Controlo de um UAV para Aquisição de Dados em Explorações Agrícolas

— Versão Final —

Por

João Afonso Sousa Martins

Orientador: Pedro Miguel Mestre Alves da Silva

Co-orientador: Carlos Manuel José Alves Serôdio

Dissertação submetida à
UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO
para obtenção do grau de
MESTRE
em Engenharia Electrotécnica e de Computadores, de acordo com o disposto no
DR – I série– N°151, Decreto-Lei n.º 115/2013 de 7 de Agosto e no
Regulamento de Estudos Conducente ao Grau de Mestre da UTAD
DR, 2.ª série – N.º 133 de 13 de Julho de 2016

Controlo de um UAV para Aquisição de Dados em Explorações Agrícolas

— Versão Provisória —

Por

João Afonso Sousa Martins

Orientador: Pedro Miguel Mestre Alves da Silva

Co-orientador: Carlos Manuel José Alves Serôdio

Dissertação submetida à
UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO
para obtenção do grau de
MESTRE
em Engenharia Electrotécnica e de Computadores, de acordo com o disposto no
DR – I série– N°151, Decreto-Lei n.º 115/2013 de 7 de Agosto e no
Regulamento de Estudos Conducente ao Grau de Mestre da UTAD
DR, 2.ª série – N.º 133 de 13 de Julho de 2016

Orientação Científica :

Pedro Miguel Mestre Alves da Silva

Professor Auxiliar do
Departamento de Engenharias da Escola de Ciências e Tecnologia0
Universidade de Trás-os-Montes e Alto Douro

Carlos Manuel José Alves Serôdio

Professor Associado com Agregação do
Departamento de Electrotécnica da Escola de Ciências e Tecnologia
Universidade de Trás-os-Montes e Alto Douro

"Aqueles que se sentem satisfeitos sentam-se e nada fazem. Os insatisfeitos são os únicos benfeitores do mundo."

Walter S. Landor

Resumo — Ao longo da evolução humana, a agricultura sempre desempenhou um papel fundamental. Na sequência da Revolução Industrial, também nesta atividade se assistiu a uma progressiva mecanização de processos, com uma crescente substituição de trabalho manual por trabalho com recurso a máquinas. Apesar de fortemente mecanizada, mesmo atualmente a atividade agrícola continua fortemente dependente de uma série de fatores associados ao solo e ao clima (fatores edafoclimáticos), sendo estes fundamentais para o sucesso da atividade e práticas agrícolas. Assim, resulta necessário o conhecimento e análise contínua destes parâmetros, por forma a conhecer a sua variabilidade e antecipar dessa forma o impacto que essas variações possam ter sobre a atividade agrícola, a curto, médio e longo prazo. Neste contexto, recentemente começaram a ser desenvolvidos projetos tendentes a automatizar a prática agrícola, desta forma tornando-a mais cómoda e eficiente. Este projeto consistiu na elaboração de uma interface que servisse de estação de controlo em solo com o objetivo de controlar à distância um veículo aéreo não tripulado *Quadrotor*) de modo a proporcionar avanços na produtividade agrícola, garantir uma maior qualidade dos alimentos uma vez que os fatores edafoclimáticos que condicionam a sua produção são fortemente monitorizados e finalmente, proporcionar a sustentabilidade futura desta atividade. O veículo usado recolherá dados edafoclimáticos através de sensores. Para concretizar este projeto foram definidos vários objetivos, nomeadamente, a identificação do controlador aéreo mais adequado face à interface e ao objetivo final (desenvolvimento de uma interface que torne o *Quadrotor* autónomo e que este consiga recolher dados edafoclimáticos dentro de uma certa área agrícola), e a definição do protocolo de comunicação mais adequado. Adicionalmente, pretendeu-se compreender e utilizar de forma otimizada todas as ações que o *Quadrotor* permite realizar, tais como a execução de comandos enviados pela interface.
Palavras-Chave: Protocolo *Mavlink*, Veículo Aéreo não Tripulado, *Quadrotor*, *ArduPilotMega*, Interface em *Java*, Interface em *Android*, Estação de Controlo em Solo.

Abstract — Throughout human evolution, agriculture has always played a key role. Following the Industrial Revolution, also a progressive mechanization of processes took place in this activity, with a growing replacement of manual labor for labor using machines. Although strongly mechanized, even today agricultural activity remains strongly dependent on a series of factors associated with soil and climate (edaphoclimatic factors), which are fundamental to the success of agricultural activity and practices. Thus, the knowledge and continuous analysis of these parameters it is necessary, in order to know its variability and anticipate the impact that these variations can have on the agricultural activity, in the short, medium and long term. In this context, projects have recently begun to be developed to automate agricultural practice, thus making it more convenient and efficient. One of these projects consisted in the elaboration of an interface that served as a ground control station with the objective of remote control of an unmanned aerial vehicle (*Quadrotor*) in order to provide advances in agricultural productivity, ensure a higher quality of food since the soil and climate factors that condition their production are closely monitored and finally, provide the future sustainability of this activity. This vehicle will collect edaphoclimatic data using sensors. To accomplish this project, several goals were defined, namely, the identification of the most appropriate air microcontroller in view of the interface and the final objective (development of an interface that makes *Quadrotor* autonomous and it is able to collect edaphoclimatic data within a certain agricultural area), and the definition of the most adequate communication protocol. Additionally, it was intended to understand and optimize all the actions that the *Quadrotor* can perform, such as the execution of commands sent by the interface.

Key Words: *Mavlink* protocol, Unmanned Aerial Vehicle, *Quadrotor*, *ArduPilot-Mega*, *Java* Interface, *Android* Interface, Ground Control Station.

Agradecimentos

Ao longo da realização desta dissertação contei com o apoio, conselhos e colaboração de inúmeras pessoas, que desta forma contribuiram para a sua concretização, e às quais gostaria de deixar um especial agradecimento.

Ao Professor Doutor Pedro Miguel Mestre Alves da Silva, Professor Auxiliar do Departamento de Engenharias da Universidade de Trás-os-Montes e Alto Douro, orientador desta dissertação, pela motivação, sugestões, ideias inovadoras e orientações.

Ao Professor Doutor Carlos Manuel José Alves Serôdio, Professor Associado com Agregação do Departamento de Engenharias da Universidade de Trás-os-Montes e Alto Douro, co-orientador desta dissertação, pelas seus conselhos, observações e orientações.

A todos os meus colegas do Mestrado em Engenharia Electrotécnica e de Computadores da Universidade de Trás-os-Montes.

Aos meus Pais, pelo voto de confiança depositado e por me terem tentado ajudar sempre que possível. Quero deixar também um especial agradecimento à restante família.

À minha amiga Joana Lopes Martins, Licenciada em Genética e Biotecnologia e futura Mestre em Biologia Funcional e Biotecnologia de Plantas, por toda a ajuda e disponibilidade que teve para comigo ao longo do ano, principalmente nos conselhos

na escrita desta dissertação, sem os quais não teria conseguido.
À minha namorada Mara Daniela Martins Lopes, pelo tempo, motivação, conselhos e toda a atenção despendida.
A todos, um grande e sincero obrigado!

UTAD, João Afonso Sousa Martins
Vila Real, 15 de Maio de 2018

Índice geral

Índice de tabelas

Índice de figuras

Glossário, acrónimos e abreviaturas

Lista de acrónimos

Sigla	**Expansão**
API	*Application Programming Interface*
APM	*ArduPilot Mega*
DC	*Direct Current*
ESC	*Electronic speed control*
GCS	*Ground Control Station* (Estação de Controlo em Solo)
GPS	*Global Position System* (Sistema de Posicionamento Global)
HALE	*High Altitude and Long Endurance* (Alta Altitude e Longa Duração)
LQ	*Linear Quadratic*
MALE	*Medium-Altitude Long-Endurance* (Altitude Média e Longa Duração)
MAV	*Micro Air Vehicle*
PID	*Proporcional Integral Derivativo*
SBC	*Single Board Computer*

Sigla	**Expansão**
UAV	*Unmanned Aerial Vehicle* (Veículo aéreo não tripulado)

1 Introdução

1.1 Motivação

A agricultura é o nome dado a uma atividade humana, praticada desde o neolítico. Através dela é efetuado o cultivo de plantas com a finalidade de obter alimentos, produzir medicamentos, ferramentas, etc. Paralelamente à evolução do Homem, esta atividade sofreu também várias evoluções sendo cada vez mais visível a substituição do trabalho manual pela maquinaria própria. As condições edafoclimáticas são um dos aspecto-chave para o sucesso desta atividade. Para que uma exploração agrícola seja viável, é necessário realizar uma análise exaustiva de vários parâmetros edafoclimáticos, de modo a prever a sua viabilidade a longo prazo [15].

A utilização de veículos aéreos não tripulados na agrícultura trouxeram algumas vantagens, sendo elas[15]:

- A precisão com que se consegue monitorizar grandes áreas agrícolas.
- A sua monitorização consiste em voos periódicos, o que permite obter uma monitorização durante todo o período de produção.

- A possibilidade de estar equipado com sensores que recolhem dados edafoclimáticos provenientes das culturas agrícolas ou até a captação de imagens ou vídeos, adicionando um aspeto visual à monitorização.

- A sua utilização reflete uma redução de custos nos equipamentos e mão-de-obra. Para além disso, verificam-se ganhos significativos na eficiência de execução das tarefas que a prática agrícola exige.

Tendo estes fatores em conta, no caso deste trabalho foi elaborado um *software* com o objetivo de comandar autonomamente um *Quadrotor* até ao terreno agrícola desejado (dentro de um determinado alcance), equipado com vários sensores que recolhem os principais dados edafoclimáticos considerados fulcrais para a situação prática em questão. Os capítulos que se seguem retratam todo o conhecimento de base necessário sobre o *Quadrotor*, assim como os métodos usados para a concretização deste projeto.

1.2 Objetivos

O principal objetivo desta dissertação consistiu no desenvolvimento de uma estação de controlo em solo para monitorizar e comandar um veículo aéreo não tripulado, nomeadamente, um *Quadrotor*. Devido à instabilidade que a sua natureza apresenta e à dificuldade em o controlar, especialmente em condições de exterior, o *Quadrotor* revela-se um verdadeiro desafio para quem o utiliza como objeto de estudo em sistemas autônomos.

Deste modo, foram estabelecidas vários objetivos secundários para assim atingir o objetivo principal definido nesta dissertação:

1. Estudo e compreensão do protocolo de comunicação *Mavlink*.

2. Identificação de um modelo de *Quadrotor* associado à dinâmica do sistema.

3. Desenvolvimento de uma interface com o objetivo de realizar uma comunicação, através de um canal próprio de telemetria (usa um *Open Source Firmware* onde é possível trabalhar em parceria com o protocolo de comunicação *Mavlink*), entre o veículo e a estação de controlo em solo, de modo a conseguir controlar o *Quadrotor*.

4. Identificação de possíveis otimizações futuras ao projeto.

1.3 Organização da Dissertação

O uso de um veículo aéreo não tripulado na atividade agrícola tráz inúmeras vantagens como a possibilidade de monitorizar grandes áreas, uma maior comodidade e facilidade na recolha de dados edafoclimáticos que condicionam a produção agrícola e finalmente, a redução da mão-de-obra necessária.

Esta dissertação está organizada em 5 capítulos: Introdução, Enquadramento Tecnológico, Conceção e Implementação, Resultados e Conclusão e Trabalho Futuro.

No primeiro capítulo "Introdução"é abordada a motivação, ou seja, o problema em causa e de que forma se pretendia resolvê-lo. Para além disso também foram apresentados os objetivos definidos e organização geral da dissertação.

No segundo capítulo "Enquadramento Tecnológico"é apresentado o estado da arte, nomeadamente a História do *Quadrotor*, projetos elaborados, tipos de UAV, o porquê do uso do *Quadrotor* neste projeto, etc. Também são abordadas as principais vantagens e desvantagens dos motores destinados a UAV e como estes funcionam. Posteriormente, é dado a conhecer os tipos e o modo de funcionamento dos Sistemas de Posicionamento e Navegação destinados aos UAV. Por fim, é explanado o modo de funcionamento do protocolo de comunicação usado entre o microcontrolador (modelo APM 2.5) do *Quadrotor* e o GCS (*Ground Control Station*) (protocolo *Mavlink*) e desta forma justificada a sua importância na comunicação e controlo da aeronave.

No terceiro capítulo "Conceção e Implementação"é apresentado todo o planeamento

do sistema proposto, bem como as soluções usadas para a sua implementação. Os resultados obtidos com a implementação do trabalho elaborado são descritos no quarto capítulo "Resultados".

Finalmente, no último capítulo desta dissertação "Conclusão e Trabalho Futuro"é feita uma conclusão geral, bem como uma reflexão sobre as possíveis otimizações futuras relativamente ao trabalho elaborado.

2 Enquadramento Tecnológico

O presente capítulo apresenta todo o estado da arte referente ao tema dos *Quadrotor*. Para tal foi feita uma exaustiva pesquisa bibliográfica. Nesse sentido, ao longo deste capítulo foram abordados inúmeros temas, tais como a história do *Quadrotor* e apresentados alguns projetos académicos e comerciais. Foi também explanado exaustivamente o conceito de Unmanned Aerial Vehicle (UAV), os movimentos básicos de um *Quadrotor* e feita uma abordagem ao sistema de posicionamento e navegação, comumente usado neste tipo de veículos. Finalmente, foi explanado detalhadamente o protocolo de comunicação *Mavlink*, onde é explicado com detalhe em quê que consiste e o seu modo de funcionamento.

2.1 História do *Quadrotor*

O desejo de voar, partilhado pela Humanidade há tempos imemoriais, foi concretizado há cerca de um século. Neste período de tempo, foram entretanto desenvolvidos vários modelos de aeronaves que, apesar da grande maioria delas se ter revelado incapaz de voar, contribuiram para o desenvolvimento da aeronáutica. Desta forma, surgiram as primeiras aeronaves com capacidade de transporte aéreo de um piloto

que as controlava.

2.1.1 *Wright Flyer*

Em 1902, os irmãos Wright desenvolveram o primeiro planador. Após este grande sucesso, foram mais longe e concretizaram o que muitos pensavam impossível, isto é, inventar o primeiro avião. Este processo refletiu uma enorme complexidade devido à necessidade da criação de um sistema de propulsão [1].

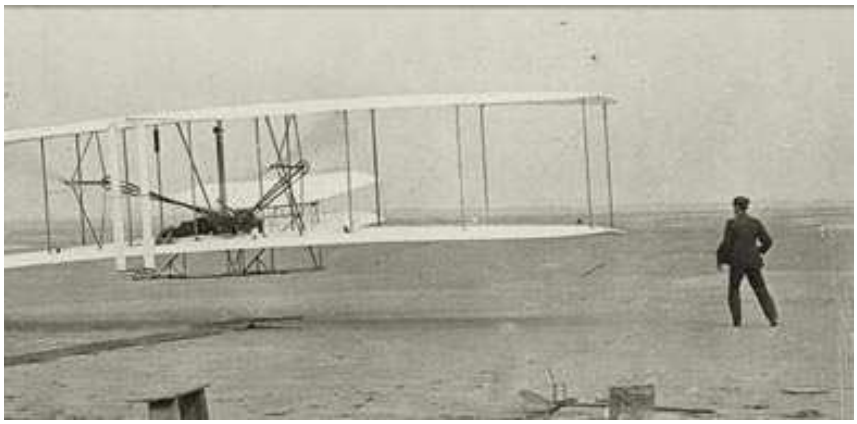

Figura 2.1 – *Wright Flyer*, 1902 [1].

Os irmãos estimaram um peso total que envolvia o peso do avião juntamente com o peso do piloto. Com base nessa estimativa, eles calcularam os requisitos de potência, impulso e velocidade para assim conseguir reunir as caracteristicas do motor para este efeito. A 17 de dezembro de 1903, Wilbur e Orville Wright fizeram quatro breves voos com sua primeira aeronave motorizada, registando a sua invenção como o primeiro avião de sucesso (Figura 2.1) [1].

2.1.2 *Gyroplane*

Em 1907, Charles Richet e os irmãos Breguet conseguiram concretizar o seu sonho de voar, deixando assim, a sua marca na história da Humanidade como sendo os criadores da primeira aeronave de asas rotativas designada de *Gyroplane* (Figura 2.2). Esta aeronave pesava cerca de 578 Kg e possuía 4 motores controlados por uma alavanca mecânica. Apesar de tudo, este projeto revelou-se um fracasso devido à difícil pilotagem, à sua instabilidade e ao limite de altitude pois apenas conseguia elevar-se 1,5 m acima do solo [31, 34].

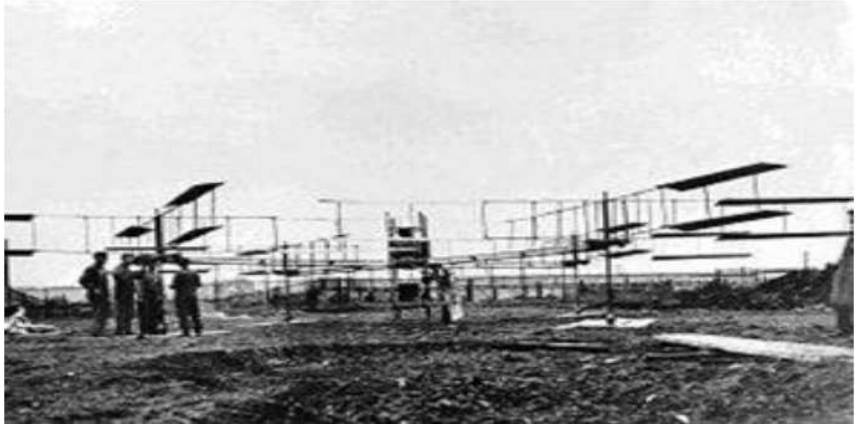

Figura 2.2 – *Gyroplane*, 1907 [31].

Os irmãos Breguet criaram ainda outras aeronaves com o objetivo de concretizarem um voo vertical, tendo deste modo, contribuído para o desenvolvimento do helicóptero convencional [31].

2.1.3 *Aeronave de Etienne Oemichen*

Em 1920, o Engenheiro Etienne Oemichen desenvolveu várias aeronaves que possuiam asas rotativas. Após Etienne ter realizado vários projetos que se revelaram um fracasso conseguiu desenvolver, mais precisamente em 1922, uma aeronave com boa controlabilidade e estabilidade denominada de *Oemichen No.2.* (Figura 2.3). A aeronave possuia 4 motores e 8 propulsores. Após alguns testes, esta aeronave

conseguiu manter-se no ar alguns minutos precorrendo cerca de 1 Km de distância [31, 34].

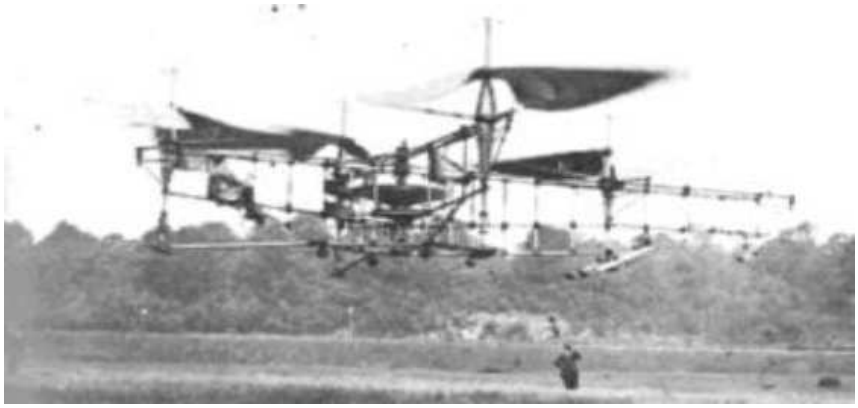

Figura 2.3 – *Aeronave criada por Etienne Oemichen*, 1920 [31].

2.1.4 *Flying Octopus*

Em 1921, Georges Bothezat construiu um dos maiores helicópteros registados na altura sob contrato com o Exército dos EUA. A aeronave concebida por Bothezat consistia numa aeronave constituída por 4 rotores (cada rotor estava situado em cada extremidade da sua estrutura em forma de uma cruz). No âmbito de auxiliar o controlo deste veículo, Georges Bothezat pensou em colocar outro conjunto de 4 rotores fazendo com que a aeronave em questão possui-se um total de 8 rotores em ação [31, 34].

