UNIVERSIDADE D
COIMBRA

José Pedro Azevedo Sobral de Moura Correia

AUTONOMOUS TARGETING SYSTEM FOR A FIREFIGHTING DRONE

COMPENSATION FOR WIND DISTURBANCE

VOLUME 1

Dissertação no âmbito do Mestrado Integrado em Engenharia Mecânica na área de Produção e Projeto, orientada pelo Professor Doutor Carlos Xavier Pais Viegas, coorientada pelo Engenheiro Mestre Babak Cherheh e apresentada ao Departamento de Engenharia Mecânica da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Outubro de 2021

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

Autonomous Targeting System for a Firefighting Drone - Compensation For Wind Disturbance

Submitted in Partial Fulfilment of the Requirements for the Degree of Master in Mechanical Engineering in the speciality of Production and Project.

Sistema de Mira Automática para um Drone de Combate a Incêndios - Compensação para Efeito do Vento

Author

José Pedro Azevedo Sobral de Moura Correia

Advisor[s]

Carlos Xavier Pais Viegas

Babak Chehreh

Jury

President	**Professor Doutor Miguel Rosa Oliveira Panão** **Professor Auxiliar da Universidade de Coimbra**
Vowel	**Professor Doutor António Manuel Gameiro Lopes** **Professor Auxiliar da Universidade de Coimbra**
Advisor	**Professor Doutor Carlos Xavier Pais Viegas** **Professor Auxiliar Convidado da Universidade de Coimbra**

Coimbra, October, 2021

“My brain is only a receiver, in the Universe there is a core from which we obtain knowledge, strength and inspiration. I have not penetrated into the secrets of this core, but I know that it exists.”

Nikola Tesla

ACKNOWLEDGEMENTS

The development of this dissertation would not be possible without the unconditional support of those who helped me throughout this last year. Whether by technical suggestions or through motivation, all the help was extremely precious for me to be able to complete this thesis.

To begin with, I would like to thank my advisors, Professor Doctor Carlos Viegas and Master Engineer Babak Chehreh, for all their support, availability, and essentially for their teachings, that allowed me to finish this dissertation with levels of knowledge, awareness and critical spirit superior to those I owned at the beginning of this project. To my colleague Gonçalo Rodrigues, who worked closely with me and who also made this dissertation possible through his important suggestions, I would like to extend my gratitude. Aditionally, I thank the funding entities of the project, namely the European Fund for Regional Development through national funds.

Remaining in support of technical issues, I would like to extend my appreciation to Doctor Patrick Neumann, for the wind estimation algorithm developed and for all the help he provided me when I asked him about it, which made it much more accessible to be fully understood. Similarly, I am also grateful to Professor Doctor António Gameiro Lopes and Professor Doctor Almerindo Ferreira, for their suggestions regarding the wind tunnel tests I had to carry out. At a time when it seemed that it would not be possible to obtain relatively precise results, their help was essential to complete this task. I would also like to highlight the roles of Professor Dr. Lino Marques and Mr. Sérgio Ramalho, for their accessibility in allowing me a few days to finish this dissertation, and to my colleague Master Engineer Ehsan Zakeri for his support and for all the knowledge transmitted over the last two months.

To conclude, I would like to express my gratitude to those who have supported me not only this past year, but throughout this entire five-year cycle. Firstly, to my friends José, António, Hugo and João, I thank them for their closeness, trust and loyalty. Lastly, to my family and my girlfriend, Filipa, I keep in my memory all your advice, patience and faith for this step to be completed, and for your daily affection.

Abstract

Currently, largely due to the phenomenon of Global Warming, forest fires have become increasingly common and devastating catastrophes, not only in Portugal but also across the planet. Despite the unsurpassed efforts by firefighters universally to combat these tragedies, it is urgent to implement technological systems that can help (or even replace) these people in these fights, in order to save not only these lives, but also those of all the people possibly affected by fires.

This dissertation aims to introduce unmanned aerial vehicles (globally known as drones) in the field of firefighting in a more active and autonomous way compared to what is currently happening, so that they can combine the inspection and monitoring of fires with the possibility of being the first extinguishing agent, by directing a water jet over them. To this purpose, a system was designed that could fulfill the intended functions, but with the least amount of hardware for that purpose, in order to minimize the vehicle's total weight. These components were selected based on the state-of-the-art for the tasks of local wind estimation (to be considered during the water jet trajectory) and detection of hotspots in a certain area. While this dissertation will focus on the first challenge, the second will be the object of study of Gonçalo Rodrigues' dissertation.

The work carried out and presented in this dissertation focused on the adaptation of an algorithm for determining the local wind, selected from several scientific works in this field, to the required project, to verify if it is possible to design a mission for these unmanned vehicles to estimate the speed and direction of the wind autonomously. All the programming, the communications between hardware, and all the measurements and tests carried out in software and in a wind tunnel, will be the scope of this dissertation. Project results and future developments to be considered will also be discussed.

Keywords [Forest Fires], [Unmanned Aerial Vehicles], [Wind Estimation], [Automation].

Resumo

Atualmente, em grande parte devido ao fenómeno do Aquecimento Global, os incêndios florestais têm-se tornado catástrofes cada vez mais habituais e impetuosas, não só em Portugal como também em todo o planeta. Apesar dos esforços inexcedíveis por parte dos bombeiros universalmente para combater estas tragédias, urge implementar sistemas tecnológicos que possam ajudar (ou até substituir) estas pessoas nestes combates, a fim de salvar não só estas vidas, como também as de todos os possíveis afetados por incêndios.

Esta dissertação tem o intuito de introduzir os veículos aéreos não tripulados (globalmente conhecidos por *drones*) no ramo do combate a incêndios de uma forma mais ativa e autónoma relativamente ao que acontece atualmente, de forma que estes possam aliar a inspeção e monitorização de fogos à possibilidade de serem o primeiro agente de extinção, ao orientarem um jato de água sobre os mesmos. Para tal, projetou-se um sistema que pudesse cumprir com as funções pretendidas, mas que dispusesse da menor quantidade de *hardware* para o efeito, a fim de minimizar o peso total do veículo. Estes componentes foram selecionados com base no estado de arte para as tarefas de medição local do vento (a considerar aquando da trajetória do jato de água) e deteção de pontos quentes numa determinada área. Enquanto que esta dissertação se focará no primeiro desafio, o segundo será objeto de estudo da dissertação de Gonçalo Rodrigues.

