação Civil (2016). Regulamento n.º 1093/2016. *Diário da República n.º 238/2016, Série II*, E - Entidades administrativas independentes e Administração autónoma:36613 – 36622.

[Al-Kaff et al., 2018] Al-Kaff, A., Martín, D., García, F., de la Escalera, A., and Armingol, J. M. (2018). Survey of computer vision algorithms and applications for unmanned aerial vehicles. *Expert Systems with Applications*, 92:447 – 463.

[AliExpress, 2017a] AliExpress (2017a). 30a Simonk ESC (4 pcs com bec) para f450 s500 s550 rc quadcotper helicóptero. Disponível em https://pt.aliexpress.com/item/M-30A-30A-SimonK-ESC-4pcs-with-BEC-For-RC-Quadcotper-Helicopter/32250741238.html?spm=a2g0s.9042311.0.0.nC6ooS (consultado a 10 de Maio de 2018).

[AliExpress, 2017b] AliExpress (2017b). 4 pcs 2212 do motor 920kv brushless motor + 1045 prop para quadcopter f450 x525. Disponível em https://pt.aliexpress.com/item/Flycat-A2212-2212-1000KV-Brushless-Outrunner-Motor-For-F450-X525-Quadcopter-Multi-copter/32681345064.html?spm=a2g0s.9042311.0.0.nC6ooS (consultado a 10 de Maio de 2018).

[AliExpress, 2017c] AliExpress (2017c). F01979 1 par 10x4.5 "1045 1045r cw ccw hélice adereços de plástico preto para dji f450 f550 500 multi-copter fpv rc quadcopter zangão. Disponível em https://pt.aliexpress.com/item/F01979-1Pair-10x4-5-1045-1045R-CW-CCW-Propeller-Plastic-Props-Black-for-DJI-F450-500/32634648402.html (consultado a 10 de Maio de 2018).

[AliExpress, 2017d] AliExpress (2017d). Flysky afhds FS-i6 2.4g 6ch rc receptor transmissor com FS-ia6 fs-ia6b para heli avião uav zangão multicopter. Disponível em https://pt.aliexpress.com/item/Wholesale-FlySky-FS-i6-2-4G-6CH-AFHDS-RC-Transmitter-With-FS-iA6-FS-iA6B-Receiver/32577622860.html (consultado a 10 de Maio de 2018).

[AliExpress, 2017e] AliExpress (2017e). Multi-helicóptero quad-helicóptero kit quadro F450 quadx quad multicopter kk mk mwc oem241 ul21600. Disponível em https://pt.aliexpress.com/item/F450-Multi-Copter-Quad-copter-Kit-Frame-QuadX-Quad-MultiCopter-KK-MK-MWC-OEM241-UL21600/32786160808.html?spm=a2g0s.9042311.0.0.nC6ooS (consultado a 10 de Maio de 2018).

[Arduino, 2012] Arduino (2012). NewPing Library. Disponível em https://playground.arduino.cc/Code/NewPing (consultado a 3 de Março de 2018).

[Arduino, 2014a] Arduino (2014a). Servo Library. Disponível em https://www.arduino.cc/en/reference/servo (consultado a 3 de Março de 2018).

[Arduino, 2014b] Arduino (2014b). SPI Library. Disponível em https://www.arduino.cc/en/reference/SPI (consultado a 3 de Março de 2018).

[Arduino, 2014c] Arduino (2014c). Wire Library. Disponível em https://www.arduino.cc/en/reference/wire (consultado a 3 de Março de 2018).

[Arduino, 2015] Arduino (2015). RF24. Disponível em https://playground.arduino.cc/InterfacingWithHardware/Nrf24L01 (consultado a 3 de Março de 2018).

[Arduino, 2018a] Arduino (2018a). Arduino Mega 2560 rev3. Disponível em https://store.arduino.cc/usa/arduino-mega-2560-rev3 (consultado a 22 de Maio de 2018).

[Arduino, 2018b] Arduino (2018b). Arduino Uno rev3. Disponível em https://store.arduino.cc/usa/arduino-uno-rev3 (consultado a 22 de Maio de 2018).

[Arduino, 2018c] Arduino (2018c). How to interface GPS module (NEO-6m) with arduino. Disponível em https://create.arduino.cc/projecthub/ruchir1674/how-to-interface-gps-module-neo-6m-with-arduino-8f90ad (consultado a 10 de Maio de 2018).

[Arduino, 2018d] Arduino (2018d). MPU-6050 accelerometer + gyro. Disponível em https://playground.arduino.cc/Main/MPU-6050 (consultado a 16 de Maio de 2018).

[ArduPilot, 2012] ArduPilot (2012). ArduPilot Mega. Disponível em http://www.ardupilot.co.uk/# (consultado a 27 de Abril de 2018).

[ArduPilot FlyMaple, 2013] ArduPilot FlyMaple (2013). Building ardupilot for flymaple on linux. Disponível em http://ardupilot.org/dev/docs/building-apm-for-flymaple.html (consultado a 27 de Abril de 2018).

[Association for Unmanned Vehicle Systems International, 2013] Association for Unmanned Vehicle Systems International (2013). The Economic Impact of Unmanned Aircraft Systems Integration in the United States. Disponível em http://www.auvsi.org/our-impact/economic-report (consultado a 9 de Novembro de 2017).

[Austin, 2010] Austin, R. (2010). *Unmanned Aircraft Systems - Human Factors in Aviation*, volume The Deployment Of UAV Systems of *Aerospace Series*, pages 245–280. Wiley.

[Boccalatte et al., 2004] Boccalatte, A., Gozzi, A., Grosso, A., and Vecchiola, C. (2004). Agent-service. in F. Maurer and G. Ruhe (eds.). *Proceedings of the 16th International Conference on Software Engineering and Knowledge Engineering (SEKE'2004)*, page 45–50.

[Brooking, 2015] Brooking, J. (2015). Project YMFC-AL - the arduino auto-level quadcopter. Disponível em http://www.brokking.net/ymfc-al_main.html (consultado a 20 de Maio de 2018).

[Cai et al., 2009] Cai, G., Chen, B. M., Lee, T. H., and Dong, M. (October 2009). Design and implementation of a hardware-in-the-loop simulation system for small-scale UAV helicopters. *Mechatronics*, 19:1057–1066.

[Câmara, 2013] Câmara, A. (2013). Controlo de tráfego aéreo usando o microsoft flight simulator x. Master's thesis, Department of Informatics Engineering, University of Coimbra, Coimbra, Portugal.

[Carvalho, 2014] Carvalho, J. C. P. V. (2014). Simulação Realista de uma Plataforma Aérea. Master's thesis, Faculdade de Engenharia da Universidade do Porto.

[Components101, 2017] Components101 (2017). HC-SR04 Ultrasonic Sensor. Disponível em https://components101.com/ultrasonic-sensor-working-pinout-datasheet (consultado a 10 de Maio de 2018).

[Craighead et al., 2007] Craighead, J., Murphy, R., Burke, J., and Goldiez, B. (2007). A Survey of Commercial Open Source Unmanned Vehicle Simulators. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 852–857.