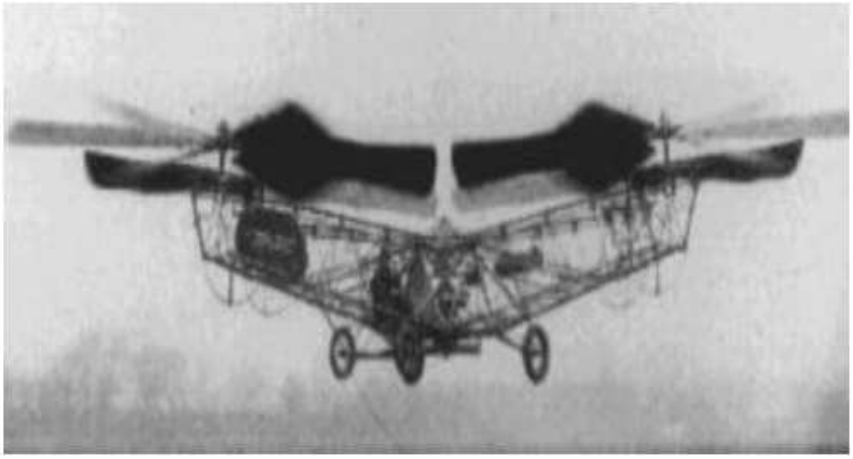

Figura 2.4 – *Aeronave Flying Octopus*, 1922 [31, 34].

Em 1922, o *Quadrotor* denominado de "Flying Octopus"(Figura 2.4) apesar de ter registado voos com sucesso, obteve sempre baixas altitudes e baixas velocidades de deslocamento. Devido aos resultados que este veículo revelou e aos altos custos financeiros que o projeto exigia levaram a que tivesse sido cancelado [31, 34].

2.1.5 *Convertawings*

O *Flying Octopus* e outras tentativas de conceção de um veículo multirotor com uma performance satisfatória, revelaram que este tipo de aeronaves eram de controlo complexo, o que levou os investigadores a focaram-se no helicóptero convencional, com apenas um rotor. Em 1956, a partir do conceito da aeronave de Oemichen e Bothezat surgiu um protótipo designado de *Convertawings Model A* (Figura 2.5), que executou vários voos com sucesso [31, 34].

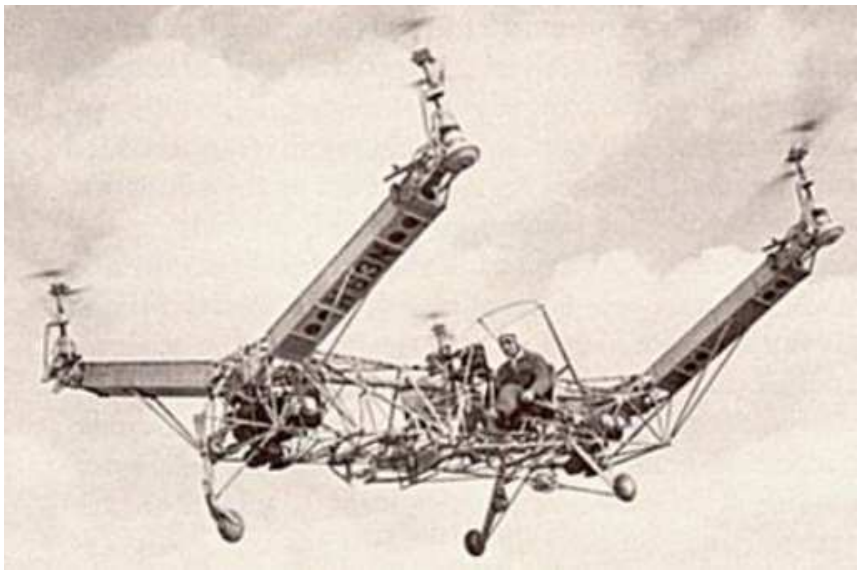

Figura 2.5 – Protótipo *Convertawings Model A*, 1956 [31].

2.1.6 *Curtiss X-19*

Em 1963, a aeronave *Curtiss X-19* (Figura 2.6) realizou o seu primeiro voo. Este veículo de 4 rotores e com descolagem na vertical, foi contruido com o objetivo de transportar passageiros. No entanto, o projeto revelou-se pouco funcional devido ao lento mecanismo de transição entre o voo na vertical e o voo horizontal [31, 34].

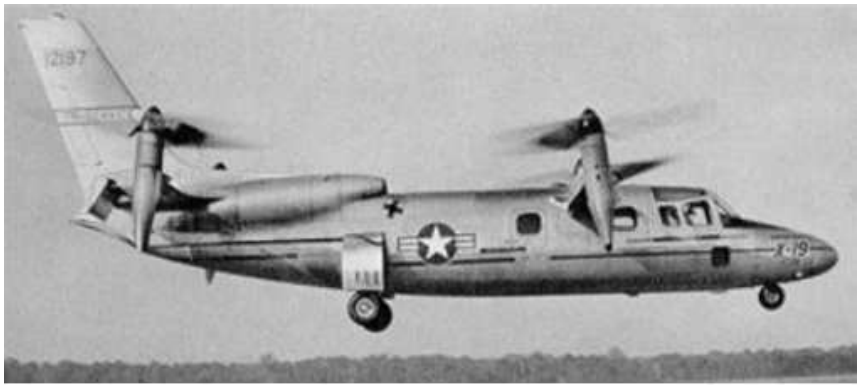

Figura 2.6 – Primeiro voo da aeronave *Curtiss X-19*, 1963 [31].

2.1.7 *Bell X-22*

Em 1966, registou-se o primeiro voo de duas aeronaves, denominadas por *Bell X-22* (Figura 2.7) [31, 34], encomendadas pela marinha dos Estados Unidados da América à empresa *Bell Helicopter*. As aeronaves desejadas tinham de possuir a capacidade de descolagem vertical. A transição entre o voo vertical e o voo horizontal desta aeronave foi efetuada com sucesso, no entanto, o projeto foi cancelado por não atingir a velocidade máxima exigida de 525 Km/h [31].

Figura 2.7 – Primeiro voo da aeronave *Bell X-22*, 1966 [31].

2.1.8 *Quad Tilt Rotor X-19*

Mais uma vez, em 2005, a marinha solicitou à empresa *Bell Helicopter* em parceria com a *Boeing*, o desenvolvimento de uma nova aeronave, denominada *Quad Tilt Rotor X-19* (Figura 2.8). Esta aeronave devia ser capaz de obter velocidades de voo elevadas, capacidade de suportar carga elevada e apresentar voo de longo alcance [31, 34].

Figura 2.8 – Aeronave *Quad Tilt Rotor X-19*, 2005 [34].

Mais recentemente, a empresa Moller desenvolveu várias aeronaves com o objetivo de criar uma alternativa viável ao automóvel, ou seja, um veículo aéreo capaz de transportar pessoas, podendo realizar o voo sobre as rodovias. A última criação desta empresa, o *M400 Skycar* (Figura 2.9), consistiu numa aeronave com quatro rotores canalizados. Este avião, aterrava e levantava na vertical e conseguia transportar quatro passageiros, com uma velocidade máxima de 445 km/h. Apesar disso, esta empresa luta ainda atualmente contra vários problemas de validação para que este projeto se torne comercial [34].

Figura 2.9 – Aeronave *M400 Skycar*[34].

Após analisar toda esta linha evolutiva, é possível concluir que os veículos aéreos não tripulados de asas rotativas (helicópteros) apresentam vantagens consideráveis sobre o avião de asa fixa, quando ambos são sujeitos à mesma aplicação.

Um helicóptero durante o voo não necessita de manter uma velocidade frontal para se sustentar no ar e consegue realizar mudanças de direção em áreas congestionadas que não podem ser realizadas pelo raio da curva de movimento de um avião. Para além disso, a capacidade de pairar sobre uma localização aumenta a facilidade de vários sensores recolherem dados sobre uma determinada área e de navegação [30].

Uma desvantagem apresentada pelo helicóptero tradicional consiste na sua complexidade mecânica, pois é necessário um rotor central na sua constituição para que seja possível uma variação na sua elevação, ou seja, para que este consiga aterrar e/ou descolar. Para além disso, os helicópteros tradicionais possuem um rotor vertical na cauda para compensar o binário (força que age num objeto fazendo com que o mesmo gire) de reação na fuselagem causada pelo rotor principal. O rotor da cauda do helicóptero tradicional introduz uma complexidade adicional, uma vez que esta cauda se estende para além do disco do rotor, aumentando a área necessária para uma operação segura [30].

O *Quadrotor* supera estas desvantagens do helicóptero tradicional sem comprometer as vantagens de uma aeronave de asa rotativa. Para além disso, este tipo de aeronave não necessita de um rotor na cauda, uma vez que o par de rotores de contrarrotação equilibra o binário. Adicionalmente, o conjunto de carga permite que os quatro rotores com pequenos eixos substituam o eixo central articulado do helicóptero de rotor simples [30].

O *Quadrotor* possui também um impulso único que é usado para compensar o peso total e não para contrariar o binário. Para além disso, a força proveniente dos 4 rotors é totalmente utilizada para transportar a carga [30].

2.2 Projetos Académicos

Os UAV (Veículo aéreo não tripulado) deixaram de ser algo ocultado e exclusivo do exército e começaram a ser elaborados inúmeros projetos académicos e comercializados vários tipos destas aeronaves. Os projetos desenvolvidos e os diversos tipos de UAV criados serão abordados de seguida.

2.2.1 Universidade de Stanford

A Universidade de Stanford iniciou, em 1991, um projeto intitulado de *Mesicopter* (Figura 2.10). Este projeto tinha como objetivo desenvolver um *Quadrotor* numa escala de um centímetro, de modo a conseguir estudar a atmosfera do planeta e realizar a exploração espacial de outros planetas [31, 34].

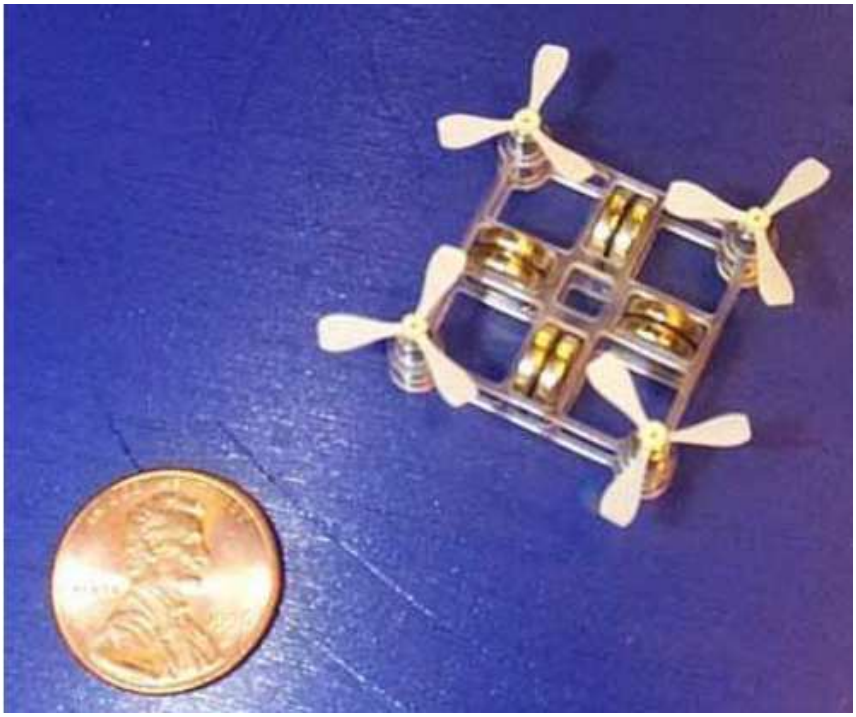

Figura 2.10 – Projecto Mesicopter, 1991 [31].

Na linha do projeto apresentado anteriormente surgiu mais recentemente o projeto

STARMAC, que consistia numa plataforma de testes para diversos UAV com o objetivo de demonstrar multifuncionalidades numa plataforma a atuar no mundo real. O *STARMAC* era composto por seis veículos equipados com equipamento de medida e computação, que permitiam a sua operação completamente projeto surgiram dois veículos distintos, o *STARMAC* I desenvolvido em 2004 e posteriormente, o *STARMAC* II desenvolvido em 2007[31, 34].

2.2.2 Escola Politécnica Federal de Lausanne (EPFL)

Na EPFL, o estudante de Doutoramento Samir Bouabdallah desenvolveu o *Quadrotor OS4* (Figura 2.11) cujo o objetivo era desenvolver e testar métodos de controlo e a resolução de problemas de peso e autonomia associados ao *Quadrotor*.

Figura 2.11 – *Quadrotor* OS4 [31, 34].

Para o controlo do veículo foram utilizadas várias técnicas de controlo da altitude. A primeira a ser testada foi a técnica de controlo de Lyapunov. Posteriormente, foram testadas duas outras técnicas de controlo do *Quadrotor* (PID, Proporcional Integral Derivativo e LQ, do inglês *Linear Quadratic*). O objetivo foi determinar qual destas

seria a mais robusta para o controlo da altitude do veículo. Posteriormente, foram também testadas as técnicas de controlo da altitude *Sliding-mode* e *Backstepping* com e sem ação integral. Finalmente, após a comparação de todas as técnicas testadas concluiu-se que a técnica *Backstepping* era a mais robusta para o controlo da altitude e posição do veículo, permitindo um voo autónomo[31, 34].

2.2.3 Universidade Nacional da Austrália (Australia National University - ANU)

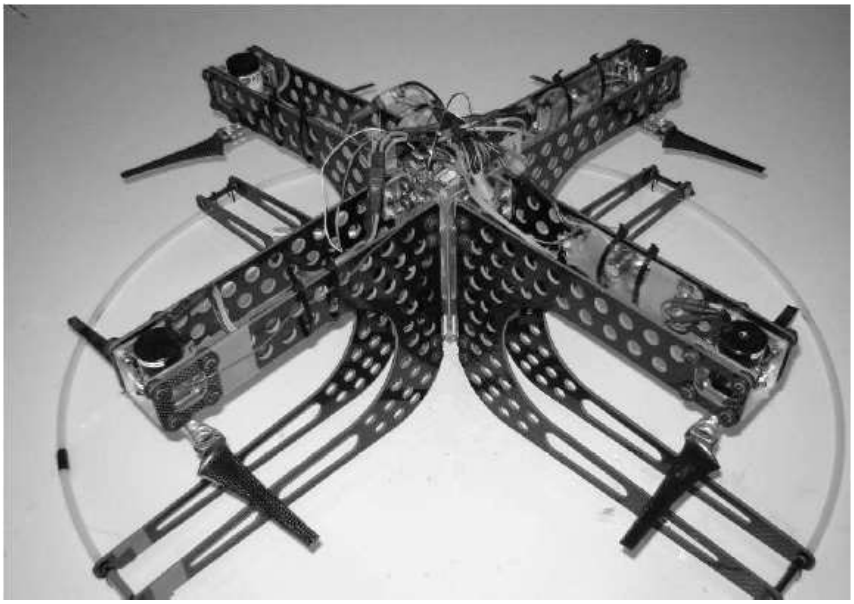

Figura 2.12 – *MARK II X4 Flyer* [34].

Em 2002, a Universidade Nacional da Austrália desenvolveu um protótipo de *Quadrotor*, o *MARK I*, que apresentava um *design* inovador com os motores em posição invertida. Mais tarde, surgiu um *upgrade* deste modelo, o *MARK II X4 Flyer* (Figura 2.12). Similarmente ao modelo utilizado no presente projeto, também este modelo apresentava um sistema de comunicação entre o *Quadrotor* e uma estação de controlo em solo (GCS, do inglês *Ground Control Station*) baseada numa ligação via *Bluetooth*. Os dados provenientes do *Quadrotor* por telemetria eram visualizados mediante a utilização de um GCS [34].

2.2.4 Universidade da Pensilvânia

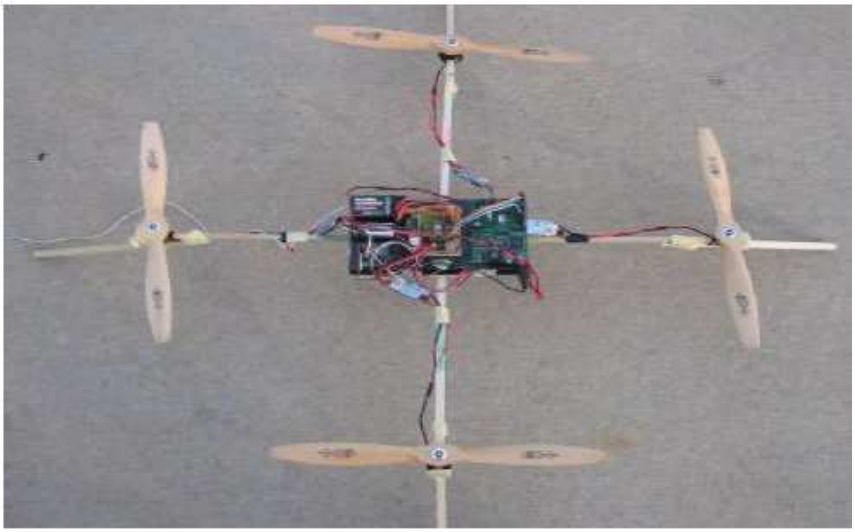

Figura 2.13 – *Quadrotor* da Universidade da Pensilvânia [31, 34].

Em 2005, o estudante de mestrado Scott D. Hanford apresentou a dissertação intitulada *"A small semi-autonomous rotary-wing unmanned air vehicle"*, que tinha como objetivo o desenvolvimento de um veículo aéreo de quatro motores semi-autónomo usando materiais de baixo custo como sensores, motores e controladores. O veículo (Figura 2.13) possuía um transmissor e um recetor de rádio para efetuar a comunicação. Para a medição da atitude do veículo (ângulo em relação ao solo) foram utilizados giroscópios e acelerómetros de baixo custo [31].

2.2.5 Universidade de Oakland

Figura 2.14 – Quadrotor *Microraptor* [31, 34].

Em 2008, a Universidade de Oakland desenvolveu um *Quadrotor* designado *Microraptor* (Figura 2.14), com o objetivo de participar na competição *AUVSI* (*Association for Unmanned Vehicles Systems International*) por uma equipa de alunos, tendo ficado em quinto lugar na competição [31].

O *Microraptor* consistia em dois subsistemas, o veículo aéreo e a base que era responsável pela telemetria e o controlo manual do veículo. A estrutura do veículo aéreo foi concebida em fibra de carbono com um peso aproximado de 1400 g e uma autonomia de voo de 15 minutos. A posição do veículo era obtida através de um módulo de GPS que também fornecia a velocidade em voo. No controlo do veículo existiam dois microcontroladores, um dedicado à aquisição da telemetria, aquisição dos dados de GPS, cálculo da altitude e comunicação com a base. O outro microcontrolador estava encarregue de receber os dados do IMU (*Inertial measurement unit*) e efetuar o controlo de posição e estabilidade. A técnica de controlo utilizada para a monitorização deste veículo foi o PID (*Proportional Integral Derivative*) [31].

2.3 Exemplos de Projetos Comerciais

Para além de todos estes projetos universitários, também existe uma grande variadade comercial deste tipo de veículos o que enfatiza ainda mais o interesse neste tipo de projetos.

2.3.1 Empresa Dranganflyer Innovations Inc.

A empresa *Dranganflyer Innovations Inc.* é uma empresa especializada em veículos controlados remotamente. Fundada em 1998 por *Zenon* e *Christine Dragan*, esta empresa desenvolveu um *Quadrotor* com elevado desempenho e possuidor de uma câmara embutida, o que possibilita o registo de imagens durante o voo.

Figura 2.15 – Modelo de UAV *Draganflyer X4 Helicopter* [2].

Este *Quadrotor* designado *Draganflyer* (Figura 2.15) foi contruído com o objetivo de conseguir realizar operações de salvamento, vigilância, inspeção, observação aérea, observação táctica, captação de imagens e vídeos aéreos [31]. Hoje em dia esta empresa continua a investir no desenvolvimento de novos veículos, promovendo cada vez mais o modelo *Dranganflyer* [31].

O modelo *Draganflyer X4 Helicopter* é um UAV com uma estrutura em fibra de carbono, nylon e alumínio. Este tipo de *Quadrotor* é constituido por vários componentes, sendo eles: quatro motores, controladores dos motores, fonte de energia, três acelerómetros, três giroscópios, um sensor de pressão barométrica, comunicações com uma frequência de 2,4 GHz para o controlo e telemetria com antenas omnidirecionais e comunicação de vídeo a uma frequência de 5,8 GHz com antenas unidireccionais e o equipamento de controlo. Este modelo pesa cerca de 680 g e tem uma capacidade de *payload* de 250 g. [31].

Figura 2.16 – Modelo de UAV *Draganflyer X8 Helicopter* [3].

O modelo *Draganflyer X8 Helicopter* (Figura 2.15) é um melhoramento do modelo apresentado anteriormente. Houve um aumento do seu peso passando a um total

de 1700 g e possui a capacidade de *payload* de aproximadamente 1000 g. Tem uma autonomia de voo de 20 minutos [31].

2.3.2 Empresa Microdrones

Figura 2.17 – Modelo de UAV MD4 - 200 [4].

A empresa Microdrones criou dois UAV designados de MD4-200 (Figura 2.17) e MD4-1000 (Figura 2.18). Estas aeronaves possuíam GPS e transmitiam os dados provenientes da telemetria para uma base com a opção de também serem gravados num cartão de memória. Para além disso, o modelo MD4-200 apresentava a capacidade de *payload* de 200 g e o modelo MD4-1000 continha uma capacidade de *payload* de 1000 g [31].