O trabalho realizado e apresentado nesta tese, focou-se na adaptação de um algoritmo de determinação do vento local, selecionado entre vários trabalhos científicos na área, ao projeto requerido, a fim de verificar se é possível conceber uma missão para estes veículos não tripulados estimarem a velocidade e direção do vento de forma autónoma. Todo o ciclo de projeto do sistema, desde a programação e das comunicações entre o *hardware* à realização de medições e ensaios em *software* e em túneis de vento, será âmbito desta dissertação. Os resultados do projeto e futuros desenvolvimentos a considerar serão igualmente debatidos.

Palavras-chave: [Incêndios Florestais], [Veículos Aéreos Não-Tripulados], [Determinação do Vento], [Automação].

Contents

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS/ ABBREVIATIONS

List of Symbols

$\boldsymbol{\phi, \theta, \zeta}$ – Roll, Pitch and Yaw Angles

$\boldsymbol{\alpha}$ – Drift Angle

$\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}}, \vec{\boldsymbol{u}}$ – Flight, Ground and Wind Vectors

$\boldsymbol{r_v, r_w, r_u}$ – Flight, Ground and Wind Speeds

$\boldsymbol{\theta_v,\ \theta_w,\ \theta_u}$ – Flight, Ground and Wind Angles

$\boldsymbol{C_D}$ – Drag Coefficient

$\boldsymbol{A_P}$ – Project Area

$\boldsymbol{F_D}$ – Drag Force

$\boldsymbol{m}$ – Vehicle's Mass

$\boldsymbol{\rho}$ – Air Density

$\boldsymbol{g}$ – Acceleration due to Gravity

$\boldsymbol{u}$ – Flow Velocity

$\boldsymbol{W, T}$ – Weight and Thrust Forces

$\boldsymbol{\psi}$ – Inclination Angle

$\boldsymbol{R^2}$ – Coefficient of Determination

$\boldsymbol{M_x, M_y}$ – Moments in the X and Y-axis read by the Force-Torque Sensor

$\boldsymbol{F_{x_{cell}}, F_{y_{cell}}}$ – Forces in the X and Y-axis read by the Force-Torque Sensor

$\boldsymbol{F_x, F_y}$ – Components of the Wind Force in the X and Y-Axis

$\boldsymbol{F_{wind}}$ – Wind Force

$\boldsymbol{\sigma}$ – Slope Angle of the Tilting Structure

$\boldsymbol{a, b}$ – Arms of the $\boldsymbol{F_{x_{cell}}}$ and $\boldsymbol{F_{y_{cell}}}$ forces

Acronyms/Abbreviations

ASCII – American Standard Code For Information Interchange

CAD – Computer-Aided Design

CAGR – Compound Annual Growth Rate

CFD – Computational Fluid Dynamics

EKF – Extended Kalman Filter

ESC – Electronic Speed Control

GCS – Ground Control Station

GPS – Global Positioning System

GUI – Graphical User Interface

I^2C – Inter-Integrated Circuit

IDE – Integrated Development Environment

IMU – Inertial Measurement Unit

IO – Input/Output

IP – Internet Protocol

IR – Infrared

SPI – Serial Peripheral Interface

TCP – Transmission Control Protocol

UART – Universal Asynchronous Receiver/Transmitter

UAV – Unmanned Aerial Vehicle

UDP – User Datagram Protocol

UKF – Unscented Kalman Filter

USB – Universal Serial Bus

1. INTRODUCTION

Since humans began to have access to more and better tools, the demand for carrying out the necessary tasks for their survival with the least possible risk has increased over time. Some good examples of this are hunting tools, machines for all types of work, and transport vehicles. Within the latter, the most prominent are, inevitably, the ones of air type, such as airplanes, which provide extreme security in long travels when compared to any other alternative. Thus, and considering the growing dangers associated with wildfires, it is now necessary to find ways to reduce the human risks related to this theme. The continuous increase in the number of firefighting losses requires rapid and, most importantly, efficient measures. Unmanned aerial vehicles (UAV's) stand out as a viable and reliable solution to this problem, due to their size and ease of control, which allows their use in places that are dangerous and difficult for humans to access.

Therefore, this thesis aims to introduce these vehicles in firefighting in a consistent way, where they cannot only autonomously locate a hotspot but also direct the water jet in that direction. To make it possible, it is important to take into account, firstly, the location of the points with higher temperatures, with the help of onboard sensors. The next obstacle lies in the evaluation of the local wind, in order to anticipate its effect on the jet trajectory.

This research work was partially funded by the Project F3 Desenvolvimento de um sistema de agulheta portante (POCI-01-0247-FEDER-033616), co-financed by the European Fund for Regional Development through national funds (Compete2020 and Portugal2020).

In the next section of the present chapter, the evolution of firefighting techniques over time and the impact of this approach on the market will be discussed in more detail. Then, the objectives of the work are explored, with a view to creating a model that presents advantages over those currently existing on the market, and the chapters and subchapters covered in this dissertation will be introduced and summarized.

a)

b)

Figure 1.1. a) Project logo. b) Funding groups.

1.1. Motivation

Since the beginning of time, the need to control wildfires has been recognized as essential to human life. However, like any engineering project, the reproducibility and efficiency of the various adopted systems have been increasing throughout history. In the third century B.C., Ctesibius of Alexandria, Greece, invented the first manual fire pump, which is represented in Figure 1.2, that served as the basis for most techniques adopted in the following centuries. However, the first fire brigades were only created at the beginning of the 18th century, in France. Thenceforth, other systems have emerged, like the first fire hose in 1672 (Jan van der Heyden, Netherlands – Figure 1.2), the fire hydrant in 1801 (Frederick Graff, USA), and the fire extinguisher in 1813 (George Manby, UK – Figure 1.2).

a)  b)  c) 

Figure 1.2. Some of the most important inventions in the history of firefighting: **a)** the first fire pump; **b)** the first fire hose; **c)** the first fire extinguisher. Adapted from [1].