[Elecrow, 2018] Elecrow (2018). NRF24L01+PA+LNA Wireless Module. Disponível em https://www.elecrow.com/wiki/index.php?title=NRF24L01%2BPA%2BLNA_Wireless_Module (consultado a 10 de Maio de 2018).

[European Aviation Safety Agency, 2012] European Aviation Safety Agency (2012). Civil drones. Disponível em https://www.easa.europa.eu/ (consultado a 9 de Novembro de 2017).

[Força Aérea Portuguesa, 2018] Força Aérea Portuguesa (2018). Agusta-Westland EH-101 Merlin. Disponível em http://www.emfa.pt/www/aeronave-17-agusta-westland-eh-101-merlin (consultado a 27 de Março de 2018).

[Gabinete de Prevenção e Investigação de Acidentes com Aeronaves, 2012] Gabinete de Prevenção e Investigação de Acidentes com Aeronaves (2012). Investigação de acidentes e incidentes. Disponível em http://www.gpiaa.gov.pt/ (consultado a 9 de Novembro de 2017).

[Hallenborg, 2008] Hallenborg, K. (2008). Information Science and Computing. In *Advanced Studies in Software and Knowledge Engineering*, chapter Core Design Pattern for Efficient Multi-agent Architecture, pages 29–36. Supplement to the International Journal "Information Technologies and Knowledge", Institute of Information Theories and Applications FOI ITHEA, Sofia, 1000, P.O.B. 775, Bulgaria.

[Haulman, 2003] Haulman, D. L. (2003). Unmanned Aerial Vehicles in Combat. Technical Report 200520, Air Force Historical Research Agency, Maxwell Air Force Base, Alabama, USA.

[Hughes, 2013] Hughes, R. (2013). The producer consumer pattern. Disponível em https://dzone.com/articles/producer-consumer-pattern (consultado a 6 de Junho de 2018).

[Joshi, 2017] Joshi, D. (2017). Exploring the latest drone technology for commercial, industrial and military drone uses. Disponível em http://www.businessinsider.com/drone-technology-uses-2017-7 (consultado a 3 de Abril de 2018).

[Kamrani and Ayani, 2007] Kamrani, F. and Ayani, R. (2007). Using on-line simulation for adaptive path planning of uavs. In *Distributed Simulation and Real-Time Applications, 2007. DS-RT 2007. 11th IEEE International Symposium*, pages 167–174.

[Leite, 2015] Leite, C. E. T. (2015). Plataforma de Controle para Múltiplos Veículos Aéreos Não Tripulados (VANTs). Master's thesis, Universidade Federal do rio Grande do Sul.

[Meyer et al., 2012] Meyer, J., Sendobry, A., Kohlbrecher, S., Klingauf, U., and von Stryk, O. (2012). Comprehensive Simulation of Quadrotor UAVs using ROS and Gazebo. In *Simulation, Modeling, and Programming for Autonomous Robots*, pages 400–411, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Microsoft, 2008] Microsoft (2008). Simconnect SDK Reference. Disponível em https://msdn.microsoft.com/en-us/library/cc526983.aspx (consultado a 27 de Abril de 2018).

[Neto, 2017] Neto, T. L. P. (2017). Flexible Air Traffic Control Management. Master's thesis, Faculdade de Engenharia da Universidade do Porto.

[Newcome, 2005] Newcome, L. R. (2005). *Unmanned Aviation: A Brief History of Unmanned Aerial Vehicles*. American Institute of Aeronautics and Astronautics.

[North Atlantic Treaty Organization, 2015] North Atlantic Treaty Organization (2015). Standardisation Agreement (STANAG) 4586: Standard Interfaces of UAV Control System (UCS) for NATO UAV Interoperability. *NATO Standardization Agency (NSA)*, page 1–14.

[Observador, 2017] Observador (2017). Incidentes com drones. Disponível em https://observador.pt/2017/06/26/incidentes-com-drones-dispararam-no-ano-passado-este-ano-ja-houve-dez/ (consultado a 20 de Novembro de 2017).

[OpenPilot Manual, 2014] OpenPilot Manual (2014). CC3D/Atom. Disponível em http://opwiki.readthedocs.io/en/latest/user_manual/cc3d/cc3d.html (consultado a 27 de Abril de 2018).

[Paparazzi, 2003] Paparazzi (2003). Paparazzi UAV. Disponível em http://wiki.paparazziuav.org/wiki/Main_Page (consultado a 27 de Abril de 2018).

[Pixhawk, 2013] Pixhawk (2013). Pixhack V3. Disponível em https://docs.px4.io/en/flight_controller/pixhack_v3.html (consultado a 27 de Abril de 2018).

[Pixhawk, 2016] Pixhawk (2016). Cube. Disponível em https://docs.px4.io/en/flight_controller/pixhawk-2.html (consultado a 27 de Abril de 2018).

[Raza and Gueaieb, 2010] Raza, S. A. and Gueaieb, W. (2010). Intelligent Flight Control of an Autonomous Quadrotor. In Casolo, F., editor, *Motion Control*, chapter 12. InTech, Rijeka.

[RCMoment, 2017] RCMoment (2017). ZOP power 3s 11.1v 2200mah 30c xt60 plug LiPo battery for 210 250 280 racing drone 450 helicopter rc car boat. Disponível em https://www.rcmoment.com/p-rm8993.html (consultado a 10 de Maio de 2018).

[Robotics, 2013] Robotics, E. (2013). Erle-Brain 3. Disponível em https://erlerobotics.com/blog/ (consultado a 27 de Abril de 2018).

[RobSense, 2016] RobSense (2016). Phenix Pro. Disponível em http://www.robsense.com/pro-en.html (consultado a 27 de Abril de 2018).

[Silva, 2011] Silva, D. C. (2011). *Cooperative Multi-Robot Missions: Development of a Platform and a Specification Language*. PhD thesis, Faculdade de Engenharia da Universidade do Porto.

[Stratistics Market Research Consulting, 2017] Stratistics Market Research Consulting (2017). UAV Drones-Global Market Outlook. Technical Report 4519515.

[United States Department of Defense , 2005] United States Department of Defense (2005). Unmanned Aircraft Systems Roadmap 2005-2030. Technical Report ADA445081, Office of the Secretary of Defense.

[UVS International, 2012] UVS International (2012). Remotely Piloted Systems. Disponível em https://uvs-international.org/ (consultado a 20 de Novembro de 2017).

[Veloso et al., 2014a] Veloso, R., Kokkinogenis, Z., Passos, L. S., Oliveira, G., Rossetti, R. J. F., and Gabriel, J. (2014a). A platform for the design, simulation and development of quadcopter multi-agent systems. In *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–6.

[Veloso et al., 2014b] Veloso, R., Oliveira, G., Passos, L. S., Kokkinogenis, Z., Rossetti, R. J. F., and Gabriel, J. (2014b). A symbiotic simulation platform for agent-based quadcopters. In *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–6.

REFERÊNCIAS

Anexo A

Anexos

Build and Integrate a Quadcopter in a Simulation Platform (User Manual)

Leonardo Ferreira

July 12, 2018

Contents

1 Introduction

In recent years, Unmanned Aerial Vehicles or UAVs have grown at a rapid pace due to recent developments in the Aeronautics section. This accelerated growth also happened thanks to the characteristics presented by this kind of vehicles. Their size and features allow them to perform a varied set of tasks[8]. However, most of the solutions offered by the market are expensive and complex and that's where this project comes in.