Figura 2.18 – Modelo de UAV MD4 - 1000 [5].

Todos os modelos descritos acima estavam equipados com sistemas de segurança que preveniam o voo com bateria fraca ou quando existia perda de sinal entre a base e o veículo (sistemas de *failsafe*).

2.4 Unmanned Aerial Vehicles (UAV)

2.4.1 Tipos de UAV

O *Quadrotor* foi escolhido como a plataforma de estudo para esta dissertação, pois apresenta em geral robustez, vários modos de voo, bom equilíbrio e elevada popularidade na comunidade de UAV [28]. Existem vários tipos de UAV e estes são classificados de acordo com a sua utilização (Adaptado de [26]):

1. Altitude elevada e longa duração de voo (HALE do inglês high altitude and long endurance):

- Duração de Voo (por tarefa): cerca de 35 horas
- Comunicação: Satélite

- Peso total: cerca de 900kg
- Exemplos de modelos: Global Hawk (avião de asa fixa), Helios (avião de asa fixa), Dark Star (avião de asa fixa)

2. Altitude média e longa duração de voo (MALE do inglês medium altitude and long endurance):

- Duração de Voo (por tarefa): cerca de 35 horas
- Comunicação: Satélite
- Peso total: cerca de 200kg
- Exemplos de modelos: Predator (avião de asa fixa), Reaper (avião de asa fixa)

3. Tático:

- Duração de Voo (por tarefa): cerca de 6 horas
- Comunicação: L,C e S-Band
- Peso total: <45kg
- Exemplos de modelos: Shadow (avião de asa fixa), Hunter (avião de asa fixa), Scan Eagle (avião de asa fixa)

4. *Small*:

- Duração de Voo (por tarefa): cerca de <2 horas
- Comunicação: LOS Radio
- Peso total: <11kg
- Exemplos de modelos: Raven (avião de asa fixa), Javelin (avião de asa fixa), Wasp (avião de asa fixa), Draganflyer (*Quadrotor*)

5. *Micro aerial vehicle*:

- Duração de Voo (por tarefa): cerca de <1 horas

- Comunicação: LOS Radio ou Wifi

- Peso total: <2,27kg

- Exemplos de modelos: Black Widow (*Quadrotor*, Dragonfly (*Quadrotor*, ArduPlane (avião de asa fixa)

O modelo usado neste dissertação insere-se na categoria dos *Micro aerial vehicle*.

2.5 Movimentos básicos de um *Quadrotor*

Os Quadrotores são compostos por 6 eixos responsáveis pelo controlo do seu movimento, o qual é feito através de translações dos eixos X, Y e Z e rotações dos eixos *Roll*, *Pitch* e *Yaw* [31]. As hélices são colocadas conforme a rotação dos motores, de modo a que o *Quadrotor* consiga realizar as rotações e/ou translações desejadas. Os motores rodam em dois sentidos: no sentido dos ponteiros do relógio (CW do inglês *clockwise*) ou no sentido contrário aos ponteiros do relógio (CCW do inglês *counter clockwise*). Os braços do *Quadrotor* identificados a vermelho na Figura 2.19, identificam a parte da frente do veículo, tornando desta forma mais fácil identificar para que direção está a ocorrer o deslocamento.

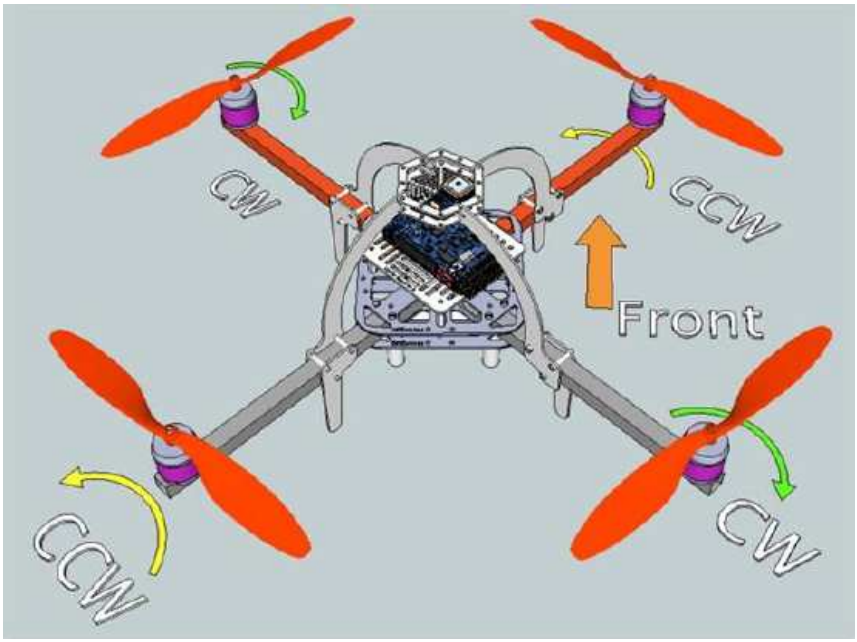

Figura 2.19 – Parte da frente do *Quadrotor* [34, 31].

Uma vez que este tipo de UAV possui 4 hélices, utiliza 1 movimento de translação (o *Throttle* (Z)) e 3 movimentos de rotação, nomeadamente o Roll (ϕ), Pitch (θ) e Yaw (ψ).

2.5.1 *Throttle* (Z)

O *Throttle* trata-se de um movimento que o *Quadrotor* faz de modo a diminuir ou aumentar a sua altitude (Figura 2.20). Para que esse movimento seja possível é necessário que os ângulos ϕ , θ e ψ se mantenham inalterados e que o aumento ou diminuição da velocidade das hélices 1, 2, 3 e 4 (respetivamente aumento ou diminuição da altitude) seja igual, de forma a que o veículo se movimente apenas ao longo do eixo do Z [31, 32].

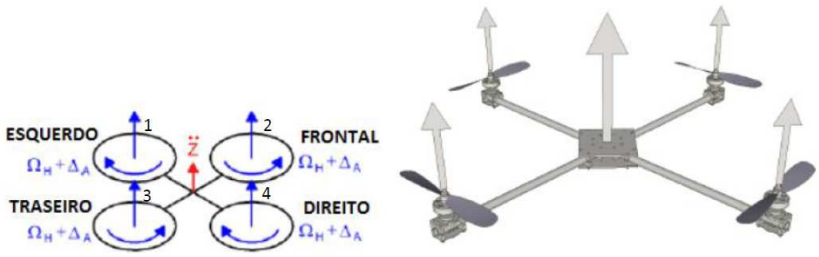

Figura 2.20 – Throttle (Adaptado de [32]).

2.5.2 Roll (ϕ)

O ângulo ϕ faz com que o *Quadrotor* consiga inclinar-se de modo a voar para a esquerda ou para a direita (Figura 2.21). Para isso, o *Throttle*, assim como os ângulos ψ e θ têm de permanecer constantes durante a rotação do *Roll*. Para que o veículo apresente um movimento no sentido do *Roll* positivo (para a direita), a velocidade das hélices 1 e 4 têm de aumentar e a velocidade das hélices 2 e 3 têm de diminuir. Contrariamente, o *Quadrotor* movimenta-se no sentido do *Roll* negativo (para a esquerda) caso a velocidade das hélices 1 e 4 diminua e velocidade das hélices 2 e 3 aumente. O aumento e a diminuição da velocidade de cada par de hélices, independentemente do movimento desejado, têm a mesma variação [32].

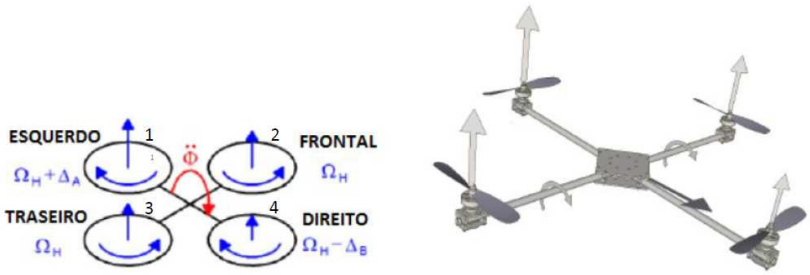

Figura 2.21 – Roll (Adaptado de [32]).

2.5.3 *Pitch* (θ)

Para que o veículo consiga avançar ou retroceder sem alterar a sua altitude, é necessário que ocorra uma rotação positiva ou negativa, respetivamente, do ângulo θ (Figura 2.22). Para isso, o *Throttle* e os ângulos ψ e ϕ têm de permanecer constantes. Similarmente ao movimento do *Roll*, se a velocidade das hélices 2 e 3 aumentar e das hélices 1 e 4 diminuir, o *Quadrotor* irá movimentar-se para a frente (*Pitch* positivo). Contrariamente, o veículo movimenta-se para trás, caso a velocidade das hélices 2 e 3 diminua e das hélices 1 e 4 aumente (*Pitch* negativo). O aumento e a diminuição da velocidade de cada par de hélices, independentemente do movimento desejado, têm a mesma variação [32].

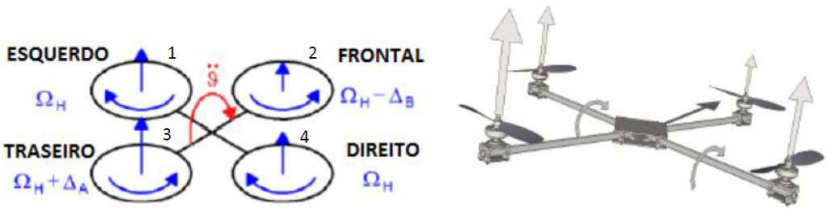

Figura 2.22 – Pitch (Adaptado de [32]).

2.5.4 *Yaw* (ψ)

O *Quadrotor* pode rodar no sentido dos ponteiros do relógio ou no sentido contrário a este, através da rotação no ângulo ψ (Figura 2.23). Para isso, o *Throttle* e os ângulos θ e ϕ têm de permanecer constantes. Similarmente às rotações nos eixos *Roll* e *Pitch*, o veículo roda no sentido *Yaw* positivo quando a velocidade das hélice 1 e 3 aumenta e das hélices 2 e 4 diminui. Contrariamente, verifica-se a rotação do *Quadrotor* no sentido *Yaw* negativo quando a velocidade das hélice 1 e 3 diminui e das hélices 2 e 4 aumenta. O aumento e a diminuição da velocidade de cada par de hélices, independentemente do movimento desejado, têm a mesma variação [32].

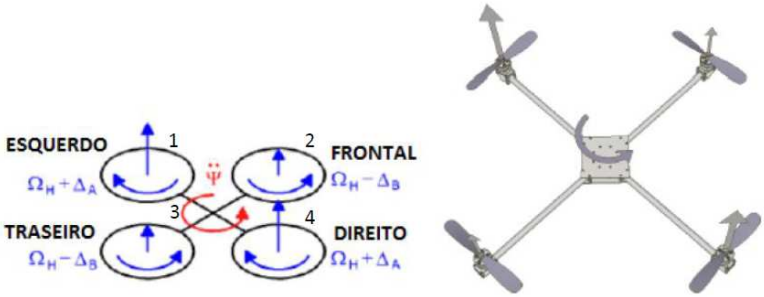

Figura 2.23 – Yaw (Adaptado de [32]).

A ação dos motores é a grande responsável pelos movimentos que o *Quadrotor* pode tomar. Desta forma, é importante tomar conhecimento quais os tipos de motores indicados para UAV e como é que este são controlados.

2.6 Princípios de Funcionamento dos Motores destinados a UAV

O controlo dos motores deve ser escolhido conforme o tipo de motores usado. Para adquirir variações de velocidade por parte do motor é usado um sinal PWM (*Pulse Width Modulation*), gerado a partir de um circuito integrado ou por um microcontrolador [23].

O sinal PWM é usado como técnica para o controlo de circuitos analógicos, utilizando tempos de trabalho (*duty cycles*) e variando, de uma forma períodica, a tensão média na carga. O sinal permite controlar a potência dos motores por completo, uma vez que o PWM consoante o *duty cycle* estipulado consegue fazer com que os motores trabalhem com uma certa percentagem de potência [23].

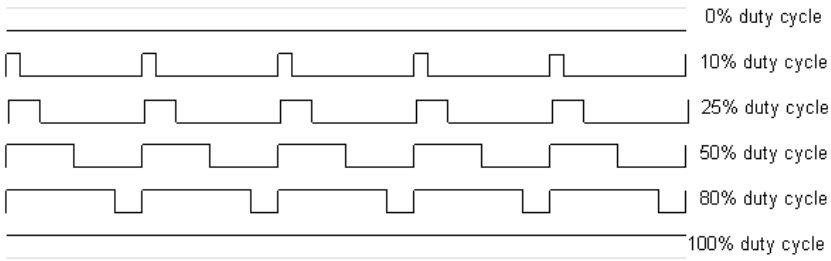

Figura 2.24 – Sinal PMW com diferentes *duty cycles* ([6]).

Deste modo, o PWM faz com que o motor seja alimentado e desligado em determinados períodos de tempo (em milissegundos). Devido à inércia do próprio motor, estas mudanças fazem com que se mova a uma velocidade constante. Deste modo, quanto maior a largura do pulso, maior será a velocidade atingida. Os tipos de motores (ambos controlados por PWM) mais utilizados no aeromodelismo são os motores DC [23].

2.6.1 Motores DC (*Direct Current Motors*)

Os motores DC (*Direct Current*), como o próprio nome indica, são motores de corrente contínua, muito usados em projetos que envolvem mini robôs. Este tipo de motor é bastante popular em mecanismos de controlo remoto, como por exemplo, em carros telecomandados. O rotor é um dispositivo rotativo que está situado no centro do motor DC e que é composto por bobinas de cabo condutoras da corrente. Essa corrente é fornecida ao rotor e a força através da qual o motor gira é proporcional à corrente fornecida pelas bobinas. Quanto maior for a tensão, maior será a corrente aplicada e mais força o motor irá desenvolver [23].

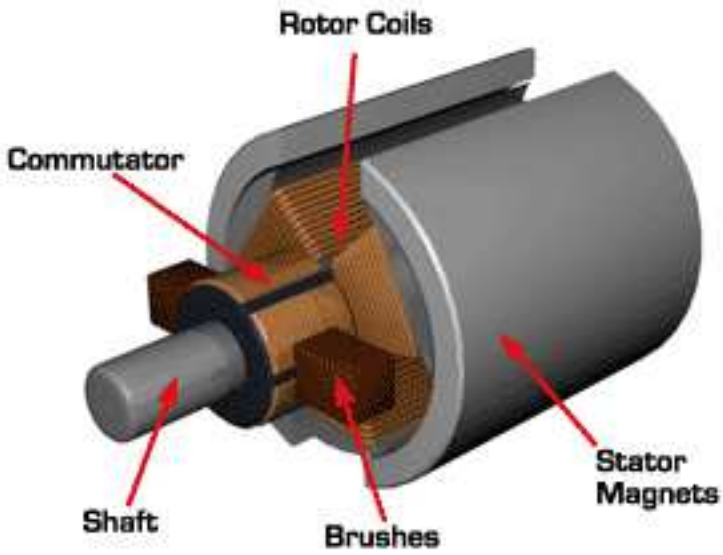

Figura 2.25 – Constituição do Motor DC ([7]).

O motor DC é composto por um eixo ligado ao rotor, sendo esta a parte do motor que gira. O estator é composto por um íman e o comutador tem a função de transferir a energia da fonte de alimentação para o rotor (Figura 2.25) [22].

2.6.2 Motores DC sem escovas (*Brushless Direct Current Motors*)

Os motores BLDC (*Brushless Direct Current Motors*) (Figura 2.26), tal como o nome sugere, não têm escovas para comutação. O estator neste tipo de motores é formado por material ferromagnético com os respetivos enrolamentos e o rotor consiste num íman permanente. É de salientar que esta disposição está invertida relativamente à do motor de íman permanente com escovas. Normalmente, nestes motores sem escovas são colocados três sensores de *Hall* no estator para detetar a posição do rotor e fazer a comutacão da alimentacão nos enrolamentos, de acordo com a informacão obtida nesses sensores [24, 22].

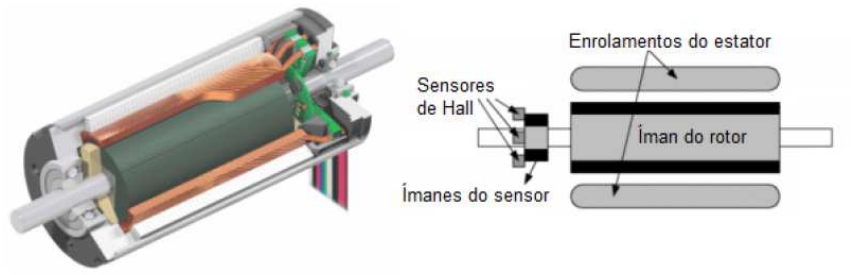

Figura 2.26 – Constituição do Motor BLDC ([7]).

Os motores BLDC devem ser comutados tão próximo quanto possível do instante em que o ângulo entre os fluxos gerados pelo estator e pelo rotor é de 90°, de modo a maximizar o binário. Geralmente, os motores BLDC industriais têm enrolamentos de três fases ligadas em estrela, constituídas por várias bobines e respetivos pólos e o rotor pode ter vários pares de pólos. Como a comutação deste motor é realizada eletronicamente, para que rode é necessário que os enrolamentos do estator sejam alimentados por uma tensão contínua com uma sequência. Assim, são criados sequencialmente pares de pólos no estator que atraem pólos opostos do rotor [24, 22].

Estes motores têm como principais vantagens um baixo custo de manutenção, boa eficiência, vida útil longa, operação silenciosa, extensa gama de velocidades e desgaste mecânico reduzido. Dado que o rotor é constituído por ímanes permanentes torna-se mais leve e, por isso, tem uma inércia inferior à dos rotores com núcleo de ferro, o que lhe confere uma boa dinâmica [24, 22].

Este motor porque necessita de um controlador eletrónico para funcionar apresenta um custo mais elevado. A elevada eficiência, tamanho reduzido e o baixo custo de manutenção torna-lo indicado para aplicações em que a fiabilidade é importante [24, 22].

2.7 Sistema de Posicionamento e Navegação nos UAV

Os UAV possuem diversos equipamentos que permitem ao utilizador uma referência espacial do veículo, através da obtenção dos dados referentes à longitude, latitude e altitude atuais. Este tipo de informação é essencial para quem deseja realizar uma navegação à distância [10, 27].

O sistema de posicionamento possibilita ainda a obtenção das coordenadas geográficas e da altitude do veículo. Para além disso, no caso da aeronave possuir uma câmara que transmite em direto, o utilizador obterá referências visuais e, ao mesmo tempo, a localização quase exata do local onde o UAV se encontra [10, 27].

Estes sistemas de posicionamento podem revelar-se mais ou menos precisos. Conforme o tipo de navegação desejado (navegação de curtas distâncias ou longas distâncias), deverá ser usado o sistema de posicionamento mais adequado [27]. Para uma navegação de curtas distâncias ou em que o utilizador tem sempre contacto visual com o veículo, não há necessidade de implementação de qualquer tipo de sistema de posicionamento [10, 27].

As forças militares, como foi referido anteriormente, desenvolveram e testaram exaustivamente inúmeros veículos aéreos não tripulados, muitas das vezes com o objetivo de salvar vidas humanas o que requeria um sistema de posicionamento com o menor erro possível (grau de exatidão elevado)[10, 27].

2.7.1 Sistema de Posicionamento e Navegação via Rádio

Este sistema de posicionamento e navegação tem como base a medição do tempo de propagação de ondas electromagnéticas, nomeadamente, ondas rádio entre o emissor (UAV) e o recetor (GCS). Existem 2 tipos de sistemas de posicionamento e navegação: por satélite e mediante bases terrestres [10, 27].

2.7.2 Navegação por Satélite

O sistema de posicionamento e navegação via Rádio permite adquirir coordenadas geográficas e altitude, através de dados de posicionamento geoespacial provenientes de satélites localizados em torno do planeta Terra. Atualmente, existe um grande investimento por parte de alguns países que promovem o desenvolvimento de diversos sistemas de navegação orientados por estes satélites [27].

O GPS (*Global Positioning System*), desenvolvido pelos Estados Unidos da América, tornou-se o primeiro sistema deste género a existir e a ser utilizado. Atualmente, estão a ser desenvolvidos outros sistemas com a mesma metodologia, como por exemplo, o GALILEO (União Europeia) e GLONASS (Federação Russa)[27].

O GPS funciona usando 3 componentes, como se pode verificar na Tabela 2.7.2:

Tabela 2.1 – Componentes do GPS[27]

Componente	Caraterísticas
Espacial	24 satélites (com relógios atómicos) e 3 de reserva (para eventuais falhas).
Controlo	5 estações espalhadas pelo planeta com o objetivo de monitorizar as posições do satélites e sincronizar os seus relógios atómicos.
Utilizador	Recetor GPS (equipado com um relógio de quartzo).