Nowadays, robotics is playing a key role in the development of new strategies that allow safer and more efficient fire extinguishing. Due to the large area covered in forest environments, it was necessary to implement mobile robotic systems that could be

transported in the well-known fire combat vehicles, in order to help fire brigades. Initially, this automation process of firefighting started with the development of unmanned ground vehicles that, since they could be remotely controlled, allowed to reach more dangerous and closer to the fire areas, with the only restriction between these vehicles and the water tank being the hose's length (although some can have a small tank inside of them).

a)  b)  c) 

Figure 1.3. a) Segway Robotics firefighting robot. Adapted from [1]. **b)** Thermite Robot. Adapted from [2]. **c)** TAF 20. Adapted from [3].

Although these advances are considered interesting in the firefighting industry, it is possible to verify that the technology used in this field is extremely rudimentary when compared to that developed, for example, for war applications. Moreover, global warming has increased the number and intensity of natural disasters worldwide, which forces the continued progress in the search for further advances. This phenomenon has not only raised the average temperature of the planet but has also dried up the forest fuels, which causes a higher rate of fire transmission. Climate change triggers a "snowball" effect, as the expansion of wildfires significantly increases the content of carbon dioxide and other greenhouse gases in the atmosphere, which consequently will contribute to the fastest warming of our planet. Figure 1.4 (Peterson et al. [4]) demonstrates that global warming has led to an evolution of the cumulative burnt forest area in the western United States, from 1985 to 2015, approximately twice as large as would be expected without this phenomenon.

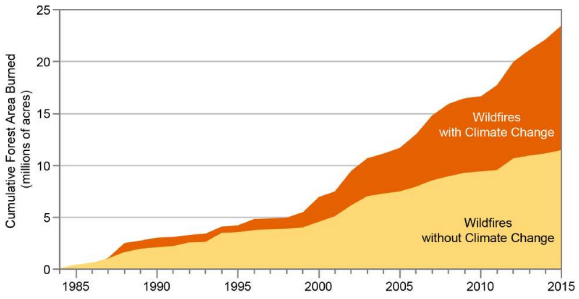


Figure 1.4. Influence of global warming on the evolution of the cumulative burnt forest area in the western United States, between 1985 and 2015. Adapted from [4].

In that sense, it becomes increasingly necessary to find new and solid solutions that own advantages over the antecedents and, mainly, that allow to acquire local data to facilitate fire control and for future study purposes. Thus, unmanned aerial vehicles emerged to fill a gap in this industry related to the gathering of hotspots coordinates and the location of human beings in potential danger, thanks to the implementation of IR sensors in an aerial view, which allows not only to have a better notion of the development of the fire but also to see through smoke and at night. Furthermore, it has also proved to be important in bringing supplies to firefighters or civilians located in areas of difficult access, particularly those that are impossible to reach except by air.

Nevertheless, this approach can still be improved if the vehicle itself can control the fire by dispersing water jets. This is the premise of the project explored in this dissertation, which seeks to study the possibility of incorporating UAVs in firefighting in a more active way, where they cannot only survey the burning area and send data to the command center but also be the first fire control agent, pouring water over the flames. It is presumed that the rate of combated fires will increase significantly with the implementation of a model similar to this one, partly thanks to the shorter time spent moving the hose to the hotspots, due to the possibility of aerial displacement (with fewer obstacles than the ground).

At the date of writing of this dissertation, there are not many solutions similar to the one intended to be designed, although it appears to be a solution that only benefits the industry. It is important to highlight the recent emergence of UAVs with vertical water jets that, despite representing an interesting solution to the problem under analysis, require the vehicle to be positioned precisely above the flame, which, in addition to the possibility of not being feasible due to obstacles, has drawbacks regarding its reliability and maintenance. Alternatively, the implementation of a directable nozzle makes it possible to position the UAV anywhere in the vicinity, thus saving it and its electrical components.

However, perhaps the greatest advantage of this configuration refers to the effect it has on saving human lives, both for civilians and firefighters. The ability to be supervised by just one person, who is also at a considerable distance from danger, represents a major advantage over the most common techniques, where dozens of humans risk their lives to extinguish a fire. Even concerning the widely used firefighting planes and helicopters, the indispensable human presence (allied to the considerable expansion of operating and maintenance costs) constitutes a relevant inconvenience of these aircraft, since the existence

of smoke that can hide trees, overhead power lines, and other obstacles can result in catastrophic consequences for the lives of the pilots who, most of the time, are concerned with the combat itself, neglecting the piloting part. In fact, a large number of fatalities have already occurred within the scope of aerial firefighting, which forces the idealization of strategies that allow reducing these catastrophes. Some worth mentioning recent examples of these took place in New South Wales, Australia (where, in January 2020, a C-130 of *Coulson Aviation*® crashed, leading to three deaths) and in Ponte da Barca, Portugal (August 2020, a *Canadair*® CL215 crashed while refilling its water tank, causing the death of the two pilots).

Ultimately, accordingly to *Allied Market Research*® [5], the current market for firefighting aircraft is estimated at approximately 82 billion euros, with the largest share (about two-thirds) referring to fixed-wing vehicles, such as the common planes and some types of UAVs, partly due to the greater autonomy and speed in their flight, as well as the larger tank inside them. However, it is estimated that this market will grow to more than 98 billion euros by 2026, with a CAGR rate of 3% between 2021 and 2026, under the great investment in rotary-wing aircraft (such as the example of this project), in order to take advantage of the easier control of these systems, as well as the hovering state, which does not exist in the competing genre.