1.1 Objective

The mission of this project is to show how to build and integrate a quadcopter in a simulation platform for multi-vehicle missions. Through this manual we detail the main functionalities of the project developed and how to achieve them. It is important to know that this work is divided in two phases: **quadcopter flight** and **quadcopter integration with simulation platform**. First, we give some insights about some important aspects related to this project and then show how to run every feature related to these two phases.

2 Quadcopter Flight - Concept

This phase assembles several components (will be explained later) in order the drone can make a sustainable flight. To fulfill this requirement, aspects such as the layout of the material, connections, configuration and calibration of each of the components to the Arduino UNO are taken into account. If some of the above is neglected or simply poorly done, it is likely that the quadcopter will show strange behaviors like flips (quadcopter rotates on itself realizing an unexpected movement that can harm the same).

3 Quadcopter Integration with Simulation Platform - Concept

All the development made for integration of quadcopter in the simulation platform is decribed. At first, the UAV agent type is explored with the representation of this agent on the simulation platform. Next is highlighted how the information flows from the real quadcopter to the simulation platform and how it is achieved. It will also be detailed how the external data is ”transformed” into virtual data so that the virtual quadcopter can interpret and use it. Following the previous point, the messages with STANAG format used to send platform commands to the real quadcopter, are discussed.

3.1 UAV agents

This agent type is different from the others present in the simulation platform because this kind of agent can receive and interpret external data and send commands to the real UAV while the others can't. Each UAV agent is created based on two parameters:

- Model - a string representing the model to be adopted in the game engine of the simulation platform;
- SIMCONNECT_DATA_INITPOSITION - represents the simconnect data structure which allows to initialize a UAV agent in a certain position. This data structure considers latitude, longitude, altitude, heading, speed and other attributes.

3.1.1 UAV Representation

To create UAV agents in the simulated environment it is necessary to use a function provided by simconnect: AICreateSimulatedObject. However, this function has a major drawback, which is that it does not give a view on the object. To counter this problem, an aircraft, the Osprey V-22, was placed in front of it so we could visualize the virtual quadcopter. The UAV agent is the helicopter behind the OSPREY-V22 (green plane) in Fig 1.

Figure 1: Quadcopter in a simulated environment

3.2 Real Data to Virtual Data

The platform is equipped with two threads. The first one extracts information from the ground control station into a data structure, which contains information about latitude, longitude, altitude and heading and the other allows the virtual quadcopter to assume a new position/layout based on the data collected previously. The platform receives messages with the following format:

Message Format - latitude | longitude | altitude | heading

Example - 41.224511 | -8.621419 | 30 | 120

After receiving a message, it is divided based on the partitions present in the message and then each piece is added to the respective list. Parallel to this process, the virtual quadcopter attributes latitude, longitude, altitude and heading are constantly being changed. When the virtual UAV receives new data, his position is updated by the function SetDataOnSimObject with the following attributes:

- SIMCONNECT_DATA_DEFINITION_ID - specifies the user-defined data definition ID;
- SIMCONNECT_OBJECT_ID - specifies the ID of the object to be changed;

- SIMCONNECT_DATA_SET_FLAG - Enables the data in a different format;
- void* - New data to be assumed by virtual quadcopter. In this case, it corresponds to the new UAV position.

3.3 STANAG Commands

The 3rd edition of STANAG 4586[7] was used to send commands from the simulation platform to the real quadcopter. This document defines several message structures used in the UAVs that allow to execute several functionalities inherent to these autonomous vehicles.
Next, we discuss the STANAG message structure, the main commands used as well as the changes done to these.

3.3.1 Structure

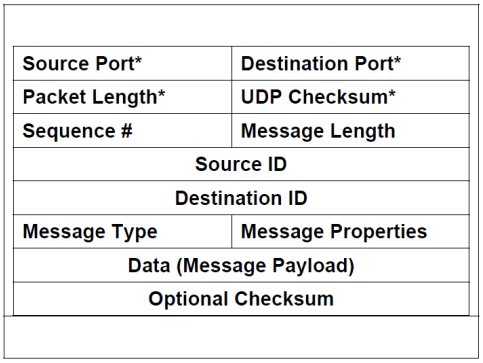


Figure 2: STANAG message structure[7]

As shown in Fig 2, each message has:

- **Sequence number** - segment the data of a message into block sequences of equal size;
- **Message length** - information size of a message;
- **Source ID** - occupies 4 bytes and concerns the component that sends the message. It is divided into three parts: the country code (8 bits), the component code (3 bits) and the remaining 21 bits are reserved for the country chosen use.
- **Destination ID** - The destination ID is similar to the source ID except that it is relative to the component that receives the message;
- **Message type** - occupies two bytes and identifies the type of message;
- **Message properties** - occupies 2 bytes and is divided into four parts: acknowledgment (1 bit), IDD version (7 bits), checksum length (2 bits) and future use (6 bits). Together they constitute two types of messages. Pull messages are messages that are transmitted in response to a request whereas Push messages are messages transmitted periodically taking into account a given event;
- **Data** - identify several fields related to the information to be transmitted.

The remaining parameters are optional and were not used since there was no need.

3.3.2 Main Commands

Three types of commands were used to simulate various quadcopter functionalities. In addition, some customization was made on them due to the fact that some parameters were not relevant and because it wasn't possible to discern which type of maneuver to perform on the quadcopter:

- **Vehicle Steering** - allows the quadcopter to move vertically and rotate.
  - Message Type - 2002;
  - Message Content or data - 7;
  - Maneuver Type - CMD_UP, CMD_DOWN and CMD_ROT;
- **AV Position Waypoint** - allows the quadcopter to move horizontally.
  - Message Type - 13002;
  - Message Content or data - 15;
  - Maneuver Type - CMD_FORW and CMD_BACK;
- **Engine** - turns on/off the quadcopter engines.
  - Message Type - 2008;
  - Message Content or data - 67;
  - Maneuver Type - ENGINE.

4 Materials used

Several materials were used to build and integrate the quadcopter in the simulation platform. Next, the hardware and software used are presented.