Os satélites e o recetor de GPS usado neste tipo de sistemas contêm um relógio altamente preciso. Os satélites GPS enviam sinais sob a forma de ondas electromagnéticas que contém várias informações, entre elas, a hora de emissão do sinal. Os recetores GPS recebem a informação sobre o tempo de saida do sinal do satélite e calculam a distância a que este se encontra[27]. A fórmula correspondente a esse cálculo é:

Velocidade de propagação da onda = Distância percorrida/Tempo de saída do sinal do modelo GPS até atingir o satélite

O sistema de GPS basea-se no método de triangulação (3 satélites) para determinar a posição exata do recetor de GPS no planeta Terra. A determinação da posição corresponde ao ponto de interceção do raio da localização prevista pelos 3 satélites. É ainda necessário um quarto satélite para a determinação da altitude do recetor de GPS [27]. A figura 2.27 ilustra o método de triangulação do modelo GPS:

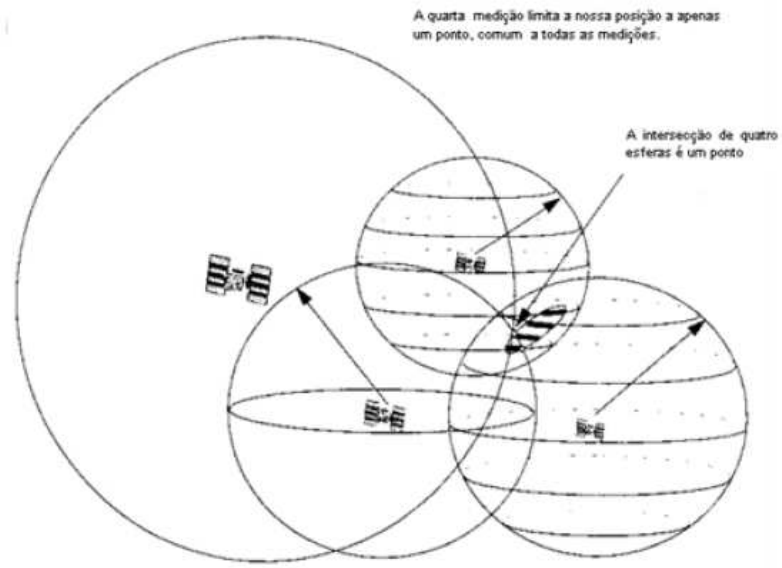

Figura 2.27 – Método de triangulação [32].

Apesar deste sistema se revelar bastante preciso no que toca à posição espacial do UAV, existem sempre erros relacionados com a sincronização dos relógios e atrasos ou desvios que o sinal electromagnético pode sofrer [27].

2.7.3 Navegação por Bases Terrestres

Como alternativa à interação com os satélites, é igualmente possível obter uma localização espacial por bases terrestres espalhadas pelo globo. Existem 3 sistemas distintos para a obtenção de uma posição espacial, sendo eles [27]:

- ADF (*Automatic Direction Finder)*

  Permite encontrar a direção relativamente a uma fonte que emite ondas de rádio. A utilização de duas fontes permite encontrar a posição precisa.

- VOR (*VHF Omnidireccional Range)*

  Uma base terrestre emite de forma constante em todas as direções ondas rádio VHF. O recetor recebe e interpreta as ondas e calcula a diferença entre sinais de forma a obter a sua posição.

- DME (*Distance Measuring equipment)*

  A emissão de ondas por um veículo atinge uma base terrestre que recebe e reenvia ondas de outras frequências (pré-estabelecidas). Através da diferença temporal, o veículo calcula a sua distância em relação à base terrestre.

2.7.4 Sistema de Navegação Inercial

O sistema de navegação inercial (Figura 2.28) consiste num outro tipo de sistema que os UAV utilizam, com o objetivo de obter as coordenadas geográficas da sua presente localização. Primeiro, utiliza a informação da posição inicial e da aceleração do veículo (obtida através de acelerómetros) para calcular a sua velocidade através de uma integração. Após calcular a velocidade do veículo é necessário saber a direção que este toma (são usados giroscópios para o cálculo da direção do UAV), para assim determinar a sua localização[27]. Devido a possíveis erros de cálculo, este sistema não é considerado muito preciso. Para contornar este problema, o giroscópio e o acelerómetro deverão trabalhar em conjunto com o GPS[27].

Figura 2.28 – Sistema de medição inercial: Giroscópio e Acelerómetro[27].

2.8 Protocolo *MavLink*

O protocolo *MavLink* é um protocolo *open source* de comunicação, destinado a pequenos veículos não tripulados. Trata-se de um protocolo de comunicação da segunda camada do modelo OSI que o *ArdupilotMega* (controlador aéreo APM) usa para conseguir comunicar com o GCS (do inglês *Ground Control Station*). Desta forma, é possível a receção e o envio bidirecional de dados entre o UAV e o GCS, tais como a indicação da localização deste via GPS, da sua velocidade em relação ao solo, etc., assim como a emissão de dados (comandos de operações) por parte do GCS, como por exemplo, Descolagem, Armar, Desarmar, *Waypoint*,etc.[26].

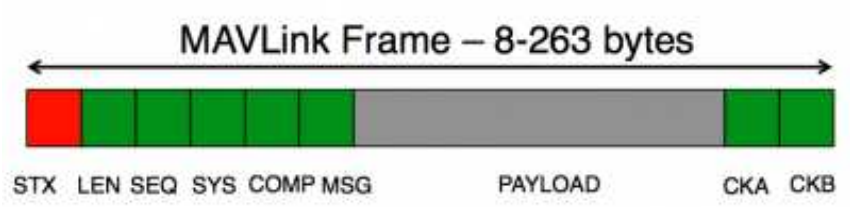

Figura 2.29 – Estrutura dos pacotes do protocolo *MavLink* [26].

Neste protocolo, os dados encontram-se sob a forma de pacotes (Figura 2.29). Uma vez enviada a mensagem a descodificar, o protocolo *Mavlink* faz uso do pacote STX (do inglês *Start of frame*) para identificar o início da mensagem enviada. Após o sinal STX ter sido analisado, é efetuada a leitura do comprimento da mensagem através do pacote LEN (comprimento do *Payload*)[26].

Uma vez concluída a leitura de todos os *bits* que compõem a mensagem, é feita a verificação do *checksum* (componentes CKA e CKB). O *checksum* consiste num método que permite detetar erros em sistemas de comunicação, ou seja, se a mensagem em questão tiver a "aprovação"deste método, o pacote é descodificado e processado.

Posteriormente, após o APM ter recebido a mensagem por parte do GCS, envia uma mensagem ACK (mensagem de confirmação da receção, do inglês *acknowledgement*) ao GCS. O APM aguarda o próximo pacote STX. Caso o *checksum* detete algum tipo de anomalia (número incorreto de *bits*) relativamente à constituição do pacote, o pacote STX é rejeitado e o APM aguarda de imediato um novo pacote [26].

Para detetar a perda de pacotes, o protocolo *Mavlink* utiliza um número de sequência em cada pacote. Caso a perda de pacotes seja significativa, deverão ser executadas ações de segurança, tais como o comando *return to launch* ou a diminuição do alcance entre o GCS e o APM [26].

A estrutura de todas as mensagens (Figura 2.29), assim como a funcionalidade de cada componente pertencente ao pacote encontra-se de seguida:

- Nome do Campo: *Start of frame* (STX)

  Valor: 0xFE

Índice: 0

Objetivo: Identifica o início da frame pertencente ao pacote.

- Nome do Campo: *Payload length* (LEN)

  Valor: 0-255

  Índice: 1

  Objetivo: Identifica o comprimento do *payload*.

- Nome do Campo: *Packet sequence* (SEQ)

  Valor: 0-255

  Índice: 2

  Objetivo: Cada componente conta a sua sequência de envio. Evita a perda de pacotes.

- Nome do Campo: *System ID* (SYS)

  Valor: 0-255

  Índice: 3

  Objetivo: Identificação dos sistemas de envio para a diferenciação dos vários sistemas presentes numa rede.

- Nome do Campo: *Component ID* (COMP)

  Valor: 0-255

  Índice: 4

  Objetivo: Identificação dos componentes de envio para a diferenciação dos diversos componentes presentes neste sistema.

- Nome do Campo: *Message ID* (MSG)

  Valor: 0-255

  Índice: 5

  Objetivo: Identificação da mensagem.

- Nome do Campo: *Payload* (Informação das mensagens)

  Valor: 0-255

  Índice: 6 até (n + 6)

  Objetivo: Conteúdo da mensagem que depende do ID da mesma.

- Nome do Campo: *Checksum* (CKA e CKB)

  Valor: ITU X.25/SAE AS-4 *hash of bytes* 1 - n+6

  Índice: (N + 7) até (N + 8)

  Objetivo: Soma de controlo que o *Mavlink* utiliza para evitar erros em sistemas de comunicação.

2.9 Mensagens do Protocolo *Mavlink*

O protocolo *Mavlink* permite estabelecer uma comunicação entre um microcontrolador aéreo (por exemplo, o APM 2.5) e um GCS. O *payload* das mensagens enviadas possui um *System ID* e um *Component ID* que diz respeito ao *Quadrotor* a controlar. O tipo de mensagens *Heartbeat* têm como principal objetivo fornecer vários dados relativos ao Quadrator ao GCS. Para além das mensagens do tipo *Heartbeat*, existem outros tipos de mensagens que podem ser utilizadas no âmbito da utilização de um protocolo *Mavlink*.

2.9.1 Mensagem *Heartbeat*

A mensagem *Heartbeat* constitui o tipo de mensagem mais importante do protocolo *Mavlink* e tem como objetivo estabelecer a conexão entre o GCS e o UAV, de forma a ser possível a emissão e receção recíproca de dados. O UAV emite várias mensagens do tipo *Heartbeat* onde está contida informação (*payload* da mensagem) relativa ao veículo. Se um certo número de mensagens *Heartbeat* for perdido, é ativado o sistema de *Failsafe*, que será abordado mais à frente. Para além disso, o *payload*

da mensagem do tipo *Heartbeat* proveniente do UAV fornece informação essencial para que o GCS consiga posteriormente emitir messagens comando para este. A informação contida no *payload* encontra-se explicada na Tabela 2.2.

Tabela 2.2 – *Payload* da mensagem *Heartbeat* (Adaptado de [18])

Informação	Objetivo
Type	Tipo de MAV (*Quadrotor,Helicopter...*)
Autopilot	Tipo de *Autopilot*
Base_mode	Modo do Sistema em Mapa de *Bits*
Custom_mode	Consiste num Mapa de *Bits* reservado para *flags* específicas do *Autopilot*
System_status	*Flag* que o sistema utiliza para dizer o seu estado
Mavlink_version	Versão de *Mavlink*

Os valores que os campos *System_status* do tipo de mensagem *Heartbeat* podem ser observados no Tabela 2.3.

Tabela 2.3 – Valores do *System_status*

Modo	Valor do *System_status*
Sistema por iniciar, estado desconhecido.	"0"
Sistema a iniciar.	"1"
Sistema a calibrar e não se encontra pronto para voar.	"2"
Sistema em *standby* e pronto para a descolagem a qualquer altura.	"3"
Sistema está ativo e pode estar a voar. Os motores estão a girar.	"4"
Sistema está num modo de voo anormal, no entanto pode estar a voar.	"5"
Sistema está num modo de voo anormal. Perdeu o controlo funcional de alguns componentes ou na sua totalidade. Sistema encontra-se em estado de alerta	"6"
Sistema iniciado irá desligar-se	"7"

2.9.2 *Set Mode*

A mensagem do tipo *Set Mode* tem como objetivo definir os vários modos, em especial os modos de voo que o UAV poderá apresentar antes e durante a missão. A informação contida no *Payload* é explicada no Tabela 2.4.

Tabela 2.4 – *Payload* da mensagem *Set Mode* (Adaptado de [18])

Informação	Objetivo
Target_System	Sistema em que é feito a mudança de modo.
Base_mode	Novo modo definido.
Custom_mode	Novo modo específico do *Autopilot*.

Os valores que os campos *Custom_mode* e *Base_mode* são iguais nas mensagens do

tipo *Heartbeat* ou nas mensagens do tipo *Set mode* e estão apresentados na Tabela 2.5

Tabela 2.5 – Valores do *Custom_mode* e *Base_mode*

Modo	Estado	Valor do Custom_mode	Valor do Base_mode
Stabilized	Armed	"0"	"217"
	Disarmed		"89"
Acro	Armed	"1"	"217"
	Disarmed		"89"
Alt_Hold	Armed	"2"	"217"
	Disarmed		"89"
Auto	Armed	"3"	"217"
	Disarmed		"89"
Guided	Armed	"4"	"217"
	Disarmed		"89"
Loiter	Armed	"5"	"217"
	Disarmed		"89"
Return_To_Land	Armed	"6"	"217"
	Disarmed		"89"
Circle	Armed	"7"	"217"
	Disarmed		"89"
Position	Armed	"8"	"217"
	Disarmed		"89"
Land	Armed	"9"	"217"
	Disarmed		"89"
Of_Loiter	Armed	"10"	"217"
	Disarmed		"89"
TOY_A	Armed	"11"	"217"
	Disarmed		"89"
TOY_M	Armed	"12"	"217"
	Disarmed		"89"
NUM_MODES	Armed	"13"	"217"
	Disarmed		"89"

2.9.3 *Command Long*

A mensagem do tipo *Command Long* tem como objetivo enviar comandos operacionais ao UAV. Os comandos podem consistir na realização de uma descolagem (*Takeoff*) e/ou aterragem (*Land*), o voo para uma determinada coordenada *Navegate to Waypoint* predefinida ou a realização de *Arm/Disarm*. A informação contida no *Payload* é explicada no Tabela 2.6.

Tabela 2.6 – *Payload* da mensagem *Command Long* (Adaptado de [18])

Informação	Objetivo
Target_System	Sistema que irá executar o comando.
Target_Component	Componente que irá executar o comando. Caso seja igual a "0"estará a englobar todos os componentes.
Command	ID do comando. Tem de estar de acordo com o ID do comando que se quer executar.
Confirmation	Caso o valor seja "0", será a primeira transmissão deste comando. Caso esteja entre "1"a "255", será a confirmação das transmissões efectuadas.
Param1	Cada um destes campos tem de estar de acordo com o comando em causa. Os parâmetros têm diferentes significados de acordo com o comando escolhido.
Param2	
Param3	
Param4	
Param5	
Param6	
Param7	

Como já foi anteriormente referido, cada comando apresenta diferentes valores de parâmetros. No caso dos comandos *Arm* ou *Disarm*, toda a estrutura do *payload*

é similar uma vez que o ID do comando é o mesmo, à exeção do valor do *Param1* que no caso de ser "1"implica que ocorra o comando "Arm"e caso seja "0"ocorre o comando "Disarm". Apesar de existirem um total de 7 parâmetros apenas o valor do primeiro parâmetro pode variar.

- Comando *Arm*:

  Target_system = "1";

  Target_Component = "0";

  Command = "400";

  Confirmation = "1";

  Param1 = "1";

- Comando *Disarm*:

  Target_system = "1";

  Target_Component = "0";

  Command = "400";

  Confirmation = "1";

  Param1 = "0";

Se avaliarmos os parâmetros do comando *Takeoff*, é apenas possível modificar o valor do Param7 que dita a altitude que o veículo irá assumir. O *Param1* retrata o valor mínimo do ângulo *Pitch* e o *Param2* e o *Param3* apresentão valores nulos. O *Param4* retrata o ângulo *Yaw* (ângulo ignorado na ausência do magnetómetro). O *param5* e *param6* correspondem, respetivamente, a latitude e a longitude a que o veículo se encontra.

- Comando *Takeoff*:

  Target_system = "1";

$Target\_Component = "0";$

$Command = "22";$

$\text{Confirmation} = "1";$

$Param7 = "\text{Altitude estipulada}";$

3 Conceção e Implementação

3.1 Conceção

O objetivo fundamental desta dissertação foi conseguir automatizar um veículo aéreo não tripulado para a recolha de dados edafoclimáticos numa certa área agrícola.

De forma a atingir o objetivo definido para esta dissertação, foi primeiramente elaborado um GCS com o intuito de estabelecer comunicação à distância com o controlador aéreo (Figura 3.1).

O controlador aéreo de voo constitui um componente do UAV, cuja principal função é controlar as suas ações e recolher dados através de diversos equipamentos anexos ao veículo tais como motores, sensores, GPS, entre outras tecnologias.

Os dados enviados pelo GCS ao controlador aéreo correspondem a mensagens de comando cujo objetivo é serem executadas corretamente pelo UAV. No caso do GCS enviar uma mensagem a solicitar informação ao controlador aéreo, por exemplo, informação sobre o estado em que se encontra o UAV, este envia-lhe uma mensagem com a informação em questão. Isto demonstra a extrema importância da comunicação bidirecional de modo a conseguir explorar todas a potencialidades na

utilização de um UAV.

Adicionalmente ao sistema descrito anteriormente, foi incorporado no UAV um SBC (*Single Board Computer*) cuja função é responder aos pedidos (servidor) que o GCS desenvolvido emite. Deste modo, promove a funcionalidade da interface (cliente).

O SBC também permite a incorporação de uma série de ferramentas adicionais tais como sensores, câmara de vídeo, etc. Adicionalmente, a comunicação é diretamente trocada ou enviada entre o GCS e o SBC, através de um sistema cliente-servidor via *socket*. A comunicação entre o controlador aéreo e o SBC foi efetuada mediante um sistema de telemetria.

Sempre que o GCS envia uma mensagem esta é emitida através de um *socket* diretamente ao SBC (o sistema *socket* usado será explanado mais detalhadamente). O SBC recebe e classifica a mensagem proveniente do GCS e envia-a ao controlador aéreo. O controlador aéreo envia uma mensagem ACK ao SBC, que por sua vez, será mais uma vez enviada ao GCS relatando deste modo o *status* da mensagem inicialmente enviada pelo GCS.

A par da interface criada, foi também elaborado um *software* para a plataforma *Android* de modo a facilitar os testes em campo aberto (bloco SPM do diagrama 3.1). Este *Software* também desempenha o papel de cliente, ou seja, sempre que necessita que o *Quadrotor* execute uma determinada ação de acordo com os comandos estipulados, envia-lhe o respetivo comando.

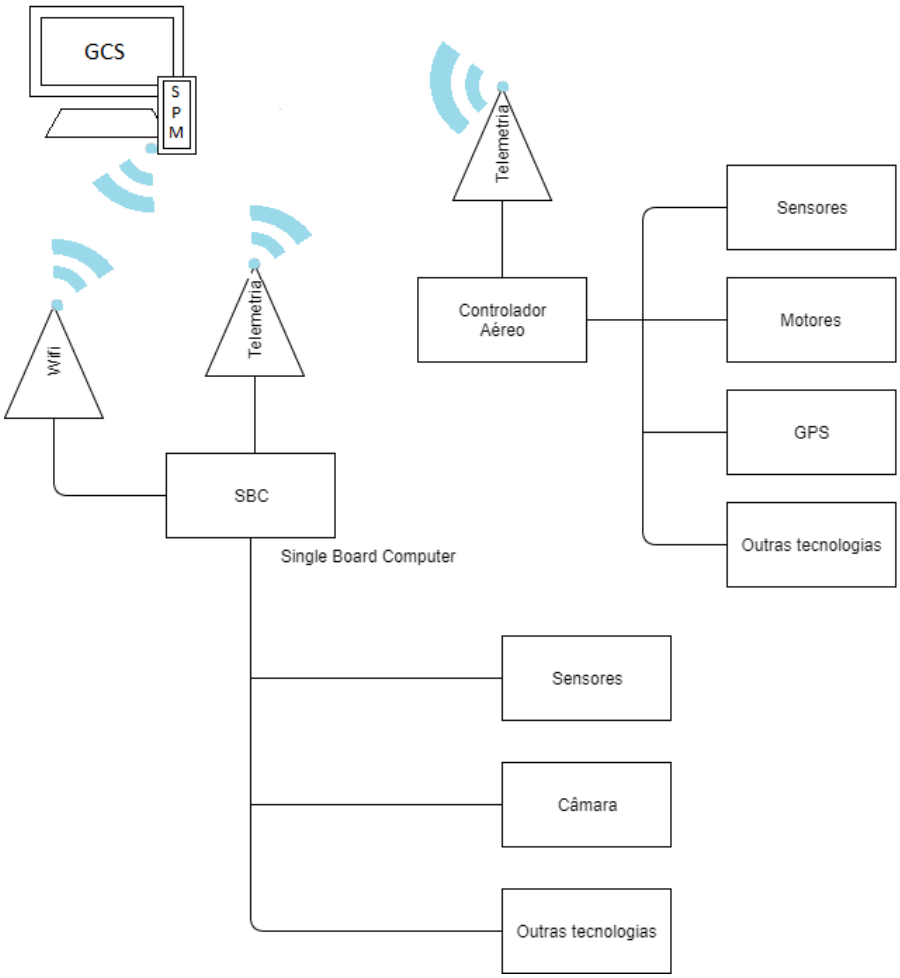

Figura 3.1 – Diagrama de Blocos - Conceção do trabalho

3.2 Materiais e Métodos

Inicialmente, foi usado um *Quadrotor* desenvolvido por um aluno de Licenciatura em Engenharia Electrotécnica e de Computadores da UTAD, que gentilmente o disponibilizou como objeto de estudo desta dissertação. Este *Quadrotor* revelou ser incapaz de concluir todos os testes finais, uma vez que revelou uma anomalia num dos ESC, tornando deste modo impossivel garantir a segurança no seu voo. Como alternativa foi utilizado um novo modelo mais atual. Este *Quadrotor* era composto por vários equipamentos, nomeadamente:

- 1 x APM 2.5 *Flight Control Board* (baseado no ATMega 2560);
- 1 x *Lipo battery* (*Turnigy* 4S 6000 mAh *Nano-tech*);
- 4 x Motores *brushless* (motores DJI 2212/920 kVs);
- 4 x Hélices de passo 94x45 (auto riscantes, 2x CW + 2x CCW);
- 4 x *Electronic speed control* 20A Skywalker;
- 1 x *Frame* f450 - Estrutura em cruz (corpo do *Quadrotor*);
- 1 x Modulo telemetria 433 MHZ (Crius 3DR Radio 433 MHZ Telemetry Module);
- 1 x GPS (Cirocomm 580 GPS module);
- 1 x Placa antivibratoria (Apropriada para o Microcontrolador em questão);
- 1 x Módulo de Potência 3DR;
- 1 x Buzzer de alarme de bateria;

3.2.1 *ArdupilotMega*

O ArdupilotMega, também conhecido por APM, é um controlador destinado a UAV, que permite o controlo autónomo nos mais diversos tipos de veículos aéreos não tripulados (Figura 3.2).