1.2. Objectives

The purpose of this work is to develop a tool for the firefighting industry that seeks to complement the configurations currently existing in the market, which present some deficiencies regarding public safety. The implementation of an automatic target on UAVs will allow tremendous assistance to fire brigades, both in the extinction of active flames and in the prevention of potential fires detected by the search for hotspots.

Therefore, in summary, the creation and development of the desired model will require the correct response to the following challenges:

- Establishment of the communication between the microcontroller and the UAV's flight controller, as well as between the microcontroller and the ground control station;
- Real-time and local wind estimation;

• Implementation of IR and rangefinder sensors, in order to estimate the location of hotspots and height data, respectively;

• Determination of the trajectory imposed on the fluid, taking into account the orientation of the vehicle and the effect of the wind;

• Development of a GUI that allows obtaining in real-time the various parameters of the UAV, as well as controlling its attitude.

However, in this dissertation, only the first two and the last tasks will be explored in detail, with the remainder being objects of study in Gonçalo Rodrigues' thesis. Regarding the several communications required, it is extremely important to guarantee a reliable, error-free, and rapid data transmission, in order to obtain accurate real-time results. For the execution of this task, a program will be developed in the *Arduino IDE*® [6] which, through the *MAVLink*® [7] communication protocol, will allow not only to extract the various flight parameters (angles, speed, etc) obtained in real-time in the flight controller and incorporate them in the developed algorithms, but also to input desired values for the movement of the vehicle.

The wind estimation consists in determining the orientation and intensity of the local wind, which are necessary to correctly define the trajectory of the water jet from the hose to the hotspot. For that purpose, an algorithm that provides the correct calculation of these variables will be implemented.

Finally, the GUI will represent an interface for the direct contact between the control station and the UAV, and it is intended to be intelligible to any possible user. It will be developed in *Microsoft Visual Studio 2019*® [8] and will allow the reading of UAV data in real-time, as well as controlling its attitude, via *Wi-Fi*.

1.3. Outline

The content of this dissertation is divided into four additional chapters. In the next chapter, an analysis will be made regarding the state-of-the-art of UAVs as a firefighting tool, and other scientific works that focus on the calculation of aerodynamic parameters (projected area and drag coefficient) of this type of system.

In the third chapter, which consists of the one that will necessarily be more extensive, all the work carried out in the scope of this dissertation will be exposed, namely

the algorithms, measurements, and tests devised throughout its formulation. Some important notions for a better understanding of the developed work will also be explained, and a GUI that makes it possible to control the UAV and visualize some parameters in real-time will be presented.

Finally, the fourth and fifth chapters concern the presentation of the results obtained in experiments, and the general conclusions of the entire work, respectively. For the latter, the results obtained will be discussed, explaining the obstacles that arose when performing each task, and future works in this field that can be carried out will be identified, namely those that can improve the efficiency and reproducibility of the projected system.

2. STATE-OF-THE-ART

The present chapter is dedicated to the current state-of-the-art concerning UAV applications in firefighting, their aerodynamic properties, and their use as wind field sensing instruments, thus covering the main subjects of this thesis.

In the first subchapter, some firefighting UAVs will be presented and their characteristics will be detailed, with the exhibition of models from different manufacturers. Furthermore, the basic structure of this type of system is also explained, with a special focus on its hardware.

The local values for the speed and direction of the wind, as well as some aerodynamic parameters depending on the UAV's angle of attack, will be addressed in the second and third subchapters, respectively. Some methodologies for obtaining these relevant parameters for the correct implementation of the system will be presented and related to those adopted in the context of this thesis.

2.1. Firefighting UAVs

Although recent, the introduction of UAVs in firefighting has already reached a wide range of different applications. In addition to the solution that is intended to be projected on the direct fire extinguishing and the preceding detection of hotspots, these vehicles have been broadly used to collect terrain data to support the land and fuel management, in order to prevent the occurrence of these catastrophes, as well as for rescue missions in places of difficult access.

Roldán-Gómez et al. [9] distinguish the three steps of a UAV firefighting mission: extinguishing, prevention, and surveillance. Regarding the first, it can be divided into two stages: monitoring (where the vehicle allows the examination of the fire spreading, providing valuable information on how it will evolve) or support to the combat (either directly, by putting out the fire, or indirectly, by supporting the firefighters). The prevention phase can be carried out before (by vegetation mapping) or after (by finding the fire perpetrators) the occurrence of a fire in a given location, and the detection of the factors that

triggered the catastrophe allows to proceed with the surveillance phase. In the latter, the analysis of the most critical areas is carried out, and more inspection measures are applied to these, in order to detect the fire at an early stage of its propagation. In addition to the firefighting mission, these vehicles are also used in analysis, damage mapping, location, and rescue tasks.

2.1.1. Existing Systems

Firstly, it is important to distinguish the two types of vehicles covered in this work: fixed-wing and rotary-wing. Although the first emerged earlier (which is why there are more research works regarding fixed-wing UAVs), the market for the latter is starting to grow a lot. The following table presents the main characteristics of each of these types of aircraft.

Table 2.1. Characteristics of fixed-wing and rotary-wing UAV's.

	Ease to control	Flight autonomy	Compact	Flight speed	Payload capacity	Better use of camera	Hovering
Fixed-wing	-	+	-	+	+	-	-
Rotary-wing	+	-	+	-	-	+	+

a)  b) 

Figure 2.1. a) Fixed-wing UAV. **b)** Rotary-wing UAV. Adapted from [10].

Not all UAVs can be adapted to the firefighting industry, as there is a wide range of characteristics and specifications that these systems must have to succeed in this complex and critical field. These include, naturally, a significant resistance to very high temperatures, the ability to be controlled over long distances (either horizontal or vertical), the existence of built-in sensors (IR, gas, GPS, etc) and cameras (or at least the possibility of adding them as payload), a stable and secure user – flight controller communication that allows data collection and, as the main limiting factor for the use of UAVs in the extinguishing of fires, considerable battery autonomy.