4.1 Hardware

- Arduino UNO - hardware board containing a programmable microcontroller. It is characterized by being very light and simple board, equipped with an ATmega328 microcontroller. It has a memory capacity of 32 KB;
- Arduino MEGA2560 -Ideal controller board for complex projects. It is characterized by being heavier board, more complex and with greater capacity when compared to the UNO board. It is equipped with an ATmega2560 microcontroller and has 54 input ports of which 16 are analog ports, 15 with PWM properties and 4 for serial communication.
- MPU6050 Sensor - has an accelerometer and a high precision gyroscope. It is equipped with a DMP (Digital Motion Processor), in charge of making complex calculations with the sensors. In our case, these data are later used by quadcopter for better navigation over the space in which it is located;
- HC-SR04 - Distance sensor capable of measuring distances between 2m and 5m with good accuracy. It has a circuit where the emitter and the receiver are connected. This sensor emits ultrasonic signals that reflect on the object reached and return to the sensor, indicating the distance to the target;
- GPS NEO 6M - gives information about his position (latitude, longitude and other details). It has good accuracy and can have several applications in the market. The communication of this module is done by serial port;
- 2 NRF24L01 + PA + LNA Radios - Module that allows wireless communication between devices such as arduino controller boards. Uses a 2.4GHZ transmitter NRF24L01. His range varies according to the presence of an antenna. Without antenna, the range is 50m outdoors while with antenna, the range goes up to 1000m. The range in indoor places varies depending on the number and arrangement of walls as well as the types of materials present on the same. For this project, two radios were used in order to allow the communication between two Arduinos (UNO and MEGA);
- 4 ESCs SimonK 30A - electronic speed controllers with 30A power. They receive Pulse Width Modulation (PWM) pulses from the controller board and turn them into rotation in the motors. They are composed of power transistors and a microcontroller. Four of these were used (each ESC is associated to an engine);
- 4 engines Brushless 920kv - quadcopter engines. They are characterized by being light, with reduced noise and with great power of rotation. There are two types of engines: one rotates clockwise and another rotates counterclockwise. It was used four engines of this nature (two of clockwise and two of counterclockwise).;
- F450 Quadcopter frame - quadcopter body. It is characterized by being a very strong and light kit;
- ZOP Power 11.1V 2200MAH 3S 30C Battery - gives power to quadcopter. It is characterized by being a lightweight battery with good durability;
- FlySky FS-i6 kit - controls the quadcopter through a radio transmitter and radio receiver. The radio transmitter is composed by a simple panel that allows to configure the channels of communication with the receiver, among other variables. It is the most affordable and lightweight UAV radios transmitters. It operates at a frequency of 2.4 GHz (can interfere with other WIFI networks). The radio receiver is sensitive device that has 6 communication channels and can operate at great distances from the

transmitter. It is also equipped with two antennas for more effective communication and free of external interference;

- 2 pairs of propellers - helps the engines to sustain flight. Generally, they need to be balanced so that the quad can maintain balance during the flight.

Basic material was also used to assist in the assembly and connection of all quadcopter components.

4.2 Technologies

We used:

- YMFC-ALL - Several flight controllers were studied; however, this was the chosen since it has all the features needed for flight;
- Visual Studio - quick tool equipped with a lot of features that can help the development like the integration with a TFS that is being used by the development team for code sharing and other stuff;
- Arduino IDE - very useful and important when working with hardware boards such as arduino UNO and MEGA2560. It has good documentation and a lot of examples that help understanding all kind of concepts related Arduinos.

5 Quadcopter Flight

In here all the materials, mounting process and configuration related to quadcopter flight are detailed. It is also shown how to execute the quadcopter flight. There are some factors that can cause some flight problems (section 5.4.1).

5.1 Materials Involved

The materials used for the flight were: quadcopter frame, arduino UNO, ESCs, FlySky radio transmitter and receiver, MPU6050 sensor, battery, engines and propellers.

5.1.1 Propellers and Engines

There are two types of propellers and engines, some designed to rotate clockwise and others to rotate counterclockwise. This aspect is very important to understand because they have a big influence on the quadcopter flight. Later on, it is explained how many do we used and where did we put them.

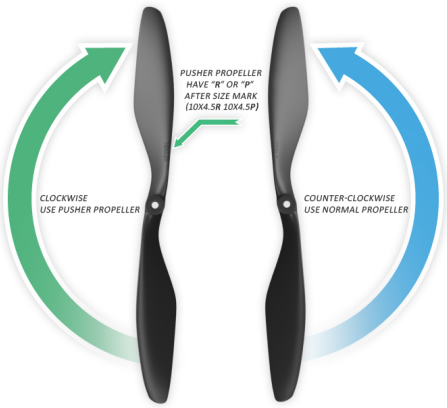


Figure 3: Propellers types[6]

As we can see in Fig 3 the propellers have different structures, depending on whether they are clockwise or counterclockwise. In our case, since two clockwise and two counterclockwise engines are used, two clockwise and two counterclockwise propellers are also used. It should be noted that a clockwise propeller must be paired with a clockwise engine (the same happens with a counterclockwise propellers and engine). The reason we need two engines and two propellers with clockwise direction and two motors and two propellers with counterclockwise direction is related to the balance and lifting of the quadcopter from the ground. When the propellers rotate, applying Newton's third law (each action establishes an action-reaction pair), they produce torque effect on the quadcopter's body in the opposite direction, for example the engines "push" the air downward while the air "pushes" the engines upwards. If all the engines rotate in the same direction, the quadcopter would continue to rotate in that direction, executing a yaw movement. To prevent this effect, motors and propellers of opposite directions were chosen in equal numbers.

5.2 Mounting Guide

Steps to be followed in order to mount the quadcopter:

- Step 1 - Mount the quadcopter frame. When this is done, the vehicle presents two levels (top and lower level);
- Step 2 - Put the arduino UNO in the center top level of the quad. This placement should be the most central possible for better weight distribution;
- Step 3 - Incorporate the MPU6050 sensor in the center under the quadcopter ((the lower level was used for a better utilization of the space). It is advisable to use double sided tape on the sensor to reduce vibration and increase performance. It should be aligned with the front of the quadcopter so as not to affect the information coming from the gyroscope. If the MPU6050 is improperly fixed or poorly positioned it can create problems during the flight, such as the occurrence of flips. The connection of this component to the Arduino UNO is demonstrated at the end of the mounting guide for the quadcopter flight;
- Step 4 - The FS-IA6 receiver (receives signals from the FS-i6 transmitter) must be on the lower level of the quad vehicle. Before connecting it to the arduino UNO it is important to understand how to read the inputs given by the radio transmitter FS-i6 (inputs change the quadcopter's thrust, pitch, roll and yaw) and, in this matter, it is important to clarify what are pins change interrupt. These pins, supported by the Arduino UNO's ATmega328 (microcontroller), are important since they allow you to interrupt the main program and perform other tasks whenever the state of a digital port is changed. With the help of to the ATmega328 microcontroller manual[2] and the arduino UNO port map (presented next), and taking into account that it is necessary to read four variables of the FS-i6 transmitter radio through four communication channels of the FS-IA6 receiver, four pins change interrupt between PCINT0 and PCINT7 are required.