O nome dado a este sistema deriva do termo *Arduino*, que consiste numa plataforma computacional de código aberto, composta por uma placa com um microcontrolador ATMEGA 2560, que é responsável por processar os dados obtidos dos sensores e atuar sobre os motores [20].

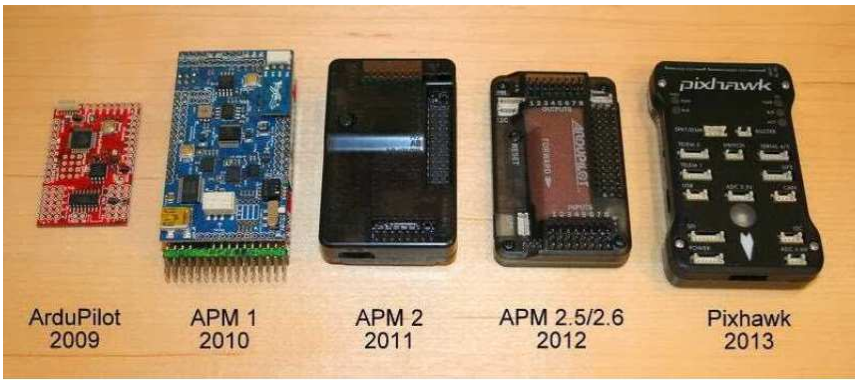

Figura 3.2 – Linha de evolução de microcontroladores destinados a UAV (Adaptado de [11])

3.2.2 Constituição do APM 2.5

Todos os controladores de voo possuem um conjunto de instruções específicas, incorporam uma solução num único circuito integrado revelando facilidades de manipulação de *bits*, apresentam portas de entrada/saida, entre outras caraterísticas. Os componentes principais pertencentes ao modelo APM 2.5 são apresentados no Tabela 3.1:

Tabela 3.1 – Constituição do modelo APM 2.5 (Adaptado de [20, 21])

Componente	Função
Microcontrolador ATMega2560 (ou ArduinoMega)	Microcontrolador de 8 bits que tem como objetivo tornar o sistema mais simples e serve como ponto de conexão para outros componentes
Conetores de expansão	Permite a adição de novos componentes e GPS
Giroscópio	Dispositivo utilizado para medição e manutenção da orientação de determinado objeto através do seu movimento angular
Acelerómetro	Dispositivo utilizado para medir as forças que atuam nas mesmas direções em que o objeto sofre alteração da sua aceleração
Sensor de pressão barométrica	Utilizado essencialmente para medir a pressão, sendo muitas vezes utilizado indiretamente em várias aplicações (por exemplo, para medir a altitude)

O modelo APM 2.5 apresenta um total de 11 pins, entre os quais 9 são analógicos (A0, A1, A2, A3, A4, A5, A6, A7 e A8) e 2 digitais (A12 e A13) (Figura 3.3). Os pins analógicos funcionam como entradas analógicas de um modo geral. A tensão máxima aceite por estes pins é de 5 V e normalmente são usados para entradas de *airspeed* e de sonar [11].

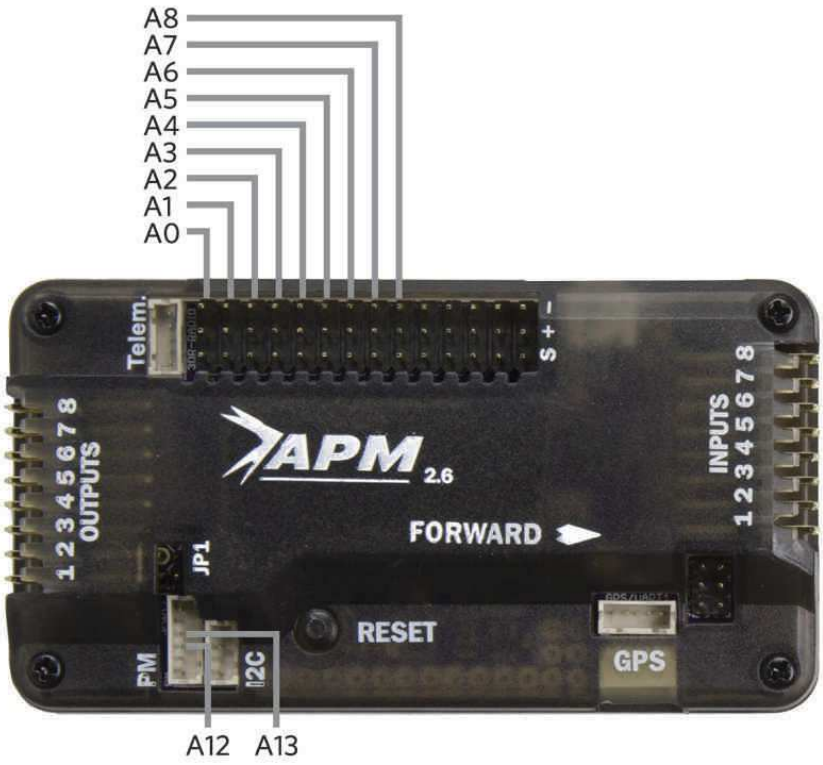

Figura 3.3 – Pinos de entrada do APM 2.5 [11]

O *3DR power brick*, cuja função é alimentar o APM 2.5, liga-se aos pins digitais, o pin A12 (corrente) e o pin A13 (tensão) [11].

Em relação aos LED (*Light-emitting diodes*) incorporados no APM indicam-nos de imediato em que estado se encontra o microcontrolador, sendo por isso uma mais valia para analisar de uma forma rápida o estado do sistema. A Figura 3.4 mostra todos os LED constituintes do modelo APM 2.5 [12].

A Tabela 3.2 apresenta de que forma os LED reagem face aos vários estados em que o sistema do APM 2.5 se pode encontrar.

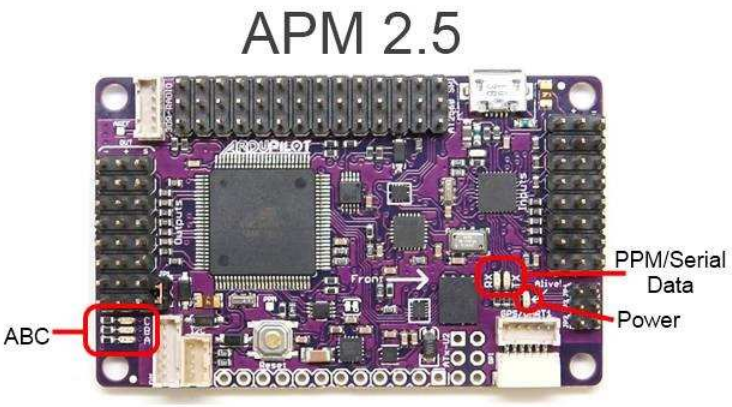

Figura 3.4 – LED presentes no APM 2.5 [12]

Tabela 3.2 – Comportamento dos LED presentes no APM 2.5 (Adaptado de [12])

Estado do LED	Estado do Sistema
LED *Power* acende	O APM 2.5 está a ser alimentado
LED A com luz fixa (luz vermelha)	Motores em estado *Armed* que irão girar à medida que o valor do *Throttle* subir
LED A pisca 1 vez (luz vermelha)	Motores em estado *Disarmed*
LED A pisca 2 vezes (luz vermelha)	Motores em estado *Disarmed* e não podem ser armados devido a uma falha no *pre-arm checks*
LED B (luz amarela) intermitente com LED A (luz azul)	Durante a calibração dos ESC (*Electronic speed control*)
LED C com luz fixa (luz azul)	GPS a funcionar sem qualquer tipo de anormalidade e em estado *lock*
LED C pisca 1 vez (luz azul)	GPS a funcionar sem qualquer tipo de anormalidade mas ainda sem atingir o estado *lock*
LED C pisca 2 vezes (luz azul)	GPS não está a funcionar corretamente ou não está conetado
LED *PPM/Serial Data* acende de um modo intermitente	Detetada receção/emissão de informação

3.2.3 Módulo de Potência 3DR

O módulo de Potência 3DR (Figura 3.5), quando ligado à *lipo battery*, faz com que o APM 2.5 receba uma tensão e uma corrente estáveis (5,37 V e 2.25 A, respetivamente). O 3DR PM permite também uma monitorização da corrente e da tensão provenientes da bateria usada e deste modo faz com que ocorra um comando do tipo *Return to Launch* quando a bateria apresenta valores de tensão baixos [14].

Figura 3.5 – Módulo de Potência 3DR [8]

Uma vez que a bateria usada consiste numa bateria de polímetro de lítio de 4 células (4S), o módulo de Potência 3DR aceita uma tensão de entrada máxima de 18 V e uma corrente máxima de entrada de 90 A.

3.2.4 Bateria de Polímetro de Lítio (*Turnigy* 4S 6000 *mAh Nano-tech*)

Existem vários tipos de baterias recarregáveis. Nesta dissertação, foi utilizada uma bateria de polímero de lítio. Este tipo de baterias consiste numa bateria recarregável, feita com a mesma tecnologia das baterias de lítio Íon, contudo, apresenta a grande vantagem de não conter líquido ou gel como eletrólito [29]. As principais vantagens e desvantagens deste tipo de baterias são (Adaptado de [29]):

Vantagens:

1. Alta densidade energética;
2. Baterias pequenas em relação à quantidade de carga;
3. Pouco peso;
4. Mais resistentes a defeitos de efeito memória e descarga a vazio;
5. Sem risco de descarga total;

Desvantagens:

1. Baixa durabilidade;
2. Tempo de vida útil curto, mesmo sem uso;
3. Complexidade na fabricação de carregadores;
4. Necessitam de cuidados extraordinários no seu manuseio;
5. O Lítio é um material instável no ambiente e caso haja lixiviação pode explodir.

Para o correto carregamento de uma bateria de polímetro de lítio, o primeiro cuidado a ter será colocá-la dentro de um saco protetor (*lipo-safe*) (Figura 3.6) previamente a iniciar o processo de carregamento. Deste modo, caso ocorra algum incidente

consegue-se minimizar os danos no ambiente envolvente. Para além disso, durante o carregamento nunca se deve deixar a bateria sem qualquer vigilância e esta deverá encontrar-se sobre superficies e/ou materiais não inflamáveis.

Figura 3.6 – Lipo-safe sobre superficie não inflamável

O carregador da bateria alimenta a bateria com uma tensão de 14,8 V. O *Turnigy ACCUCELL-6* apresenta vários modos possíveis de carregamento. No entanto, o modo de carregamento mais aconselhado é o modo *Lipo Balance*, pois este faz o controlo da tensão presente nas baterias de lítio, ou seja, o processador interno do carregador monitoriza a tensão de cada célula e a corrente que a alimenta, controlando deste modo a tensão.

3.2.5 Motores *brushless* DJI 2212/920 kVs

O modelo dos motores escolhidos para o trabalho em questão foi o modelo modelo DJI E300 *Multirotor* (motor *brushless* 2212/920 kVs) da empresa DJI (*Dà-Jiãng Innovations*) (Figura 3.7). Este modelo constituía a escolha mais adequada uma vez que se tratava de um motor DC do tipo *Brushless* (ver página 30). As principais caraterísticas encontram-se detalhadas na Tabela 3.3 [25].

Figura 3.7 – Motor *brushless* DJI 2212/920 kVs

Tabela 3.3 – Especificações referentes ao modelo DJI E300 *Multirotor* (Adaptado de [25])

Modelo	DJI E300 *Multirotor*
Rpm/*Volt* (rotações por minute/Tensão)	920 rpm/v
Corrente Máxima	30 A
Potência Máxima	370W
Eixo	4 mm
Peso	53 g
Profundidade do buraco dos parafusos	49 mm
Dimensão	28 x 24 mm

3.2.6 GPS (*U-blox NEO-6M GPS module*)

O sistema escolhido de posicionamento global (GPS, do inglês *Global Positioning System*) foi o U-blox NEO-6M GPS *module*. Este modelo pode ser usado em qualquer *ArdupilotMega* acima da versão 2 e como qualquer outro equipamento GPS tem como objetivo fornecer informação sobre a sua posição atual, com o menor erro possível [33].

3.3 Implementação

Para controlar o *Quadrotor* em questão foi elaborado um sistema de cliente-servidor via *socket*, onde foi feita a distinção de vários clientes (Interface em *Java* (GCS), *Command Line* e Interface elaborada em *Android Smartphone*).

Qualquer cliente consegue estabelecer e manter contacto com o servidor, entidade essa que atende aos pedidos dos clientes ligando-se, por sua vez, ao controlador aéreo do *Quadrotor*.

Inicialmente, o servidor fica à espera que o cliente estabeleça uma conexão, escutando um *socket* que está ligado à sua porta série. Caso o endereço ip e o número da porta estejam corretos, é enviado um pedido por parte do cliente para se ligar ao servidor. Se a ligação for aceite, o servidor recebe um novo *socket* que irá servir de apoio à comunicação entre o servidor e o cliente.

Foi usado um *Raspberry* Pi que ao ser ligado deu início a um programa *Java* desenvolvido para desempenhar a função de servidor.

Este servidor possui um endereço ip fixo e um número de porta previemente estipulado. O endereço ip do *Raspberry* Pi foi definido para um endereço ip estático, de modo a tornar possível a conexão do cliente, uma vez que o endereço ip não varia após a ligação efectuada.

O servidor é composto pelas seguintes classes: *drone*, *Messages_Response*, *SocketMessage*, *SocketServer*,*SocketStruct*, missão e por último, ponto.

A interface gráfica elaborada em *Java* representa um dos possíveis clientes que se podem ligar ao servidor. Este cliente contém as seguintes classes: *interfacemode*, *flightPlanner*, missão, ponto, *MySocket*, *SocketMessage*. O projeto deste cliente também contém um ficheiro *HTML* demoninado de *map*. Este ficheiro, assim como as classes, os atributos, os métodos e as bibliotecas usados serão explicados detalhadamente mais à frente (ver de página 66 a 69).

Foi ainda desenvolvida uma outra interface na plataforma *Android* de modo a facilitar os testes finais e um protocolo de mensagens de forma a tornar o sistema mais robusto (Figura 3.15).

Previemente ao desenvolvimento das interfaces apresentadas anteriormente, foram estudados exaustivamente os vários tipos de mensagens pretencentes ao protocolo *Mavlink*. A base de todas as mensagens foi a mensagem do tipo *Heartbeat*. Esta mensagem fornece várias informações, entre elas, o estado em que o *Quadrotor* se encontra, o seu modo de voo, entre outras. Os próximos subcapítulos apresentam o estudo das características referentes à mensagem do tipo *Heartbeat* e é feita a abordagem das mensagens de comando aceites.

3.3.1 *Heartbeat*

Com a ajuda de um dispositivo *Data Communication Analyzer*, nomeadamente AE-5105, foi possível observar a estrutura que os pacotes da mensagem *Heartbeat* assumem quando ocorre a troca de mensagens entre o GCS e o *Quadrotor*. A Figura 3.8 mostra uma pequena sequência de uma mensagem proveniente do *Quadrotor* e recebida pelo dispositivo.

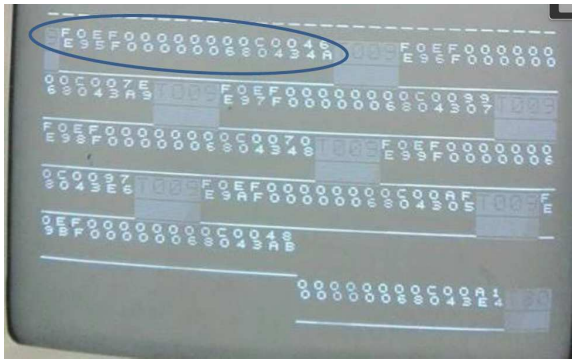

Figura 3.8 – Primeiro pacote da sequência da mensagem *Heartbeat*

A figura acima corresponde a uma mensagem *Heartbeat*, um dos vários tipos de mensagens que podem ser trocadas entre o GCS e o *Quadrotor* seguindo o protocolo *Mavlink*. A Tabela 2.2 apresenta o modelo estrutural presente em todos os pacotes. O primeiro pacote da mensagem *Heartbeat* corresponde à secção destacada na Figura 3.8.

Entre pacotes, varia o número de sequência e o valor do *checksum*. No caso do primeiro pacote da mensagem *Heartbeat* cada componente é identificado pela respetiva cor (Figura 3.9).

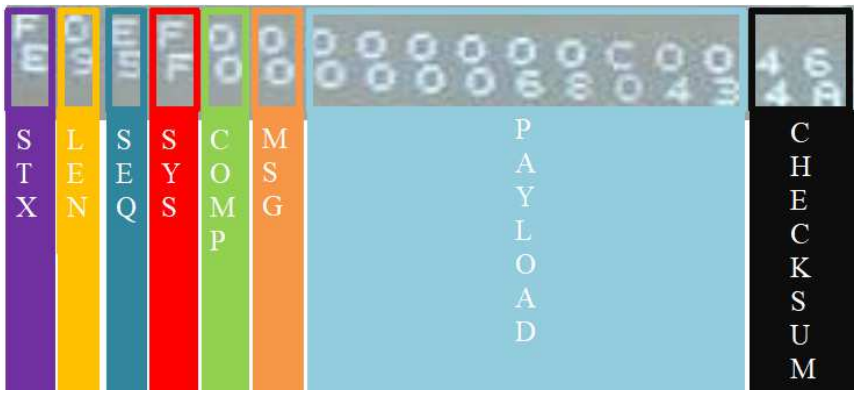

Figura 3.9 – Componentes do primeiro pacote da mensagem *Heartbeat*.

Ao analisar mais detalhadamente, foi possível verificar que o comprimento do *Payload* coincide com o valor dado pelo LEN (sendo este igual a 09). O *Payload* respetivo à mensagem em questão, sendo esta do tipo *Heartbeat*, apresenta a seguinte

constituição (Figura 3.10):

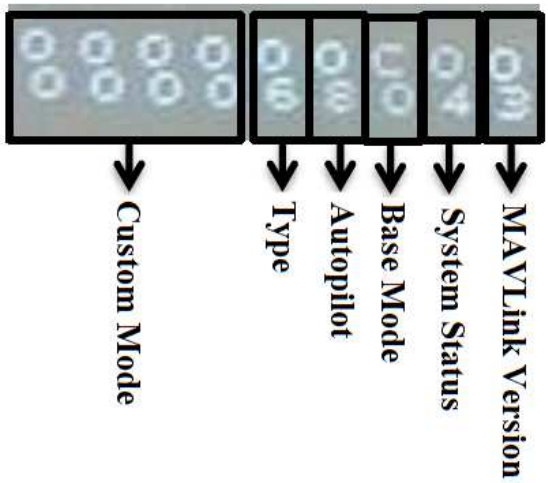

Figura 3.10 – Payload da mensagem *Heartbeat*

O *MAVLink Version*, tal como o nome indica, corresponde à versão do protocolo Mavlink usada, que neste caso foi a versão "03".

O *System Status* consiste numa *flag* que indica o estado do sistema através da função MAV_STATUS, que neste caso se encontra no estado de "*Standby*", o que significa que se encontra em "pausa"e pode ser acionado a qualquer momento.

O *Base Mode* corresponde ao modo do sistema em mapa de *Bits*.

O *Autopilot* consiste no tipo de *autopilot* usado e que está definido dentro da classe MAV_AUTOPILOT, que neste caso consiste no tipo de *Autopilot* Genérico.

O *Type* corresponde ao tipo de MAV utilizado, que neste caso se trata de um *Quadrotor*.