Therefore, some UAVs capable of directly fighting fire will be presented below, which, as will be seen, have in these characteristics (combined with the least possible cost) their strongest advantages in comparison to the competition.

Starting by describing the *Matrice 210 V2* from *DJI*® [11], which represents one of the most recognized solutions globally, it is worth noting the presence of two built-in *Zenmuse* cameras (one IR and one visual), GPS, and the possibility of adding about 1,4 kg of customizable payload. Furthermore, the implementation of a dual-battery system provides greater autonomy and can even reach a total of 38 minutes of flight, which (combined with the operating range of 8 km) makes this one of the most well-known and successful solutions on the market. Other inherent characteristics to the presented model are the resistance to water and strong wind and the installation of anti-collision beacons for situations of reduced visibility and sensors that allow a safer flight, avoiding collisions against obstacles and other aircraft.

Another example worth mentioning is the recent prototype by the Latvian company *Aerones*® [12], which promises to innovate the industry, due to its ability to perform rescue missions thanks to its capability to transport human beings. In fact, this UAV can carry 145 kilograms of weight due to the installation of 28 motors in its structure. It has also the advantages of reaching a maximum height of approximately 300 meters, which allows this to be a very common solution in inaccessible locations (such as tall buildings) to the fire trucks' ladders (about 70 meters), and the possibility of attaching a hose to extinguish the flames and an electric cable to increase the autonomy of the flight. This UAV comes equipped with data sensors, an IR sensor, a radar to avoid obstacles, sixteen batteries, two controllers, and three parachutes in case of flight failure.

a)  b) 

Figure 2.2. Some examples of UAVs used in firefighting. **a)** *DJI Matrice 210 V2*. Adapted from [11]. **b)** *Aerones* firefighting UAV. Adapted from [12].

Finally, it is worth mentioning the project announced by the British enterprise *Faradair Aerospace Ltd*® [13], which, for 2025, proposed to develop an unmanned Air Tanker with 11 m wingspan and 10 tons of payload capacity. Thus, the high volume of water carried by the commonly used airplanes and helicopters is added to the particular safety of UAVs, due to the superfluity associated with the presence of a pilot. These two aspects, combined with the reduced cost of the aircraft compared to that of an aircraft fleet that guarantees a similar capacity, indicate that this solution will be extremely beneficial.

2.1.2. UAV sensing and communication

As is globally known, the physics of a UAV system lies in the rotation of strategically placed motors that, depending on the speed and direction of rotation of each one (controlled by an ESC), allows this type of vehicle to not only move in the cartesian space but also to rotate on all cartesian axes. Additionally, they can be equipped with all types of technology, to obtain data from a location or perform certain actions, depending on the user's needs. Pressure, IR, force and collision avoidance sensors, cameras, and navigation systems (GPS) are some of the most implemented add-ons in a UAV.

However, the main complexity in the development of a model of this type is found in the establishment of communication between the user and the component where it is intended to change or obtain a certain parameter. Thus, it is necessary to add a flight controller that allows, through a messaging protocol (in this project *MAVLink*® was used), a more comprehensible and direct data flow, as well as the ability to send and receive commands and responses to/from various components of the aircraft. The flight controller can collect information of the position and inclination of the UAV thanks to the incorporation of an inertial measurement unit (IMU) which, through the presence of accelerometers and gyroscopes, can obtain and introduce variations in the vehicle's attitude. In addition, magnetometers can also be included to determine the magnetic field and, consequently, the aircraft's orientation relative to the earth's field. Figure 2.3 shows the nomenclatures for the three angles (roll, pitch and yaw) that a UAV can have concerning the reference coordinate system.

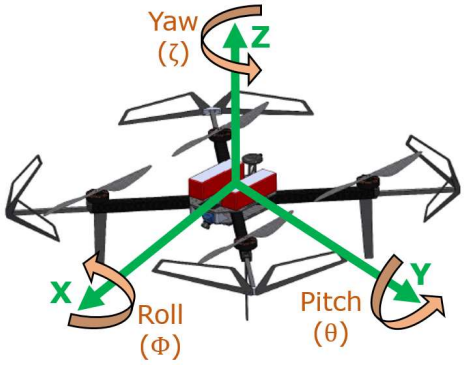


Figure 2.3. Euler angles in an UAV.

2.2. Wind characterization using UAV's

The main premise of this research work is to use the UAV's embedded sensors to automatically assess the wind field, to compensate for its effect on the trajectory of a water jet for firefighting purposes. Any error, whether non-technical or in the formulation/programming, in the implementation of the methodology that will allow the determination of the necessary variables, will lead to the malfunction of the entire automatic targeting system. Thus, in this subchapter, some methodologies (as well as their results) for wind determination in aerial vehicles that are followed in similar studies are demonstrated, as they will serve as a basis for what will be implemented in this project.

Initially, it is important to establish a chronological map regarding the advances in this field, in order to have a better notion of the evolution of this type of study with the technological evolution. It is important to note that the experiments in this field were in accordance with the way the UAV market evolved, since they started being carried out on fixed-wings vehicles, but currently focusing more on the rotary-wings type, due to their greater maneuverability.

The beginning of the work in this area dates to 2003, where Rysdyk [14] developed a set of linear equations that related the ground, flight, and wind speeds, in order to guide the trajectory of the UAV and to synchronize its cameras to a certain target. At the same time, Mokhtari and Benallegue [15] were implementing an algorithm that, using a Lyapunov function, allowed to determine the wind parameters for a quadrotor, based on its Euler angles. In 2005, Kumon et al. [16] carried out a study on how to obtain wind in small fixed-wing UAVs called kite planes, starting from the prior knowledge of the vehicle dynamics.

In 2008, two worth mentioning works appeared, namely that of Palanthandalam-Madapusi et al. [17], where a method that worked as an extension of the unscented Kalman filter was developed, providing an estimation of the wind disturbance, and that of Van den Kroonenberg et al. [18], where a pitot tube was used to determine the wind speed in micro quadcopters. However, the latter proved not to be valid, since the implementation of such a system does not guarantee the best operation in applications with wide variations of direction and low speeds (less than 12 m/s) and requires the wind direction to be aligned with the axial direction of the tube for a precise evaluation.