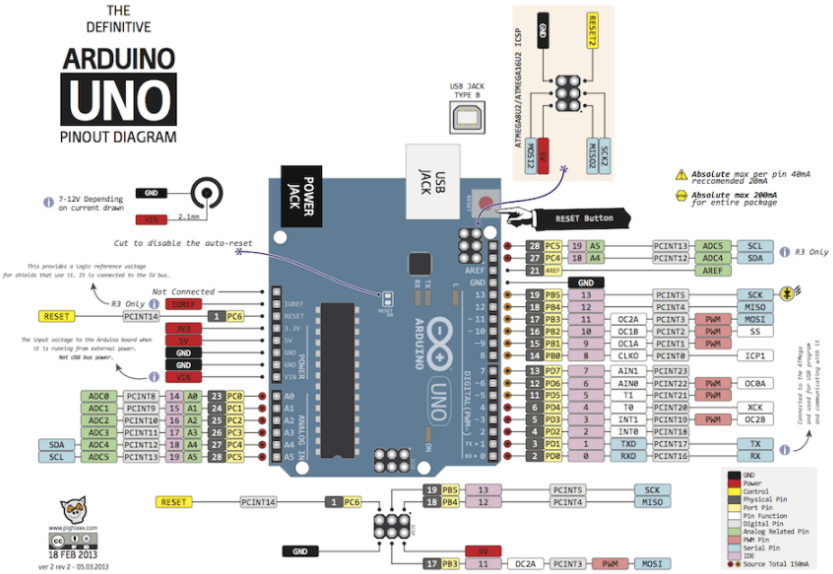


Figure 4: Arduino UNO pin map[3]

Through Fig 4, we can see that the ports PCINT0, PCINT1, PCINT2, PCINT3 in the arduino UNO

correspond to ports 8, 9, 10 and 11. The connection of this component to the Arduino UNO is demonstrated at the end of the mounting guide for the quadcopter flight.
The position of the two antennas of the receiver is also important. They must be perpendicular to each other;

- Step 5 - The battery should be also on lower level in the most center position due to his weight. It is also important to analyze ways to lower the voltage supplied by the battery to the Arduino UNO and for that we used power regulators. It can be any power regulator since he can manage any tension step down. In this case, the battery has a voltage of 12V while the Arduino UNO operates with a voltage of 5-9V. Therefore, to avoid problems in the power supply (can damage the UNO board due to power excess), a regulator with the capacity to lower the voltage supplied by the battery to levels acceptable by the Arduino UNO was used. The regulator was placed on the lower level at the rear of the quadcopter;
- Step 6 - Put the engines in quadcopter extremities. The front right and rear left arms of the quad must have counter-clockwise engines while in front left and rear right must have clockwise engines;
- Step 7 - The ESCs must be allocated under the quadcopter arms the closest to the center. Then connect the signal from them (only the white wires, the others do not need to be connected) to the arduino UNO outputs. The ESCs must also be connected to the engines by connecting the three wires of the engines with their three ESCs inputs. The connection of this component to the Arduino UNO is demonstrated at the end of the mounting guide for the quadcopter flight;
- Step 8 - The propellers must be put on the top of the engines. Notice that a counterclockwise propeller must always be on a counterclockwise engine and vice-versa.

In terms of connections, we have done the following ones to arduino UNO:

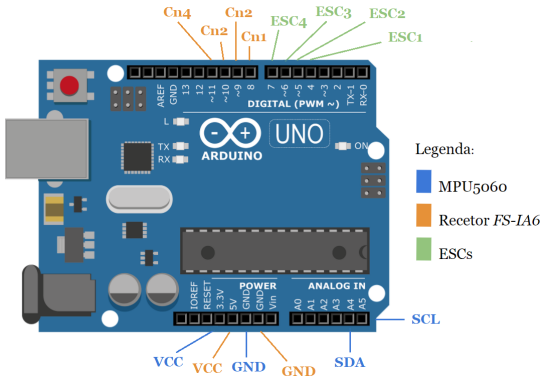


Figure 5: All the connections made on UNO for flight

The Fig 5 is an example on how to connect the components (FlySky transmitter and receiver, ESCs and MPU6050) to the UNO board.
The MPU6050 sensor has four signals: GND, VCC, SDA and SCL. The GND (arduino UNO has three GND ports) and VCC (5V and 3.3V ports are available being the 5V port the chosen since the 3.3V was already occupied) are the standard ones while the SDA (A4) and SCL (A5) are the ones that retrieve the gyroscope information.
The ESCs must be connected to any digital port with preference to those that have PWM characteristics since ESCs work based on pulses. The ESC1 is located on the front right of the quadcopter, the ESC2 on the rear right, the ESC3 on the rear left and the ESC4 on the front left. The ports chosen were 4, 5, 6 and 7 respectively.

The FlySky connections are a little more trickier than it looks. With the help of the ATmega328 microcontroller manual[2] and the arduino UNO ports map presented previously we chose the ports PCINT0, PCINT1, PCINT2 and PCINT3 which represent the ports 8, 9, 10 and 11 on the arduino UNO. It is possible to use other ports if we change them on the software (requires knowledge and it is not advised for beginners).

5.3 Configuration Guide

Important Note: Before configure the quadcopter, it is advisable to remove the propellers to avoid unnecessary accidents.

Since there is no need to present a very elaborate but efficient and sustainable flight, we chose the simplest and best-documented software, **Project YMFC-AII**[1] .
Although being a simple software it implies some knowledge, especially if you need to change the controller board. This software was developed to work on the arduino UNO; however, it is possible to change to other boards. If so, changes must be done on the pins change interrupt.
To configure the arduino UNO of the quadcopter, it is necessary to include three files, through Arduino IDE. Steps to be followed in order to configure the quadcopter:

Include the file "YMFC-AL_setup.ino" - responsible for configuring the quadcopter with the FlySky FS-i6 radio transmitter. The FS-IA6 receiver communication channels are tested and configured and the quadcopter's gyro axes are calibrated correctly using the transmitter radio handles (thrust, pitch, yaw and roll). The information is then stored in the Arduino's memory, that is, in the EEPROM (Electrically-Erasable Programmable Read-Only Memory) for later use by the Flight Controller. After including the file, some tasks need to performed. These tasks are detailed on the step 5 (Receiver and gyro check) on the software author's page: `http://www.brokking.net/ymfc-al_main.html`;

Include the file "YMFC-AL_esc_calibrate.ino" - calibrates the four ESCs arranged in the quadcopter so that all the engines work in tune and at the same speed. In addition it allows us to observe if the quadcopter's gyroscope is well calibrated and if each of the engines is working correctly. After including the file, some tasks need to performed. These tasks are detailed on the step 6 (Calibrate the ESC's) on the software author's page: `http://www.brokking.net/ymfc-al_main.html`;

Include the file "YMFC-AL_Flight_controller.ino" - main component that relates the previous elements so that the quadcopter flies. It takes into account information from engines, ESCs, battery, gyroscope, among others. The only task that this step requires is load the file into the arduino UNO.

After loading all these files into the Arduino, the quadcopter is ready to receive commands from the FlySky FS-i6 transmitter radio and take off. If the quadcopter manifests strange behavior like flips, the whole configuration process must be repeated. If the quadcopter can fly but with aggressive behavior then it is recommended to do what is called PID (Proportional integral derivative) Tunning. This method allows you to tune the quadcopter flight by changing some variables from the "YMFC-AL_Flight_controller.ino" file.

5.4 How to run the quadcopter flight

After completing the mounting and configuration of the quadcopter and testing the components, the quadcopter is ready to take off. Steps to make it fly:

- Choose an open area where there is no risk of hurting people (an area with little wind is preferred);
- Put the quadcopter on a flat surface where the slope is reduced (the less slope the better). The slope influences the gyroscope readings;

- Turn on the quadcopter and radio transmitter batteries and wait until all engines spin in tune;
- Slowly increase the thrust (push the left handle up) so the quadcopter can take off;
- When in the air you can execute pitch movement by variating the right handle (up and down), the roll by variating also the right handle (left and right) or the yaw by changing the values on the left handle (left and right);
- To land the quadcopter, slowly decrease the thrust, that is, push the left handle down;
- If the quadcopter show aggressive behavior in the air or ground, it is recommended to do the PID Tunnig.