O *Custom Mode* consiste num mapa de *Bits* reservado para *flags* específicas do *Autopilot*.

O termo *Payload* é definido como o conteúdo da mensagem trocada entre o GCS e o *Quadrotor* [26]. Para enviar uma mensagem é necessário adequar o respetivo *Payload*

consoante o tipo de mensagem que queremos enviar. Por exemplo, se pretendermos enviar uma mensagem do tipo *Heartbeat*, teremos um *Payload* com um comprimento e uma constituição diferente relativamente a um *Payload* de uma mensagem do tipo comando. O comprimento do *payload* da mensagem *Heartbeat* é de 9 *Bits*, enquanto que no caso da mensagem do tipo comando é cerca de 12 *Bits*.

3.3.2 Comandos usados via *Mavlink*

Como já foi referido, o protocolo *Mavlink* define um conjunto de regras que se aplicam à comunicação estabelecida entre o GCS e o APM. Este também permite enviar comandos operacionais, sendo desta forma importante conhecer os tipos de mensagens de comando disponiveis, que são aceites pelo APM. Os comandos utilizados neste projeto, bem como a sua função, serão explicados detalhadamente de seguida:

- *Arm/Disarm*

  O comando *Arm/Disarm* (MAV_CMD_COMPONENT_ARM_DISARM) permite colocar os motores, caso o quadrotor esteja no chão, num dos dois estados: *Armed* ou *Disarmed*. Para que os motores fiquem no estado *Armed* é necessário que o primeiro parâmetro (*Param1*) seja igual a "1". Caso seja igual a "0", os motores ficaram no estado *Disarmed* (Ver Tabela 2.6) [19].

  Ainda é possível, embora não seja recomendável, colocar os motores no estado *Disarmed* em pleno voo caso o segundo parâmetro (*Param2*) deste comando tenha o valor "21196"[19].

- *Takeoff*

  O comando *Takeoff* (MAV_CMD_NAV_TAKEOFF) faz com que o *Quadrotor* inicie um voo vertical a uma altitude definida. Se o comando for enviado durante uma missão, caso a altitude atual do *Quadrotor* seja menor em relação à altitude enviada pelo comando, o *Quadrotor* irá subir até a altitude enviada. Caso contrário, o comando será ignorado e a missão continua. Contudo, não é possível enviar uma mensagem deste tipo com o intuito de fazer com que o

veículo baixe a sua altitude. Como já foi referido anteriormente, à exeção do parâmetro 7 (*Param7*), o valor dos restantes parâmetros (*Param1 a Param6*) é inváriavel. Já o valor do *Param7* é passível de ser modificado e corresponde à altitude que o veículo irá atingir após a descolagem. Este comando deverá dar início a qualquer missão [19].

- *Navigate to Waypoint*

  O comando *Navigate to Waypoint* (MAV_CMD_NAV_WAYPOINT) faz com que o *Quadrotor* voe até uma latitude, longitude e altitude estipulada. Quando o *Quadrotor* alcança essa coordenada permanece durante um determinado período de tempo nessa mesma posição até receber instruções para avançar até a um novo *waypoint*. O tempo de espera máximo é de aproximadamente 18 horas, o que ultrapassa em muito a duração de qualquer bateria usada para este efeito [19]. Um vez que este comando é usado na missão de voo, não se verifica este problema pois o *Quadrotor* executa uma sequência de comandos (ver página 74).

  Quanto à estrutura do *payload* que constitui a mensagem, apresenta a seguinta configuração (Adaptado de [19]): o *Param1* do *payload* da mensagem enviada refere-se ao tempo de espera (em décimas de segundo) em cada ponto; o *Param2* corresponde ao raio (em metros) que o *Quadrotor* pode ocupar apartir do *waypoint* enviado (modo de minimizar o erro da leitura do GPS). Caso essa área espacial seja ocupada pelo *Quadrotor*, a missão é concretizada; o *Param3* pode tomar diferentes valores, se for "0"o veículo irá passar pelo ponto definido como *waypoint*. Caso o valor seja superior a "0", o *Quadrotor* descreve uma órbita no sentido dos ponteiros do relógio em torno do *waypoint*. Caso o valor seja inferior a "0", a órbita descrita pelo *Quadrotor* é no sentido contrário ao dos ponteiros do relógio relativamente a este mesmo *waypoint*; o *Param4* corresponde ao ângulo *Yaw* desejado; os *Param*5, 6 e 7 correspondem, respetivamente, à latitude, longitude e altitude do ponto desejado.

- *Return to Launch*

  O commando *Return to Launch* (MAV_CMD_NAV_RETURN_TO_LAUNCH)

indica ao *Quadrotor* que deve aterrar no último sítio onde os motores ficaram no estado *Armed*. Este comando não contém parâmetros e o quadrotor antes de aterrar voa até uma altitude especifica (15m por omissão). Este comando deverá ser o último comando da missão (o contexto de missão será explicado nos próximos subcapítulos) [19].

- *Land*

  O comando (*Land* (MAV_CMD_NAV_LAND) faz com que o quadrotor aterre na posição atual ou em uma latitude e longitude estipulada. Uma vez feita a aterragem, os motores não irão ficar no estado *Disarmed* até o modo de voo seja alterado para AUTO. O valor do *Param1* corresponde à altitude (em metros) que o *Quadrotor* irá assumir caso o comando *Land* seja abortado. O *Param2* e o *Param3* são parâmetros vazios. O *Param4* corresponde ao ângulo *Yaw* desejado. O *Param5* do *payload* da mensagem diz respeito à latitude desejada, caso esta seja "0"o quadrotor irá aterrar na posição atual. Os *Param* 6 e 7 correspondem, respetivamente, à longitude e à altitude desejada para a aterragem [19].

- *Set Mode*

  O comando *Set Mode* (MAV_CMD_DO_SET_MODE) permite ao *Quadrotor* mudar o seu modo de voo. O modo de voo é importante pois é através dele que é possível desempenhar todos os comandos antes, durante e após efectuar o voo sem comprometer a segurança do mesmo. O valor do *Param1* corresponde ao valor do modo de voo que se pretende que o *Quadrotor* assuma. O *Param2* e o *Param3*, respectivamente *Custom mode* e *Custom sub mode*, são parâmetros em que o utilizador pode definir o valor de correspondência que pretender de um determinado modo. O *Param4*, *Param5*, *Param6* e *Param7* são parâmetros vazios [19].

Uma vez enviado o comando, este só irá ser executado após a receção de uma mensagem de *acknowledge* ao comando específico. A mensagem de *acknowledge* tem como objetivo fornecer a informação ao cliente se o comando enviado foi executado,

não pode ser executado ou se é desconhecido. O seguinte diagrama ilustra este processo [9].

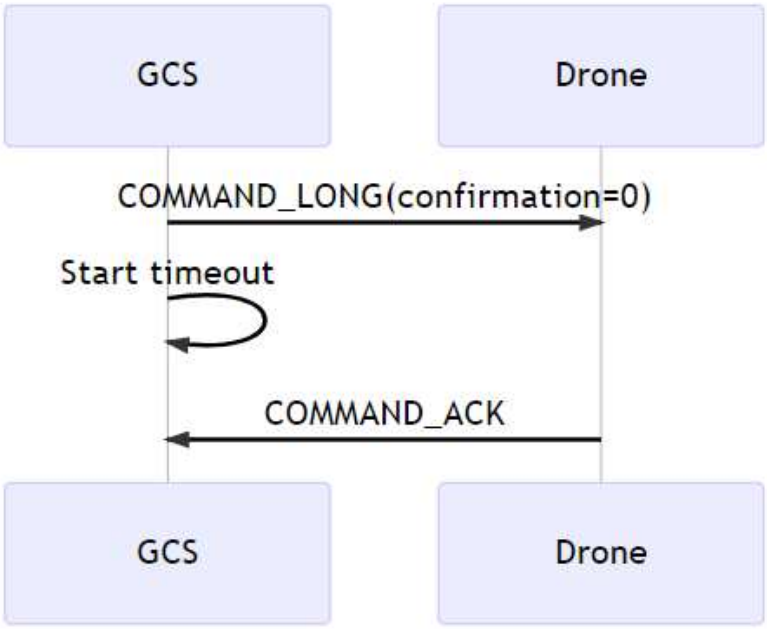

Figura 3.11 – Diagrama - Modo de funcionamento das mensagens de *acknowledge* ([9]).

3.3.3 GCS - Interface

Como já tinha sido referido anteriormente, de modo a criar uma estação de controlo no solo (GCS) foi necessário elaborar um programa que permitisse interagir com o *Quadrotor* remotamente. Esta interface foi desenvolvida em *Java*, assim como o programa cuja função é de servidor. A interface está presente no computador, enquanto que o servidor está presente no Raspberry PI.

A elaboração da interface foi realizada usando como base a API do APM 2.5, em que estão contidos todos os parâmetros pertencentes ao *Quadrotor* (foram utilizadas, dentro dos pacotes *open-source*, várias bibliotecas como base).

Um vez que foi feito um sistema usando o modelo cliente-servidor, esta interface passou a desempenhar o papel de cliente.

3.3.4 GCS - *Front End*

O termo *Front-End* consiste na parte da apresentação gráfica de um determinado *software* ocultando, aos olhos do utilizador, todo o código que o constitui.

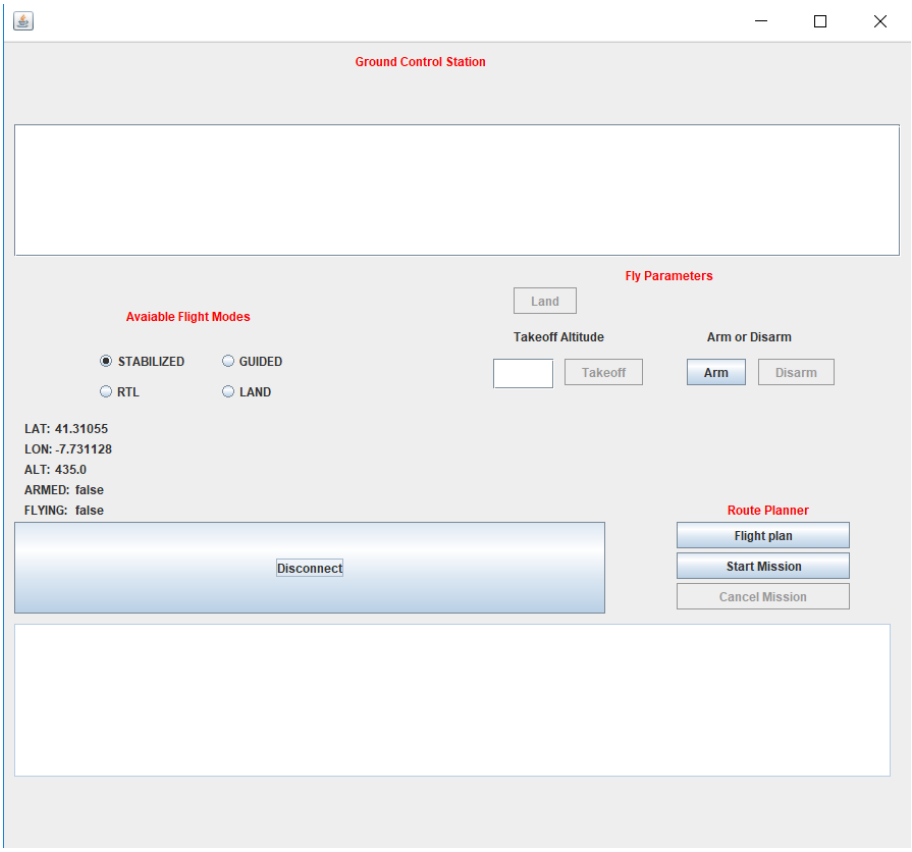

Figura 3.12 – jFrame *Ground Control Station*

Para a sua concretização foi utilizado o *Swing*, uma interface de programação de aplicações (API, do inglês *Application Programming Interface*), pertencente ao *Java* que tem como objetivo desenvolver uma interface gráfica.

A API *Swing* inclui várias classes oferecendo inúmeras possibilidades de construção em GUI, tais como painéis (*jPanel*), botões (*jButtom*) ou até janelas (*jFrame*), possibilitando desse modo realizar o *Front-End* desejado.

O *Front-End* elaborado é constituído por duas *jFrames*. A primeira *jFrame* (Figura 3.12) consiste na *frame* principal, onde é possível enviar os principais comandos operacionais, anteriormente mencionados, assim como monitorizar vários dados provenientes do *Quadrotor* tais como informação sobre a bateria, localização atual (latitude e longitude), altitude atual, entre outros.

A interação com o botão *Connect* faz com que seja criado um *thread* com o objetivo de se conectar ao servidor.

Para além disso, também é possível iniciar uma missão (missão essa que é criada na segunda *jFrame*) e cancelá-la caso seja necessário. A segunda *jFrame* (nome atríbuido de *Flight planner*, Figura 3.13) foi realizada com o objetivo de planear a missão que o *Quadrotor* seguiu.

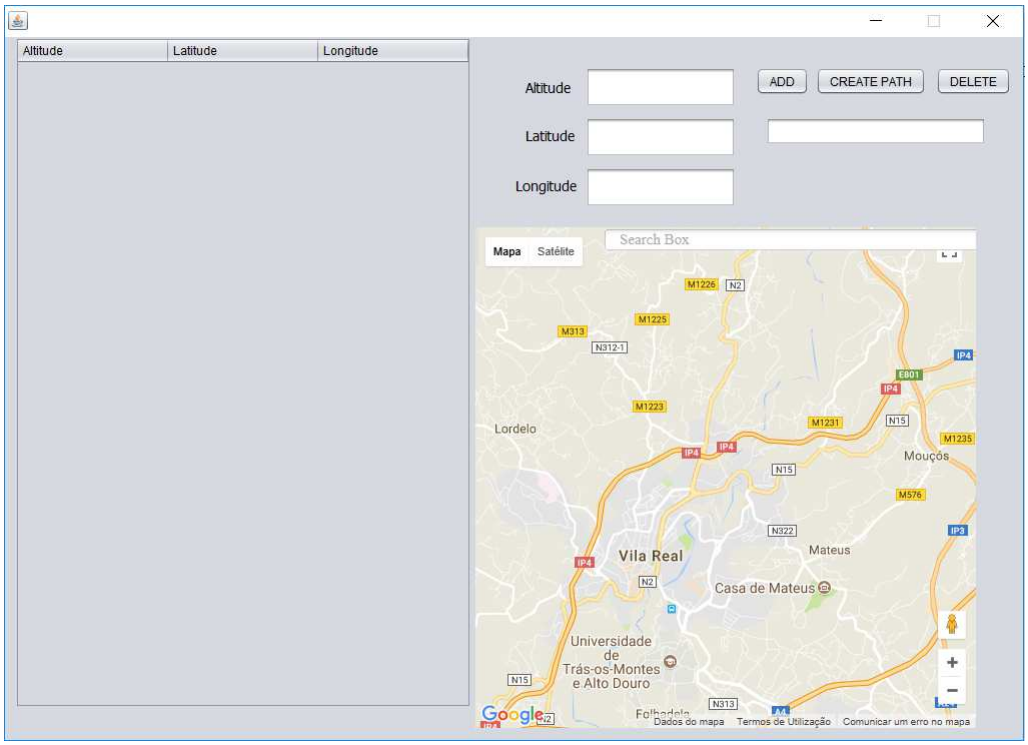

Figura 3.13 – jFrame *Flight planner*

Com o auxílio da Google API, foi possível mostrar um mapa gerado através do *script* dentro do ficheiro *HTML*, dentro de um *jPanel*. O ficheiro *HTML map*, para além

de gerar o mapa, possui vários *scripts* que realizam as ações de adicionar/remover pontos (*map markers*) e de fazer um caminho através dos pontos (*path*) (Figura 3.14).

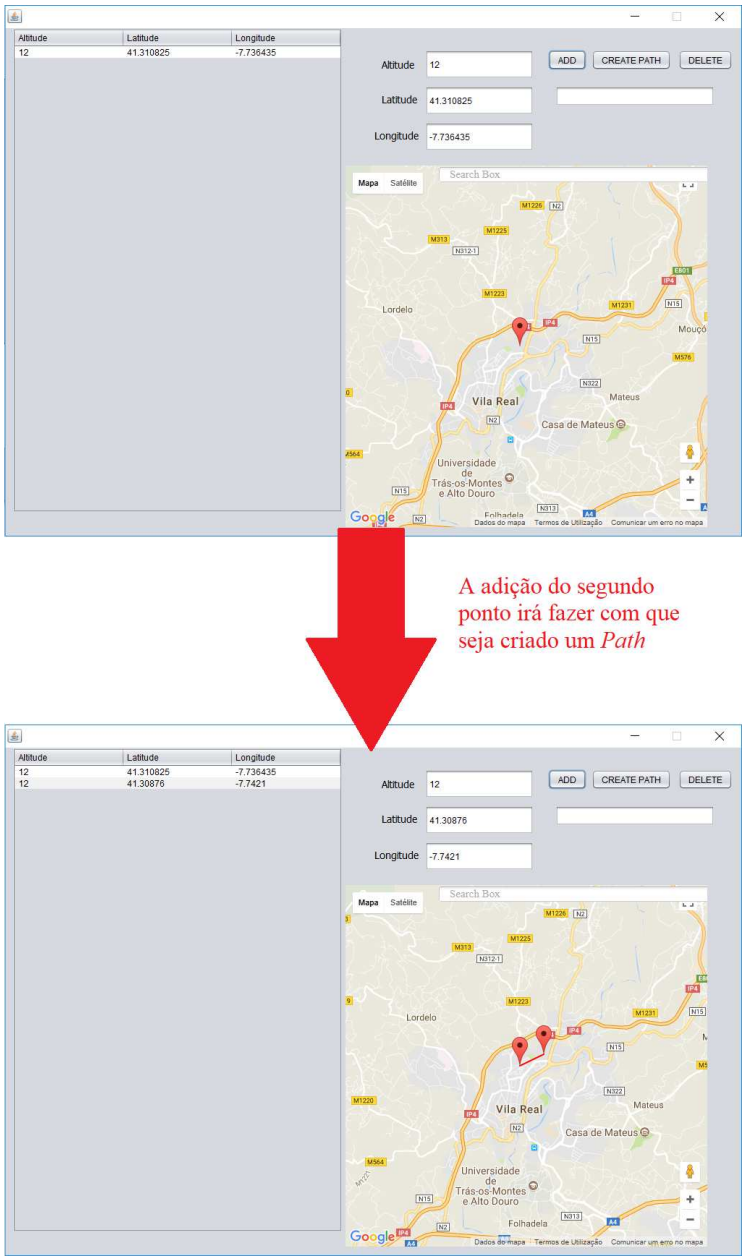

Figura 3.14 – jFrame *Flight planner* com dois *map markers* adicionados e com o *path* entre eles construído.

3.3.5 Protocolo de mensagens entre Cliente e Servidor - Back End

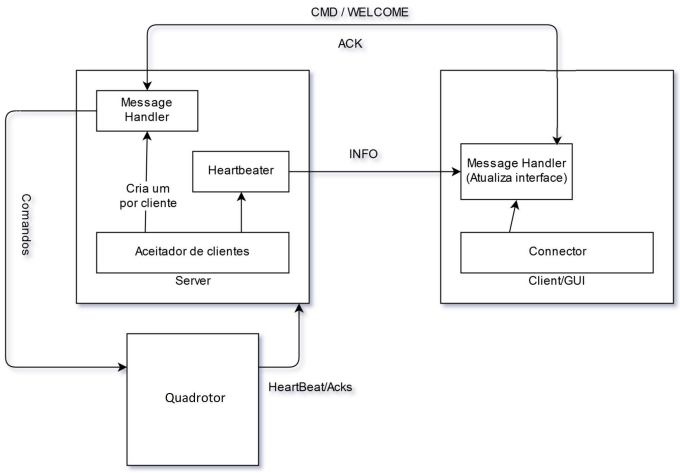

Figura 3.15 – Diagrama - Protocolo de mensagens entre Cliente e Servidor.

Foi elaborado um protocolo de mensagens entre o cliente e o servidor onde foram diferenciados vários tipos de mensagens (Figura 3.15) e vários tipos de clientes. Os tipos de mensagens existentes são:

- CMD: mensagem comando que é enviada pelos clientes ao servidor quando querem que o *Quadrotor* execute um certo comando

- ACK: mensagem *acknolegdment* que é enviada a partir do servidor para os clientes e que tem como objetivo relatar o estado de execução do comando (executou o comando, falhou a execução do comando, comando desconhecido, *time out*, entre outros)

- WELCOME: mensagem proveniente do cliente para o servidor com o objetivo de informar o servidor sobre o tipo de cliente que se conectou (3 clientes disponíveis: Interface em *Java* (GCS), *Command Line* e Interface elaborada em *Android* (*Smartphone*))

- INFO: mensagem de *heartbeat* que o servidor manda ao cliente com o objetivo de informar o cliente sobre o estado atual do sistema pertencente ao *Quadrotor* (altitude, latitude, longitude, modo de voo atual, se se encontra em voo e se está no estado *armed* ou *disarmed*). Para além disso, também serve para o cliente saber se continua conectado com o servidor.