Lately, the main focus in this type of study lies in the determination of wind parameters using the smallest number of components (with the least weight) possible, in order not to overload the aircraft's carrying capacity. Thus, in 2009, Rodriguez Perez [19] presented an innovative method, which consisted of using an optical flow sensor as a payload that, together with the reading of GPS data and vehicle's speed, allowed obtaining the wind velocity. Two years later, Chan et al. [20] developed three methods that, through the navigation equations, allowed to evaluate the wind components for fixed-wing UAVs, either through direct substitution of the data obtained onboard in these equations or through the use of the two best-known derivations of the Kalman filter (EKF and UKF). The results of this work showed that all of these approaches provide reliable and effective solutions, although the methods in which the Kalman filter is applied are more accurate than the first one (but require also a greater computational effort), as seen below.

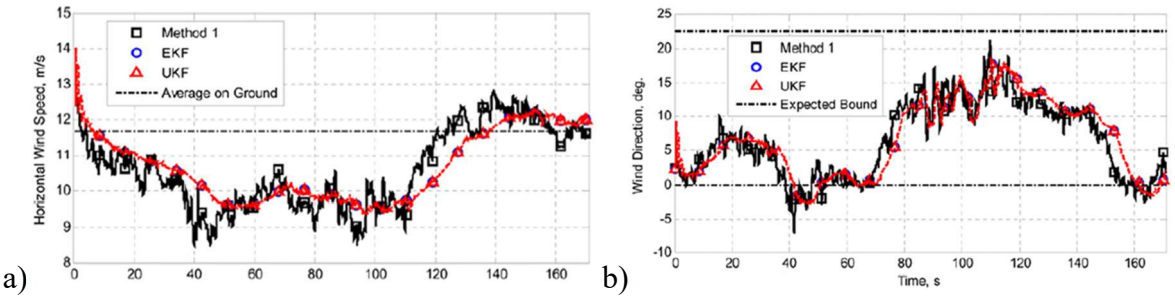


Figure 2.4. Results obtained by Chan et al. for the three methods, as a function of time, for: **a)** horizontal wind speed; **b)** wind direction. Adapted from [20].

In 2012, Mayer et al. [21] worked on a methodology that, through software simulation, focused mainly on processing data from the GPS and the flight controller, namely the vehicle's speed and the flight path azimuth.

Between 2010 and 2015, Neumann et al. [22][23][24] developed an algorithm that would allow estimating the local wind based exclusively on the data obtained onboard (IMU, GPS, etc), with the aim of implementing it in a gas distribution mapping system. The roots of this methodology refer to the wind triangle where, based on Figure 2.5.a), it can be concluded that the direction and intensity of the wind can be determined from two other vectors: the ground (effective displacement of the vehicle) and flight. The ground vector is obtained directly from the UAV's GPS receiver, but the flight vector requires a greater number of calculations and experiments to be estimated. The experiment consisted in measuring, in a wind tunnel, the inclination angle due to the wind speed, and the subsequent formulation of calibration functions that relate these two variables. Finally, in outside experiments, the results obtained with the implemented system were compared with the values measured by an anemometer, and this comparison is shown in Figure 2.5. The main advantages of this work are the fact that it can be conducted with minimal knowledge of the dynamics of the aircraft, and that there is no need to add much hardware to its payload. A particular case of this methodology for circumstances where it is possible to set the yaw angle to 0º was developed by Palomaki et al. [25] in 2017. From measurements in indoor experiments, it was estimated a correction deviation to the effect of the rotors' movement of -0.5m/s.

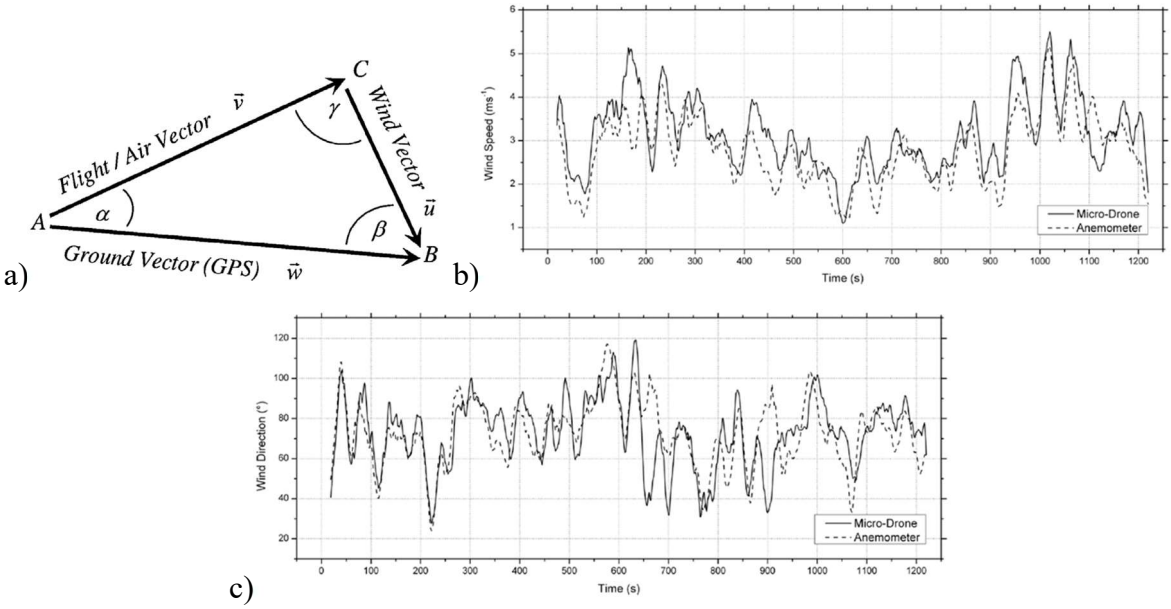
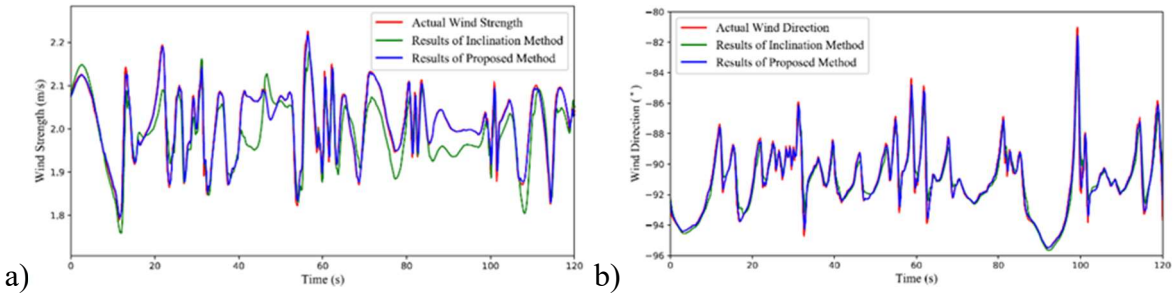