5.4.1 Flight Factors

There are some factors that can change the flight performance. These are: poor calibration on the gyroscope, defective or poor calibrated ESCs, wrong engines (there are clockwise and counterclockwise engines), misplaced propellers (also clockwise and counterclockwise propellers) and bad weight distributing.

6 Quadcopter Integration with Simulation Platform

In here all the materials, mounting process and configuration done to quadcopter integration in the simulation platform are detailed. It is also shown how to test the interaction between real quadcopter with virtual quadcopter.

6.1 Materials Involved

The materials used for the flight were: quadcopter frame, arduino MEGA2560 and UNO, HC SR04 sensor, MPU6050 sensor, GPS NEO 6-M, NRF24 radios, ESCs, battery, engines and propellers.

6.2 Mounting Guide of quadcopter

Before approaching the assembly of the quadcopter with the sensors it is important to clarify that some the components that had been installed for the flight of the quadcopter were disconnected and some were even removed. Steps to be followed in order to mount the quadcopter:

- Step 1 - After disassembling the elements of the quadcopter with the exception of the frame, the ESCs and the MPU6050 sensor, the arduino MEGA2560 was incorporated in the top level of the quadcopter. The MEGA2560 arduino was chosen over the arduino UNO because is a board with a larger number of ports available and with a larger memory capacity. With the installation of elements such as enginnes, ESCs, MPU6050, sonar, gps and radio, the number of ports to be used would be big (about 19 ports), therefore is advisable to use a board with higher port availability. Another reason for choosing the MEGA2560 arduino is because none of the elements interfered with others, a phenomenon that happened in the arduino UNO with some frequency. This behavior occurs because some ports share properties (timers and other features) that in some cases, when shared by multiple components, may give inaccurate or incorrect readings. With the help of the ATmega328 microcontroller manual[2], you can see the properties of the UNO controller board and its ports;
- Step 2 - Place the NRF24 radio on the top level of the quadcopter, preferably in a location a little away from the rest of the sensors to avoid receiving or causing interference. Being this component one of the most relevant in this project, in terms of hardware, it is important to clarify some points about it before connecting it to the Arduino MEGA2560. The NRF24 radio has several signals such as VCC, GND, CE, CSN, SCK, MOSI and IRQ (not used), signals that must be connected to ports with certain properties. These allow the communication with other devices through a standard called SPI (Serial Peripheral Interface). With the help of the ATmega2560 microcontroller manual[4] and the MEGA2560 arduino port map (below) it is possible to understand where to connect this signals to the MEGA2560 board.

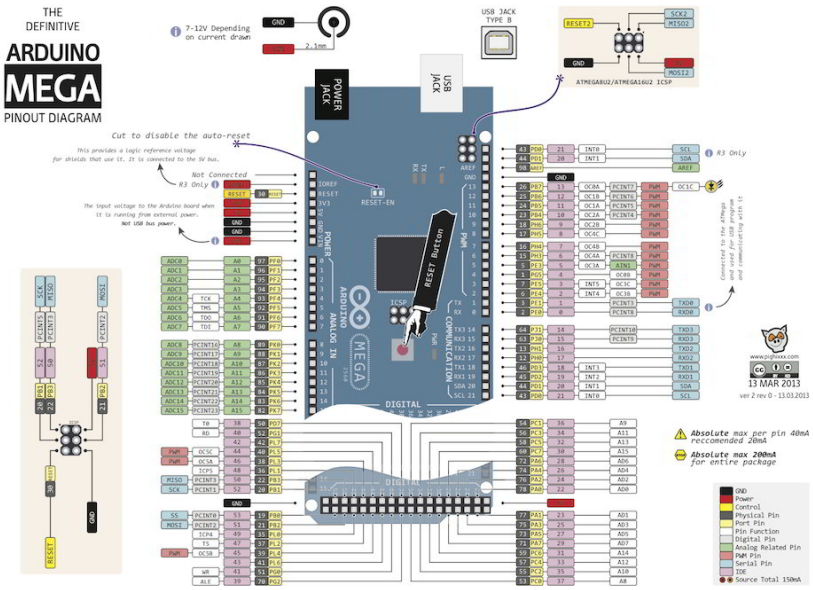


Figure 6: Arduino MEGA pin map[5]

As can be seen in Fig 6, the SCK, MOSI, and MISO signals have specific locations to be connected in the Arduino Mega2560. The CE and CSN signals can be allocated to any digital port. The connection of this component to the Arduino MEGA2560 is demonstrated at the end of the mounting guide for the quadcopter integration;

- Step 3 - Put the HC-SR04 sensor in a position where it is possible to retrieve the distance between the quadcopter and the floor and secure it so that the sensor remains secure. This sensor has several signals such as GND, VCC, TRIGGER and ECHO. The connection of this component to the Arduino MEGA2560 is demonstrated at the end of the mounting guide for the quadcopter integration;
- Step 4 - Place the NEO 6-M GPS on the top of the quadcopter, in any location as long as it does not interfere with other sensors. Must be attached to provide accurate readings. It has several signals such as VCC, GND, RX and TX. SCL. The connection of this component to the Arduino MEGA2560 is demonstrated at the end of the mounting guide for the quadcopter integration;
- Step 5 - Connect the ESCs to the MEGA2560 board. This connection must be made on ports with PWM characteristics since the ESCs operate on the basis of pulses which are then sent to the motors. Referring to the ATmega2560 microcontroller manual[4] and the MEGA2560 arduino port map (Fig 6), it is possible to identify the ports that work with PWM. The connection of this component to the Arduino MEGA2560 is demonstrated at the end of the mounting guide for the quadcopter integration;
- Step 6 - The battery, MPU6050 sensor, engines and propellers remain in the same position when were installed for flight.

Most of the components used (GPS, sonar and MPU6050) use a 5V VCC power supply. However, it is not possible to connect the three VCCs of the three components to the Arduino MEGA2560 since this board only offers one 5V port. To counter this problem, a breadboard was installed. This enables the ports reuse.