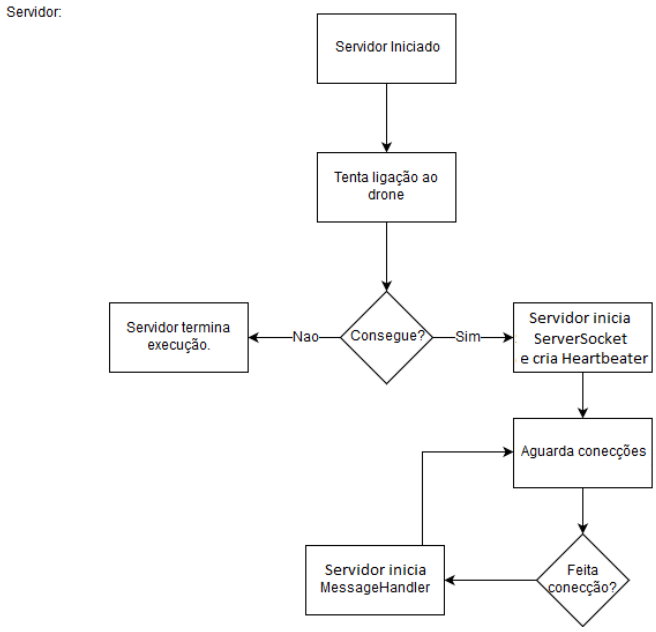

Figura 3.16 – Fluxograma - Modo de funcionamento do servidor.

O servidor (Figura 3.16) começa por iniciar um objeto *drone* que se conecta ao *Quadrotor* por via de uma porta série e obtém mensagens que fornecem informações sobre o estado atual do *Quadrotor*. Essas informações são continuamente analisadas e tratadas. É extraída e guardada a informação necessária para o bom funcionamento do *Quadrotor* (o modo de voo e o estado em que ele se encontra, etc.). Este objeto tem também o objetivo de utilizar a porta série para enviar mensagens de comando ao *Quadrotor*.

Posteriormente, o servidor começa por criar um objeto *Serversocket* que desempenha o papel de *socket* da parte do servidor. Este *socket* tem o objetivo aguardar conecções por parte de algum cliente. É também iniciado um *thread* denominado *heartbeater*

que caso esteja algum cliente conectado envia mensagens do tipo INFO para esses clientes.

Sempre que algum cliente se conecta ao *socket* do servidor é criado um *thread* denominado de *messageHandler* que tem como objetivo receber, analisar e classificar as mensagens provenientes do cliente em questão e realizar a respetiva tarefa. Dependendo do tipo de informação destacada o *messageHandler* poderá executar um comando através do objeto *drone* e depois enviar o ACK ao cliente. Caso seja uma mensagem do tipo WELCOME será retirada a informação sobre o tipo de cliente conectado (Ver Figura 3.17).

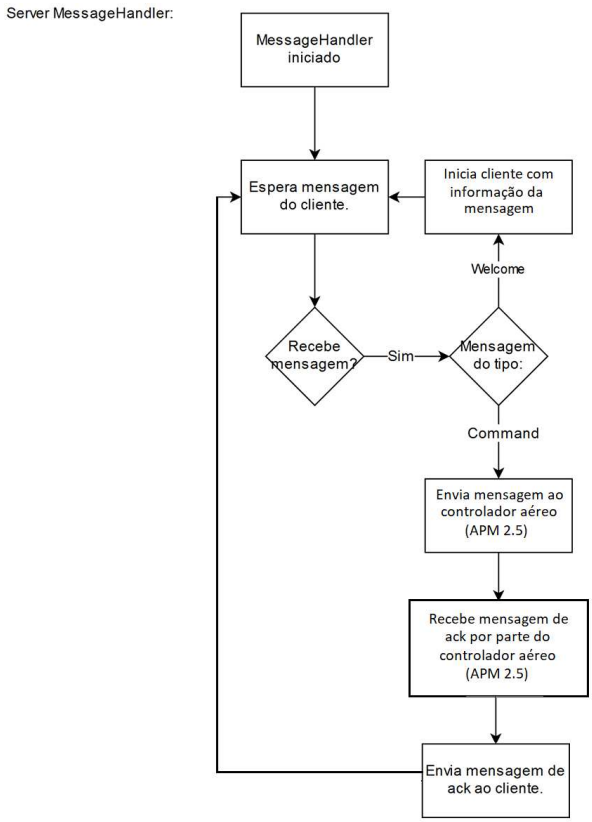

Figura 3.17 – Fluxograma - Modo de funcionamento do messageHandler por parte do servidor.

O *messageHandler* pode assumir 1 dos 3 modos, sendo eles: NORMAL, SPM e MISSAO que ditam o comportamento deste *thread* consoante as mensagens recebidas

pelo seu cliente. Os modos do *messageHandler* serão explicados de seguida:

Modo NORMAL: Modo destinado para a execução de comandos operacionais, tais como *Arm, Disarm, Takeoff, Land, Set mode,* RTL, entre outros.

Modo SPM: Modo destinado à interface elaborada em *Android* e serve para o *Quadrotor* seguir as coordenadas enviadas pelo cliente *Smartphone*

Modo MISSAO: Modo destinado à interface elaborada em *Java* que é ativado quando o cliente pretende realizar uma *Flight Mission* (ver página 74).

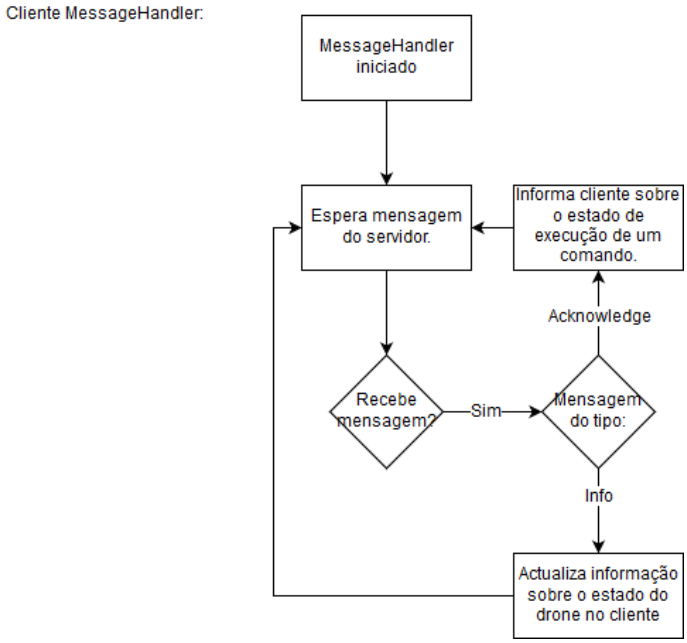

Figura 3.18 – Fluxograma - Modo de funcionamento do messageHandler por parte do client.

Por parte do cliente é também criado um *thread messageHandler* cujo objetivo é o mesmo que o *messageHandler* por parte do servidor, isto é, receber e analisar mensagens desta vez enviadas pelo servidor ao cliente e reagir consoante o conteúdo das mesmas (Figura 3.18). As mensagens recebidas por parte do cliente são do tipo INFO ou ACK. Se for do tipo INFO, é atualizada a informação sobre o estado do *Quadrotor* ao cliente. Se for do tipo ACK dá a conhecer ao cliente o resultado da tentativa de execução do comando.

3.3.6 *Missão de voo*

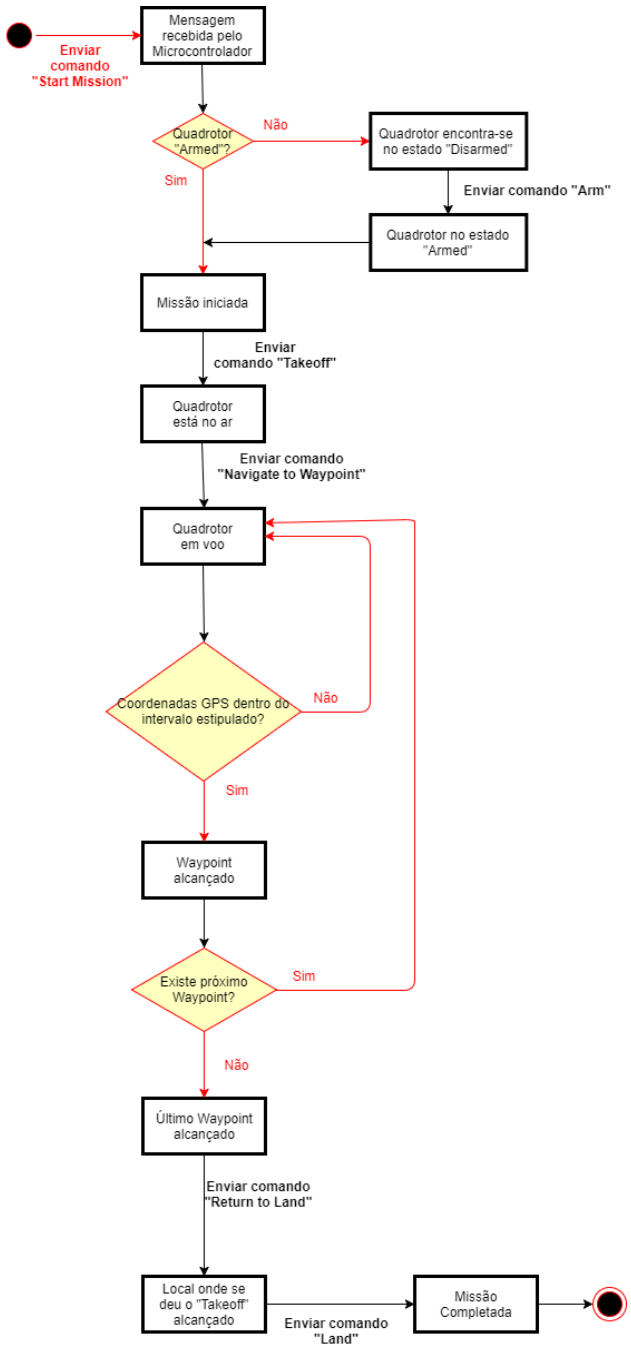

Figura 3.19 – Diagrama de Estados - *Mission Start*.

O diagrama constante na Figura 3.19 ilustra a sequência de procedimentos que ocorrem quando é enviado um comando *Start Mission*.

A missão de voo consiste na rota que o *Quadrotor* irá realizar, assim como todas as ações que este irá tomar durante a sua execução. A segunda *jFrame*, apresentada anteriormente, tem como objetivo planear toda a missão de voo, permitindo deste modo, que o *Quadrotor* estabeleça um voo autónomo dentro dos perímetros estipulados e execute ações definidas, conseguindo deste modo obter um controlo mais preciso do voo.

Relativamente à composição da missão de voo, primeiro o GCS envia um comando de "startMission"que consequentemente irá fazer com que o servidor assuma o modo MISSAO. De seguida, são enviados pelo GCS todos os pontos pertencentes ao caminho delineado pelo *FlightPlanner*. Após a receção destes pontos, o servidor armazena-los e envia as mensagens do tipo comando (*Set Mode*, *Arm* e *Takeoff*) ao *Quadrotor*. É então enviado um comando *Navigate to Waypoint* em que o destino é o primeiro ponto da missão. Quando este ponto é atingido (ou um pequeno raio à sua volta, visto existirem pequenos erros nas coordenadas GPS) pelo *Quadrotor*, é novamente enviado um comando *Navigate to Waypoint*, desta vez para o ponto seguinte. Este processo é repetido até ser atingido o último ponto da missão.

Quando este é atingido, o SBC envia um *Navigate to Waypoint* para o primeiro ponto onde foi realizado o *Takeoff*. Após a chegada do *Quadrotor* ao local, o SBC envia um comando de *Land* para que este aterre nessa posição. Uma vez que o *Quadrotor* esteja no solo, este assume o modo NORMAL e é enviado um comando de *Disarm*. Posteriormente, é enviada uma mensagem de ACK que notifica o cliente sobre este acontecimento.

É de referir que se algum comando falhar durante uma missão, a missão é abortada resultando no envio de um comando *Land*, cujo local de aterragem do *Quadrotor* é o mesmo de onde ele partiu. Adicionalmente, o cliente é notificado destes acontecimentos.

Um vez que o caminho começa a ser executado, o *Quadrotor* pode sofrer o impacto de diversos fatores externos de distintas naturezas. O código fonte do microcontrolador possui várias opções de segurança que serão descritas no próximo subcapítulo.

Foi também desenvolvida uma segunda interface para a plataforma *Android*, de modo a facilitar os testes em campo aberto. O funcionamento desta segunda interface será explicado nos próximos subcapítulos.

3.3.7 Mecanismos de failsafe

O APM 2.5, tal como qualquer outro modelo de *Ardupilot*, possui diversos mecanismos de *failsafe*. Este tipo de mecanismo tem como objetivo recuperar e/ou prevenir uma eventual perda de controlo do *Quadrotor*. Os principais mecanismos de *failsafe* são (Adaptado de [16]):

1. *Radio failsafe*;
2. *Battery failsafe*;
3. *GCS failsafe*;
4. *EKF / DCM failsafe*;
5. *Simple GeoFence*;
6. *Polygon Fence*;
7. *Crash Check*;
8. *Parachute*;
9. *ADSB Avoidance of manned aircraft*.

Uma vez que foi desenvolvido um GCS para um *Quadrotor* simples (apenas com os componentes necessários para a execução de funções básicas), somente os mecanismos *Battery failsafe* e GCS *failsafe* foram utilizados uma vez que são os mais simples de implementar.
O *Radio failsafe* é um mecanismo usado quando o *Quadrotor* é controlado por um trasmissor RC (*Radio Controller*). Este é ativado caso haja perda de contacto entre o transmissor e o veículo, fazendo com que o veículo execute certas ações de

segurança, tais como *Return to Land* ou *Land*. Uma vez que este mecanismo está destinado a transmissores RC não foi utilizado nesta dissertação.

O mecanismo EKF/DCM *failsafe* é um *upgrade* do mecanismo GPS *failsafe* para *Ardupilots* com uma versão superior à 3.3 [13]. Este mecanismo somente é compatível com o modelo *Pixhawk*.

O *Simple GeoFence* é um mecanismo que impede que o veículo, ao descolar, voe para fora de um limite previemente estipulado. Caso o ultrapasse, este mecanismo é ativado fazendo com que o veículo execute certas manobras de segurança, tais como *Return to Land* caso atinja o limite ou *Land* caso ultrapasse em 100 m esse limite. Mais uma vez, este mecanismo não se revelou apropriado a ser utilizado neste projeto.

O *Polygon Fence* tinha por base a mesma metodologia que o *Simple GeoFence*, contudo o limite da trajetória do veículo era definido mediante uma forma poligonal. Similarmente, este mecanismo não foi utilizado por não ser adequado a este caso em concreto.

O *Crash Check* é um mecanismo responsável por colocar os motores no estado *disarm*, caso o *Quadrotor* entrasse em colapso. Este mecanismo não foi usado por falta de oportunidade em tempo útil.

O mecanismo *Parachute*, similarmente ao *Crash Check*, em caso de colapso por parte do veículo aciona o paraquedas. Uma vez que o *Quadrotor* usado não possuia paraquedas, este mecanismo não se revelou adequado a este caso.

Por fim, o mecanismo *ADSB Avoidance of manned aircraft* é responsável por evitar colisões com outros UAV presentes no mesmo espaço aéreo. Apesar de ter sido utilizado um único *Quadrotor* neste projeto, poderiam coexistir outros UAVs na mesma área geográfica. Contudo, este mecanismo não foi utilizado uma vez que seria necessário que o veículo em questão tivesse um sensor próprio (*uAvionix ADS-B PING*) para a deteção de outros UAV.

3.3.8 Battery failsafe

O *Battery failsafe* é um mecanismo que tem como objetivo alertar o utilizador através de uma mensagem dentro do GCS ou mediante um sinal emitido pelo próprio

Quadrotor (som estridente de um *Buzzer* ou finalmente, através de um sinal de luz itermitente proveninte de um LED) no caso da tensão da bateria se encontrar abaixo de um certo valor estipulado durante a condição de voo ou permanência no solo [13].

Este mecanismo de *failsafe* é ativado quando a bateria principal apresenta uma tensão abaixo de 10,5 volts (tensão escolhida) e abaixo dos mAh definidos para este sistema. Um vez acionado, o mecanismo de *battery failsafe* fará com que o *Quadrotor* execute determinadas ações consoante a estado em que o *Quadrotor* se encontre[13].

Para que o *battery failsafe* seja ativado e se mantenha neste estado é necessário que o parâmetro FS_BATT_ENABLE apresente o valor "1"(*Land*: o *Quadrotor* aterra no local onde se encontra) ou "2"(RTL do inglês *Return to Launch*: o *Quadrotor* irá retornar às coordenadas onde realizou o *Takeoff* [13].

Caso o parâmetro apresente o "0", o *failsafe* encontra-se no estado desativado. Esta última situação não é recomendável pois pode pôr em causa a integridade do *Quadrotor* [13]. Na Tabela 3.4 encontram-se resumidas as diversas ações executadas pelo *Quadrotor*

Tabela 3.4 – Comportamentos do sistema no modo *battery failsafe* (Adaptado de [13])

Comportamento	Estado do Sistema
Parado no solo	O *Quadrotor* está no estado *Disarmed*
Motores ficam no estado *Disarm*	O *Quadrotor* encontra-se no modo de voo *Stabilize* ou *Acro*. O valor do *Throttle* é zero ou o veículo está pousado no solo.
Return to Launch (RTL)	Caso o valor do parâmetro FS_BATT_ENABLE apresente o valor "2". O *Quadrotor* encontra-se no modo de voo AUTO e o *Quadrotor* está dentro de uma distância de dois metros em relação às coordenadas em que o *Takeoff* foi realizado.
Land	Qualquer outro estado não mencionado

3.3.9 GCS *failsafe*

O mecanismo GCS *failsafe* tem como objetivo evitar uma eventual perda de controlo no caso de ocorrer perda de comunicação entre o *Quadrotor* e o GCS.

O modo de voo presente no *Quadrotor* é alterado quando o seu contacto com o GCS é restabelecido. Após o GCS *failsafe* intervir no sistema, o modo de voo muda para *Land* (parâmetro FS_GCS_ENABLE com o valor igual a "1") ou RTL (parâmetro FS_GCS_ENABLE com o valor igual a "2". Antes do GCS *failsafe* intervir, caso o modo de voo presente no *Quadrotor* seja AUTO o *Quadrotor* irá prosseguir com a missão)[17]. A Tabela 3.5 resume os comportamentos e quando é que os mesmos ocorrem.

Tabela 3.5 – Comportamentos do sistema no modo *GCS failsafe* (Adaptado de [17])

Comportamento	Estado do Sistema
Motores ficam no estado *Disarm*	O *Quadrotor* encontra-se no modo de voo STABILIZE ou ACRO. O *Throttle* apresente o valor igual a "0".
Return to Launch (RTL)	O *Quadrotor* está a uma distância superior a dois metros em relação às coordenadas em que o *Takeoff* foi realizado.
Land	O *Quadrotor* está dentro de uma distância de dois metros em relação às coordenadas em que o *Takeoff* foi realizado.
Continuar com missão	O *Quadrotor* encontra-se no modo de voo AUTO e o GCS permite continuar a missão dentro desse modo

3.4 Aplicação Android

Foi elaborada uma interface para a plataforma *Android*, cujo o seu objetivo era facilitar os testes com o *Quadrotor* em campo aberto face ao sistema cliente-servidor via *socket* desenvolvido (Figura 3.20). Deste modo, torna-se mais prática a utilização deste tipo de equipamentos em contexto real.

Figura 3.20 – Interface desenvolvida em Android

Esta interface contém 5 botões, sendo eles:

Connect/Disconnect: Este botão tem como principal objetivo conectar o dispositivo móvel em questão com o SBC a bordo do *Quadrotor*, de modo a conseguir obter os dados relativos à latitude, longitude e altitude relativa (em relação ao nível do mar). Para além disso, a interação com este botão faz com que os botões *arm*, *disarm*,*land*, *takeoff* e *start spm* fiquem disponíveis.