Figure 2.5. a) Wind triangle; **b)** Evolution of the wind speed. **c)** Evolution of the wind direction obtained in the experiment of Neumann et al.. Adapted from [23].

Another work that deserves to be mentioned is that of Marino et al. [26] who, in 2017, idealized a procedure that consisted in measuring the power consumed by the motors for certain intensity and incidence angle of the wind, in order to plot a correlation between these parameters. Therefore, it is only necessary to employ current sensors in the equipment's payload, which are often inherent to its constitution. The achieved results do not present good viability for high wind speeds (above 25 km/h), however, it is stated that this solution can be improved in future research, for example, by evaluating an ideal position of all motors that maximizes the efficiency of the method.

In the following year, Wang et al. [27] presented a more detailed approach to determining wind speed and direction, especially regarding the forces applied to the UAV (thrust and drag). Therefore, it was possible to estimate the effect of the angular speed of the rotors in obtaining those parameters, by comparing the solutions obtained through this method with those of the direct treatment of the data obtained onboard (Inclination Method). As seen in Figure 2.6, for the hovering state, this approximation influences, particularly, the wind speed results.



a) b)

Figure 2.6. Comparison between the results obtained for each method with the actual wind, in the work of Wang et al., for: **a)** wind strength; **b)** wind direction. Adapted from [27].

Recently, studies with different configurations have emerged, aiming to improve the effectiveness and accuracy of wind measurements, although compromising the payload capacity. Some examples are the works of: Vasiljević et al. [28], who implemented a set of wind *LIDAR*® sensors; Adkins et al. [29], where a strategy of equipping the aircraft with an FT205 wind sensor was adopted; and Thielicke et al. [30], who chose to use a lightweight ultrasonic anemometer. All of this research provided very reliable results, which shows that there is no fixed configuration for this type of study. Thus, for each case, a rule of thumb should be performed, relating an acceptable approximation level with possible cost and accessibility problems (such as the knowledge of the aerodynamic behavior of the aircraft).

2.3. UAV aerodynamics

In this subchapter, several calculation techniques developed in research works will be explored, both for the projected area and for the drag coefficient, in certain flow intensity and direction. Although these two parameters are objects of study in the same works, this subchapter is divided into two sections, where each of these properties will be individually focused.

2.3.1. Projected Area

Primarily, the concept of the vehicle's projected area will be addressed, which will be necessary for the determination of the drag coefficient, which will be investigated in the next section. In fact, the first parameter varies not only with the wind direction, but also with the orientation of the UAV (yaw angle), so, in theory, there will be a huge number of different values, due to all possible combinations between these angles. Consequently, it is extremely important to seek the best way to approximate these results to the entire range, while reducing the difficulty, cost, and time spent in this estimation.

The first method of projecting the area of a particular body onto a plane is called the "shading algorithm" and is fully explained in the work of Woo and Poulin [31], from 1990. It consisted of focusing light rays with a specific orientation against an object and observing the shadow that was formed, which corresponded to the projected area in that direction. Currently, all methodologies for determining the projected surfaces rest, directly or indirectly, on this principle. Although the calculation procedure does not present a wide range of possible adaptations and there are not many different works in this field (particularly about UAV's), some of these on the most various subjects that were considered relevant will be exposed below.

In 1998, Brown and Vickers [32] studied the variation of the projected area of particles by changing the view orientation, in order to make the works on diffraction, reflection, and refraction of electromagnetic radiation that interact with these particles more effective. As expected, since it was concluded that it is unfeasible to analyze all surfaces for all infinitesimal variations of the incident angle, the objective was to estimate values of areas that, depending on the orientation, represented the real results within a predefined range. Naturally, the more measurements are taken, the shorter the confidence interval. This methodology will serve as the basis for the work that will be carried out in this dissertation,

as it can be considered a good agreement between accuracy and accessibility. Finally, it is important to demonstrate the case studied in this work, where it was formulated a distribution function that would allow to approximate the projected area of a cube for all its possible orientations. As seen in Figure 2.7, for a total of twelve view positions, a good agreement for the calculation of this parameter was accomplished.

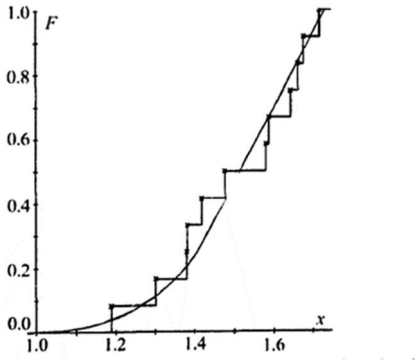


Figure 2.7. Comparison between the distribution function formulated by Brown and Vickers and the real results for the projected area for twelve orientations of the cube. Adapted from [32].