In terms of connections, we have done the following ones:

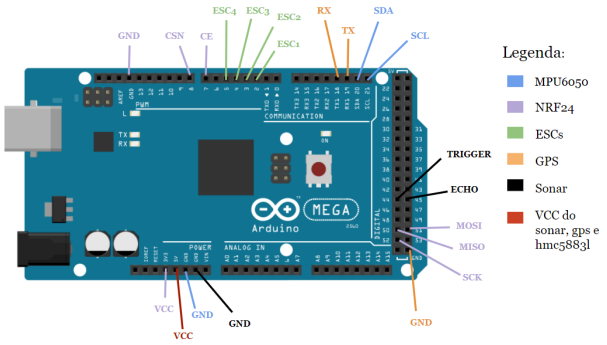


Figure 7: All the connections made on arduino MEGA2560 quadcopter

The Fig 7 is an example on how to connect the components (GPS, sonar, MPU6050, NRF24 radio and ESCs) to Arduino MEGA2560.
The MPU6050 sensor has four signals: GND, VCC, SDA and SCL. The GND (GND) and VCC (5V or 3.3V) are the standard ones while the SDA (20) and SCL (21) are the ones that retrieve the gyroscope information. The GPS also has four signals: GND, VCC, TX and RX. The GND (GND) and VCC (5V or 3.3V) are the standard ones while the TX (RX) and RX (TX) obtain data about the gps location. The Mega2560 board contains three ports with RX characteristics (ports 14, 16 and 18) and three ports with TX characteristics (ports 15, 17 and 19). The sonar is probably the most simple component in this project, equipped with four signals: GND, VCC, TRIGGER and ECHO. While GND is connected to GND and VCC to the 5V power, the TRIGGER and ECHO signals are connected to any PWM port. The ports chosen were TRIGGER (44) and ECHO (45). The ESCs, like discussed in the flight section, must be connected to any digital port with preference to those that have PWM characteristics. The ports chosen were 2, 3, 4 and 5.
The NRF24 radio connections require a little more attention when compared to the previous ones. The NRF24 radio has several signals such as VCC, GND, CE, CSN, SCK, MOSI and IRQ (not used), signals that must be connected to ports with certain properties to allow communication with other devices through a standard communication called SPI (Serial Peripheral Interface). With the help of the ATmega2560 microcontroller manual[4] and the arduino MEGA2560 ports map it is possible to identify which ports connect to these signals. The SCK, MOSI, and MISO signals have specific locations to connect to the Arduino Mega2560 while The CE and CSN signals can be allocated to any digital port. Thus, we connected the signal SCK to port 52, MOSI to 51, MISO to 50, CE to 7 and CSN to 8.

6.3 Configuration Guide of the quadcopter

The quadcopter was assembled to be able to read the GPS, sonar and MPU6050 data, send pulses from the ESCs to the engines and read commands from the simulation platform. In addition to the assembly of these components in quadcopter, software was developed to meet these needs.
First we studied the libraries SPI and RF24, NewPing, NMEAGPS and GPSport and Wire and then elaborated a file named ”Quadcopter.ino” which encapsulates all the functionalities of the components mentioned. In this file:

- We created an NMEAGPS object, a sonar object with a maximum distance of 2m with the ports declared in the assembly of the quadcopter and another object of the type radio with the ports mentioned in the assembly of the UAV. Two data structures were designed: myData and myCommand. MyData is responsible for storing sensor information (latitude, longitude, altitude and heading) while myCommand is in charge of storing the commands sent by the simulation platform;

- The pipes which the information flows were configured. This configuration allowed the exchange of information between the quadcopter and ground control station without mixing data.
- After all the setup is complete, the quadcopter is ready to receive information from the sonar, gps and MPU6050 data and notify the ground control station about the same data through the NRF24 radio component. The sonar provides the distance from the ground through the function: sonar.ping_result. The reading of the GPS data is done with a simple read, that is, gps.read() and the heading is provided by the MPU6050 with the function: Wire.read().

6.4 Mounting Guide of the ground control station

The process of setting up the ground control station is a simple and quick process to make. Steps to be followed:

- Place the NRF24 radio into the arduino UNO. As in the quadcopter mounting it is important to understand where the CE, CSN, SCK, MOSI and MISO signals should be put. With the help of the ATmega328 microcontroller manual[2] and the arduino UNO ports map it is possible to identify which ports connect to these signals. Thus, we connected the signal SCK to port 13, MOSI to 11, MISO to 12, CE to 7 and CSN to 8.

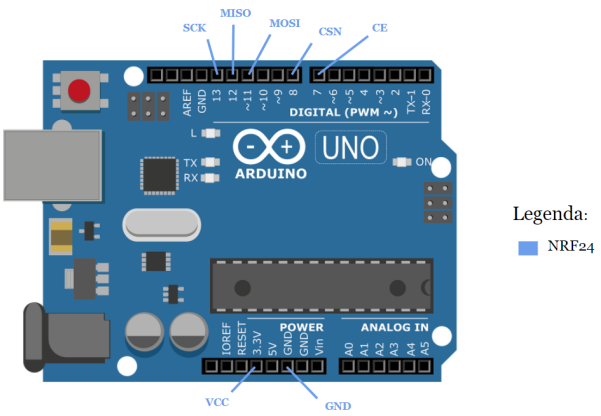


Figure 8: All the connections made on arduino UNO

6.5 Configuration Guide of the ground control station

The ground control station was assembled to serve as a bridge between the quadcopter and the simulation platform. It is able to receive/send data to the simulation platform and receive/send commands to the quadcopter. To fulfill this needs, software was developed.
In terms of libraries, we only used the SPI and RF24 libraries. Then we elaborated a file named ”EstacaoDeControlo.ino” which encapsulates all the functionalities of the components mentioned. In this file:

- It was created a radio object with the ports declared in the assembly of the ground control station station and repeated the creation of the data structures myData and myCommand (present the same role as in the Quadcopter.ino);
- As in the configuration of the quadcopter, the radio setup was done, namely in which pipes the information should be read and written.

- Once the setup is complete, the data coming from the quadcopter is ready to be sent to the simulation platform through the serial port. When the ground control station receives the commands from the platform, it sends them to the real quadcopter to execute.

6.6 Modes available

After mounting and configuring the quadcopter and the ground control station, it is possible to test the various functionalities. Before showing how to run this, is important to notice that the project is equipped with three modes that are distinct from one another and allow to test different system features.

Quadcopter Movements - show the different movements that the virtual quadcopter can assume. Unlike other modes, this mode does not use the data provided by the real quadcopter sensors and therefore does not require the ground control station or real quadcopter. The movements available are: **Vertical Take Off**, **Vertical Land**, **Move Forward**, **Move BackWards** and **Rotate**.
All these movements are used in other modes. The difference of this mode to the others is that in this the movements are given by the user in the simulation platform, while in the other modes are provoked when there are changes in the variables altitude, latitude, longitude and heading;

Sensors and Commands - allows the virtual quadcopter to move based on external data (gps, sonar and MPU6050), provided by the sensors. Once the data is received, it is processed and transformed by the simulation platform. When one of the attributes changes, the virtual quadcopter moves based on the variation felt. Besides moving the virtual quadcopter in the simulation platform based on external data, this mode also contemplates the possibility of sending commands from the platform to the real quadcopter. These commands let you tell the actual UAV what maneuver to perform. All the commands are available except the ENGINE command which is explained in the next mode;

Engines - demonstrate that it is possible to extract real data and use it in the simulation platform while the quadcopter is flying. Only use the sonar data since the rest have already been used in the previous mode. It distinguishes itself from the other modes, by making use of ESCs and engines of the real quadcopter. The user is allowed to send ENGINE-type commands. When these are sent, the ESCs send pulses that allow the engines to rotate.

In each mode it is necessary to send to the quadcopter and ground control station the files with the respective modes, that is, load the quadcopter.ino and EstacaoDeControlo.ino files of sensors mode. The same happens with the engines mode. In order to use the quadcopter movements mode it is not necessary to load any file on the controller boards of the real quadcopter and the ground control station since it does not use external data.