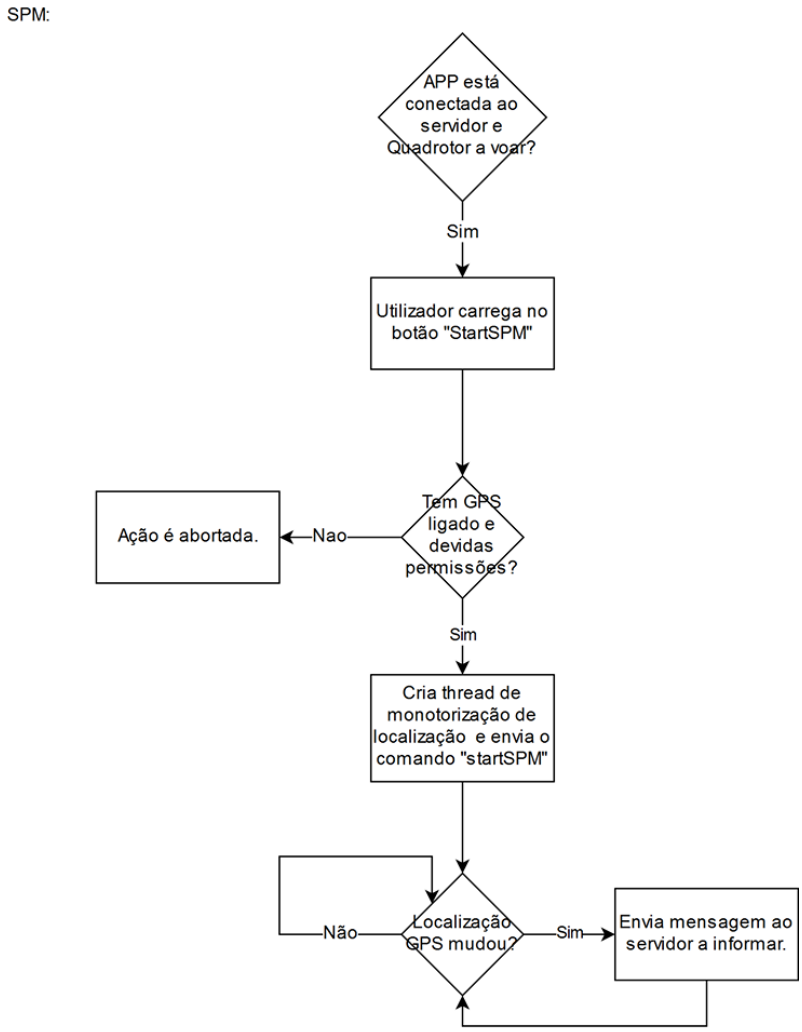

Figura 3.21 – Fluxugrama do modo de funcionamento do Start SPM.

Start SPM: Para que este comando seja exequível é imperativo que o *Quadrotor* esteja em voo (Figura 3.21). Após o *Quadrotor* se encontrar no modo *Stabilized* e *Armed* é necessário que este realize uma descolagem. Quando é criada a conecção entre o servidor e o dispositivo móvel em causa, o sistema assume o modo SPM e o servidor fica à "escuta"de possíveis mudanças de localização, através da informação proveniente do GPS do dispositivo (a cada 1,5 segundos "escuta"os valores de GPS). Sempre que ocorre uma mudança de localização é enviado ao SPM uma mensagem que é traduzida no envio ao *Quadrotor* de um comando *Navigate to Waypoint*, com a nova localização presente na mensagem enviada.

Land: Esta função faz com que seja enviado ao *Quadrotor* um comando *Land*. O modo *Land* serve apenas para fazer com que o *Quadrotor* aterre quando este recebe o comando *Land*. Caso ocorra uma quebra de conecção com o servidor, o *Quadrotor* aterra no local onde se encontra. Por fim, o veículo muda para o estado *Disarm*.
Arm/Disarm: Estes botões permitem que o *Quadrotor* entre no estado *Armed* e *Disarmed*, respetivamente. Para que este consiga entrar no modo *Armed*, é necessário que este esteja no modo *Stabilized*, por isso foi definido como condição que sempre que o botão *Arm* é interagido, o modo de voo se altere para *Stabilized*. Uma outra condição para o estado *Arm* é o veículo não se encontrar no estado *Flying* ou já no estado *Armed*. O comando *Disarm* pode ser efetuado caso o veículo se encontre no estado *Armed* e não no estado *Flying* (independentemente do modo).
Takeoff: Este botão permite que o *Quadrotor* estabeleça uma descolagem para a altitude previamente definida pelo cliente. Para que este seja capaz de descolar, tem que se encontrar previemente no estado *armed* e no modo de voo *Guided*. Contudo, para este assumir o estado *armed* tem de simultâneamente de se encontrar no modo *Stabilized*.

4 Resultados

4.1 Testes em Campo Aberto

Houve a necessidade de realizar testes em campo aberto, uma vez que se pretende que futuramente o sistema desenvolvido neste dissertação seja utilizado para a recolha de dados edafoclimáticos em campos agrícolas. A par disso, também permitiu verificar o comportamento do *Quadrotor* em campo aberto face ao sistema desenvolvido.

- 1º Teste

A implementação deste projeto permitiu obter vários resultados correspondentes aos objetivos previamente delineados. Primeiramente, foram testados vários comandos operacionais através de uma ligação série (Figura 4.1) dentro de um sistema que apenas envolvia o GCS e o *Quadrotor* (as mensagens eram enviadas diretamente para o APM). Para além disso, foi possível enviar comandos ao *Quadrotor*, o qual os executou (Figura 4.2). Foram também testados vários comandos como *Arm* (Figura 4.3), *Disarm* ou *Takeoff* (Figura 4.4).

Figura 4.1 – Quadrotor junto ao GCS.

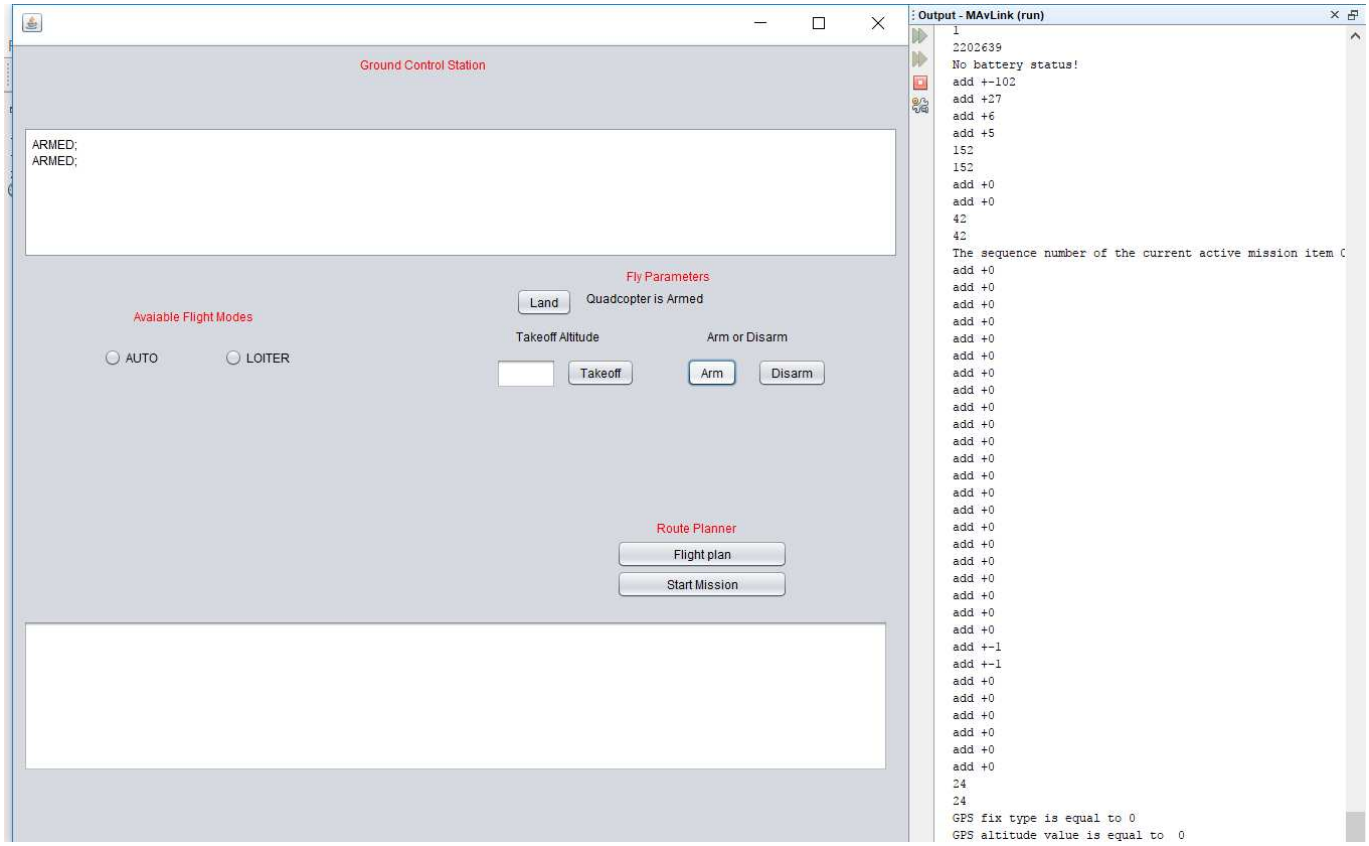

Figura 4.2 – Envio e receção de mensagens na Interface elaborada.

Figura 4.3 – Quadrotor no estado *Arm* (sem hélices).

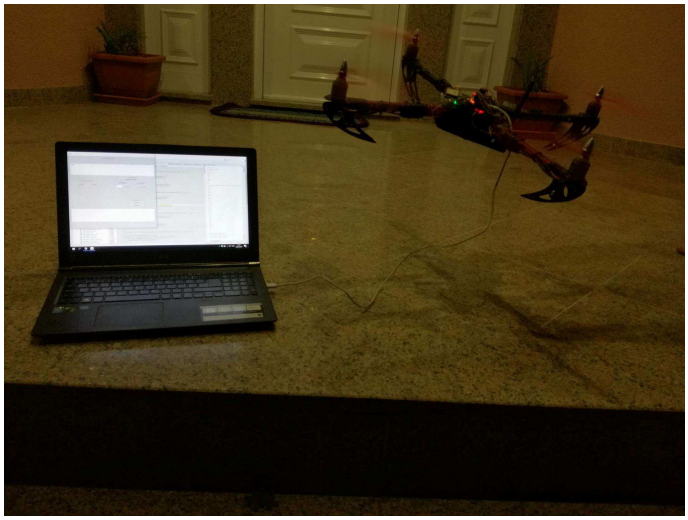

Figura 4.4 – Quadrotor a executar "*Takeoff*" de baixa altitude.

• Conclusão do 1º Teste

Durante o período de testes, este *Quadrotor* revelou uma anomalia num dos ESC, o que impossibilitou o voo do mesmo. Consequentemente, este modelo foi substituído por um modelo mais atual. No entanto, ainda foi possível registar alguns resultados antes dessa anomalia. Para além disso, o teste foi executado apenas com duas identidades: a interface GCS e o *Quadrotor*. As figuras anteriores mostravam o registo desses mesmos resultados.

Este sistema demonstrou ser pouco eficaz uma vez que revelou uma grande perda das mensagens provenientes do *Quadrotor* (um dos tipos de mensagens que foram

perdidas foram as mensagens do tipo *acknowledge*), o que pode ser observado na Figura 4.2. A anomalia por parte de um dos ESC impossíbilitou que fossem testados quaisquer outros parâmetros.

• 2º Teste (20/04/2018)

Após os resultados obtidos no 1º teste, houve um melhoramento do sistema anteriormente apresentado. Desta vez, era constituído por um sistema de cliente-servidor via *socket*, onde a interface desenvolvida em *Java* representava o papel de cliente (GCS) e o *Raspberry* Pi o papel de servidor. Este teste foi realizado com um novo modelo visto que o *Quadrotor* anterior revelou uma anomalia ao nível dos ESC. As figuras que se seguem mostram o registo desses mesmos resultados:

Figura 4.5 – Quadrotor a executar o comando "*Land*".

Figura 4.6 – Quadrotor a executar o comando "*Navegate to waypoint*".

Figura 4.7 – Quadrotor a executar o comando "*Takeoff*".

• Conclusão do 2º Teste

Este sistema apresentou um melhor desempenho em comparação com o sistema anterior. Nomeadamente, executou com êxito vários comandos tais como: *Arm*, *Disarm*, *Takeoff* em várias altitudes (Figura 4.7), *Navigate to Waypoint* (Figura 4.6) e *Land* (Figura 4.5). Contudo, revelou não ser o mais apropriado visto que houve uma perda significativa de mensagens provenientes do *Quadrotor*, o que levou a que o *Quadrotor* tivesse sido arrastado pela força do vento sem qualquer ação possível sobre ele, logo após a concretização de um comando *Takeoff* por este.

Nomeadamente, os comandos para a aquisição dos dados edafoclimáticos (como temperatura, humidade relativa do ar, intensidade luminosa, etc.) através de sensores incorporados no SBC, não foram desenvolvidos, o que seria importante realizar numa próxima etapa. Para além disso, seria importante estudar melhor o comportamento do *Quadrotor* conectado com a interface desenvolvida durante esta dissertação, na recolha dos dados edafoclimáticos pretendidos em testes *outdoor*, neste caso específico num campo agrícola.

5 Conclusão e Trabalho Futuro

Nesta dissertação foi abordado o funcionamento e os componentes do modelo do *Quadrotor* estudado. De forma a atingir os objetivos definidos, foi previamente elaborado com sucesso um sistema de cliente-servidor via *socket* com o objetivo de controlar à distância um veículo aéreo não tripulado *Quadrotor*), de modo a facilitar e automatizar a prática agrícola contribuindo para aumentar a sua produtividade, garantir a qualidade dos alimentos e proporcionar uma maior sustentabilidade económica futura desta atividade.
As principais vantagens da utilização de veículos aéreos não tripulados na prática agrícola são:

- A precisão com que se consegue monitorizar grandes áreas agrícolas com um ligeiro atraso em relação ao tempo real (existem sempre atrasos na receção dos dados).

- A sua monitorização consiste em voos periódicos, o que permite obter uma monitorização durante todo o período de produção.

- A possibilidade de estar equipado com sensores que recolhem dados edafoclimáticos provenientes das culturas agrícolas ou até a captação de várias fotografias ou vídeos, adicionando um aspeto visual à monitorização.

- A sua utilização reflete uma redução de custos nos equipamentos e mão-de-obra. Para além disso, verificam-se ganhos significativos na eficiência de execução das tarefas que a prática agrícola exige.

Até à data não foi possível implementar todas as funcionalidades pretendidas para o sistema final. O *Raspberry Pi* no sistema elaborado torna-lo mais completo, uma vez que este desempenha as funções de um servidor com a vantagem adicional de conseguir recolher dados edafoclimáticos a partir de sensores que este poderá incorporar.

O primeiro sistema testado (ver página 89) demonstrou ser pouco eficaz uma vez que revelou uma grande perda das mensagens provenientes do *Quadrotor*. Um dos tipos de mensagens que foram perdidas foram as mensagens do tipo *acknowledge*. A anomalia por parte de um dos ESC impossíbilitou que fossem testados quaisquer outros parâmetros.

O segundo sistema testado (ver página 92) apresentou um melhor desempenho em comparação com o sistema anterior e executou com sucesso os comandos enviados. Contudo, revelou não ser o mais apropriado visto que houve uma perda significativa de mensagens provenientes do *Quadrotor*, o que levou a que o *Quadrotor* tivesse sido arrastado pela força do vento sem qualquer ação possível sobre ele, logo após a concretização de um comando *Takeoff* por este.

Para além disso, seria igualmente importante realizar mais testes de modo a estudar melhor todo o comportamento do *Quadrotor* ligado ao sistema elaborado em campo agrícola, conseguindo deste modo retirar melhores conclusões e efetuar futuras otimizações.

Como trabalho futuro, seria importante implementar comandos para a aquisição dos dados edafoclimáticos (como temperatura, humidade relativa do ar, intensidade luminosa, etc.) através de sensores incorporados no SBC. Para além disso, seria importante estudar melhor o comportamento do *Quadrotor* conectado com a interface desenvolvida durante esta dissertação, na recolha dos dados edafoclimáticos pretendidos em testes *outdoor*, neste caso específico num campo agrícola.

Referências bibliográficas

[1] https://airandspace.si.edu/exhibitions/wright-brothers/online/fly/1903/. Acedido a 01 de Outubro de 2017. xxi, 6

[2] https://www.solvelight.com/media/products/draganflyer-x4-es-professional-quadcopter-uav-drone-black.png. Acedido a 27 de Setembro de 2017. xxi, 19

[3] http://s1.dmcdn.net/yCxX.jpg. Acedido a 27 de Setembro de 2017. xxi, 20

[4] https://geo-matching.com/upload/2186-general.jpg. Acedido a 27 de Setembro de 2017. xxi, 21

[5] https://http://www.magth.cn/wp-content/uploads/2017/08/magth-2017-08-16_07-02-02_281068.jpg. Acedido a 27 de Setembro de 2017. xxi, 22

[6] https://www.arduino.cc/en/uploads/Tutorial/pwm1.gif. Acedido a 01 de Outubro de 2017. xxii, 29

[7] https://media.giphy.com/media/9PwZ1rOnBuQ2Q/giphy.gif. Acedido a 01 de Outubro de 2017. xxii, 30, 31

[8] https://3drobotics.zendesk.com/hc/article_attachments/202692963/BR-APMPWR-2.jpeg. Acedido a 09 de Setembro de 2017. xxii, 55

[9] https://mavlink.io/en/protocol/command.html. Acedido a 01 de Outubro de 2017. xxii, 66

[10] Alternative UAV Navigation Systems. http://www.electronicdesign.com/embedded/alternative-uav-navigation-systems. Acedido a 01 de Outubro de 2017. 32

[11] Archived:APM 2.5 and 2.6 Overview. http://ardupilot.org/copter/docs/common-apm25-and-26-overview.html. Acedido a 09 de Setembro de 2017. xxii, 51, 52, 53

[12] Archived:LEDs (APM 2.x). http://ardupilot.org/copter/docs/common-apm-board-leds.html. Acedido a 09 de Setembro de 2017. xix, xxii, 53, 54

[13] Battery Failsafe. http://ardupilot.org/copter/docs/failsafe-battery.html. Acedido a 18 de Setembro de 2017. xix, 77, 78

[14] Common Power Module. http://ardupilot.org/copter/docs/common-3dr-power-module.html. Acedido a 09 de Setembro de 2017. 55

[15] Drones conquistam setor agrícola português. http://www.agronegocios.eu/noticias/drones-conquistam-setor-agricola-portugues/. Acedido a 13 de Setembro de 2017. 1

[16] Failsafe. http://ardupilot.org/copter/docs/failsafe-landing-page.html. Acedido a 18 de Setembro de 2017. 76

[17] GCS Failsafe. http://ardupilot.org/copter/docs/gcs-failsafe.html. Acedido a 18 de Setembro de 2017. xix, 79

[18] MAVLINK Common Message Set. http://mavlink.org/messages/common. Acedido a 13 de Setembro de 2017. xix, 40, 41, 43

[19] MAVLink Mission Command Messages . 63, 64, 65

[20] Bruno Barato. Projeto de um sistema de controle para veículos aéreos não tripulados, 2014. xix, 51, 52

[21] Widodo Budiharto. The framework of home remote automation system based on smartphone. pages 53–60, 2015. xix, 52

[22] Luís Filipe Terra Ferreira. Controlo adaptativo de um motor dc, 2010. 30, 31

[23] Fabio García, Ramiro Henao, and Fabio Valencia. Diseño e implementación del sistema de control de vuelo de un uav, 2016. 28, 29

[24] Austin Hughes. Electric motors and drivers - fundamentals, types and applications., 206. 30, 31

[25] DJI Innovations. Dji e300 multirotor propulsion system, 2014. xix, 57, 58

[26] Joseph A. Marty. Vulnerability analysis of the mavlink protocol for command and control of unmanned aircraft. Master's thesis, Department of the Air Force Air University, mar 2014. xxii, 22, 36, 37, 62

[27] Joana Miranda, João Prata, João Paupério, José Neves, José, Pereira, and José Neves. Drones- veículos aéreos não tripulados, 2015. xix, xxii, 32, 33, 34, 35, 36

[28] Sima Mitra. Autonomous quadcopter docking system. Master's thesis, Cornell University, spr 2013. 22

[29] Sebastião Neto, Ruy Oliveira, Valtemir Nascimento, and Ed' Wilson Ferreira. Avaliação de autonomia de baterias recarregáveis para aplicação em rede de sensores sem fio. pages 17–27, 2013. 56

[30] Antoninho Pegoraro. Estudo do potencial de um veículo aéreo não tripulado/ quadrotor, como plataforma na obtenção de dados cadastrais, 2013. 13

[31] José Sousa. Simulação e desenvolvimento de um veículo aéreo autónomo de quatro rotores. Master's thesis, Faculdade de Engenharias da Universidade do Porto, jul 2011. xxi, xxii, 7, 8, 9, 10, 11, 14, 15, 16, 17, 18, 20, 21, 24, 25

[32] Jéssica Sousa. Modelagem e identificação de um veículo aéreo não tripulado do tipo quadrirrotor, nov 2014. xxii, 25, 26, 27, 28, 34

[33] u-blox AG. Neo-6 u-blox 6 gps modules data sheet, 2013. 59

[34] Menno Wierema. Design, implementation and flight test of indoor navigation and control system for a quadrotor uav. Master's thesis, Delft University of Technology, dec 2008. xxi, xxii, 7, 8, 9, 10, 11, 12, 14, 15, 16, 17, 18, 25