6.7 How to run the real quadcopter with the simulation Platform

With all these aspects explored, we are ready to test the interaction between the quadcopter and simulation platform. Steps to test the integration:

- Open the project on the Visual Studio tool, start the flight simulator (with the location ”Whidbey I NAS”) and the initiate the agentService;
- After running the project the following panel is showed.

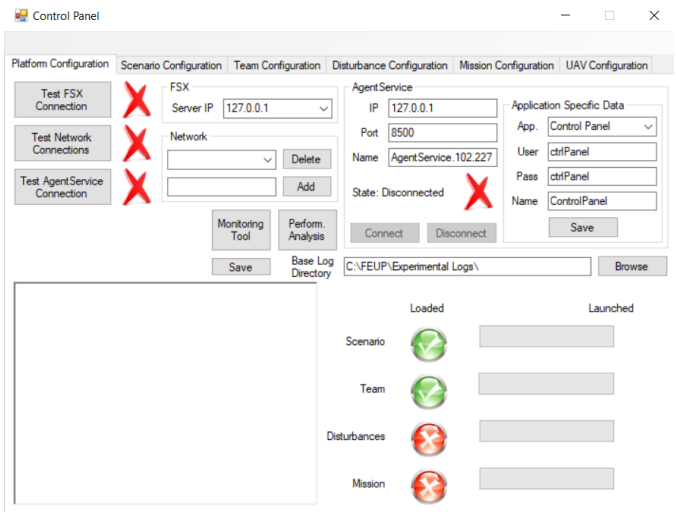


Figure 9: Initial Panel

- Press the ”Test FSX Connection”, ”Test AgentService Connection” and click on ”Connect” so the the agentService and flight simulator can connect to the project;
- Go to the Panel ”UAV Configuration” and select the mode, the port, if you want to test UAVs with/without communication (the communication is between the virtual quadcopters) and the most important, pick the agent that will serve as a digital twin (virtual agent that will move based on the readings extracted from the real UAV). Then press ”Connect Serial Port” so the simulation platform can connect to the ground control station;

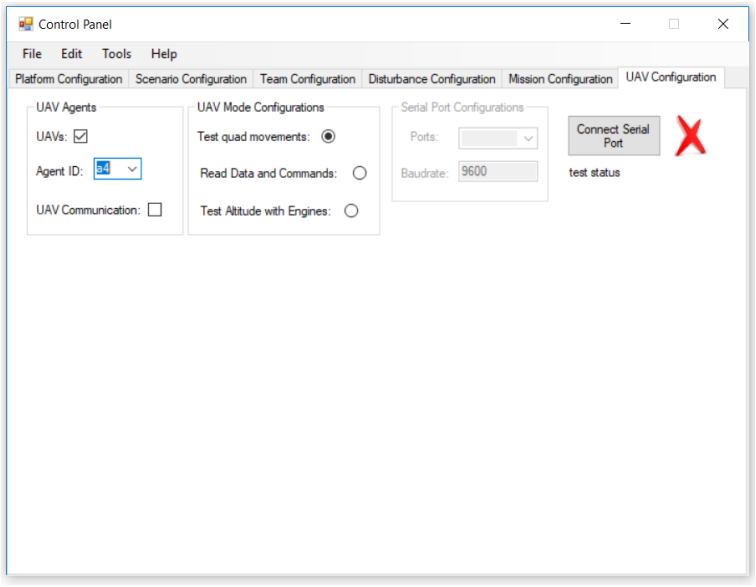


Figure 10: UAV Configuration Panel

- Once the UAV configuration is done, go to the Team Configuration Panel. Launch the vehicles (re-

member that one of them will be the digital twin). If you want to test the communication between virtual quadcopters then send more than one vehicle;

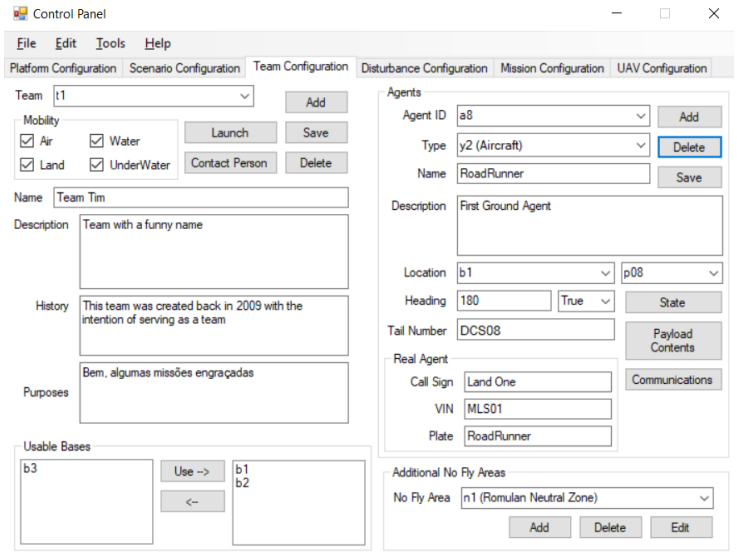


Figure 11: Team Configuration Panel

- Depending on the mode chosen, you can do different tests. The movements mode is very simple and only changes the virtual quadcopter position. If the sensors mode is chosen, you can move the real quadcopter (with your hand) up, down, front, back and rotate and you will notice that the virtual quadcopter will do the same movements with some delay (which is normal in these kind of vehicles). You can also send all kind of commands except the engine.
  If the engines mode is chosen, you can only experiment vertical movements because only altitude data is being received. You can also send the command Engine. When this is sent the real quadcopter engines rotate for 150ms. The engines rotation time is low to prevent the ESCs overheating;

References

[1] [Brooking, 2015] Brooking, J. (2015). Project YMFC-AL - the arduino auto-level quadcopter. Disponvel em http://www:brokking:net/ymfc-al_main:html (consultado a 20 de Maio de 2018).

[2] [ATmega328, 2014] Microchip. ATmega328, 2014. Available at: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf (consulted at 18 de Maro de 2018)

[3] [Arduino Pinout UNO, 2014] Marcus Jenkins. Arduino Pinout Diagrams, 2014. Available at: http://marcusjenkins.com/wp-content/uploads/2014/06/ARDUINO_V2.pdf (consulted at 18 de Maro de 2018)

[4] [ATmega2560, 2014] Microchip. ATmega2560, 2014. Available at: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf

[5] [Arduino Pinout MEGA, 2014] Marcus Jenkins. Arduino Pinout Diagrams, 2014. Available at: http://marcusjenkins.com/wp-content/uploads/2014/06/mega2.pdf

[6] [RC Groups, 2014] RC Groups. Props direction, 2014. Available at: https://www.rcgroups.com/forums/showthread.php?2097554-Props-direction

[7] [NATO, 2015] NATO (2015). Standardisation Agreement (STANAG) 4586: Standard Interfaces of UAV Control System (UCS) for NATO UAV Interoperability. NATO Standardization Agency (NSA), page 114.

[8] [Austin, 2010] Austin, R. (2010). Unmanned Aircraft Systems - Human Factors in Aviation, volume The Deployment Of UAV Systems of Aerospace Series, pages 245280. Wiley.