Over time, with the technological evolution, many software capable of facilitating the determination of the projected area in different configurations have emerged. Allied to the need to study different and complex forms, these made it possible to carry out research works in the most diverse areas. An example of this occurred in 2010, when Sagnes [33] concluded that, in a study of flexible bodies (freshwater macrophytes), the use of a multiple scale reduced the error in determining areas by 20%, compared to a single scale.

In fact, most applications that require the calculation of an object's projected area are hydro or aerodynamic, and usually for determining the drag coefficient (discussed in the next section). Ben-Yaacov et al. [34], in 2015, developed a methodology to calculate these two parameters analytically on a satellite, to estimate its related torques. This procedure stands out because the satellite's attitude is time-dependent, so the projected area is a function of time. The results obtained for the area and torque present a good approximation to those achieved either through software or through a formulated numerical method.

An example of a work that differs from the aerodynamic applications already exposed, is that of Voltz et al. [35] who, in 2017, sought to calculate the contact area between the indenter and the specimen, in a nanoindentation experiment. This approach deserves

mention because the Finite Element Method was employed, specifically by identifying the specimen nodes in contact with the indenter, as well as their coordinates.

In conclusion, it is important to clarify that, in most cases, the shapes of the objects to be projected are quite complex, which is why it is common to approximate these shapes to more regular ones, such as plates, spheres, ellipsoids, prisms, cubes, cylinders, etc. One example is the case studied in the work of Kljuno and Catovic [36], in 2018, where a fragment of a projectile was approximated to a tri-axial ellipsoid. A physical model was formulated, and its results for the projected area in each direction were subsequently compared with those achieved using CAD software, where the viability of the model was verified, as it is shown in Figure 2.8. In the particular case of a UAV, and since it is an equipment with several bodies of distinct and very complex geometry, it is not viable to adopt a strategy of this type, as each body would have to be treated separately.

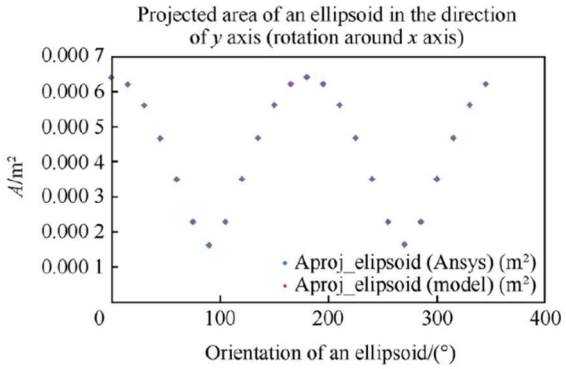


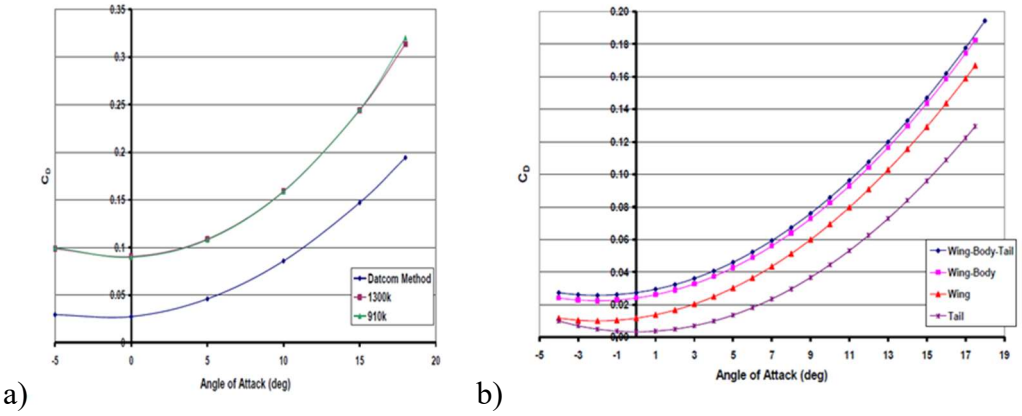
Figure 2.8. Evolution of the projected area of the ellipsoid with its orientation calculated by the physical model and by CAD software. Adapted from [36].

2.3.2. Drag Coefficient

In this section, some methodologies for determining the drag coefficient of UAVs will be exposed, as well as the most adopted assumptions to facilitate the implementation of an effective model.

The first to be presented appeared in 2008 by Israr and Dahalan [37] and sought to compare the results obtained for the lift and drag coefficients by an analytical method (Datcom) with those provided by a fluid simulation software, for several inclinations of the fixed-wing UAV, and with two distinct meshes (there were no practical benefits from refinement). Although the achieved results for the first parameter were adequate, it was necessary to add a deviation to the curve obtained by the analytical method (Figure 2.9), in

order to correctly express the evolution of the drag coefficient with the angle of attack. This difference was due to the fact that the followed method is more ideal for applications with higher speeds. In an additional experiment, the drag coefficient was measured by the analytical method for the various bodies of the vehicle, and the graph presented in Figure 2.9 was obtained, which shows the influence of the main body and wings in relation to the tail, in the calculation of total drag.

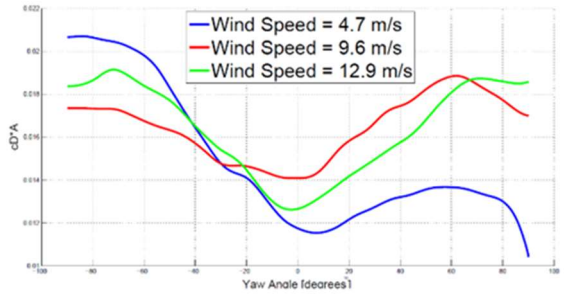


a) b)

Figure 2.9. Evolution of the drag coefficient with the angle of attack: **a)** for the entire equipment, varying the method. **b)** for the analytical method, varying the bodies Adapted from [37].

In 2014, Schiano et al. [38] developed a procedure that stands out for grouping the drag coefficient to the projected area of the UAV in a single parameter. This strategy was due to the difficulty of accurately quantifying the area for each orientation of the vehicle, because of its complex shape. A similar strategy was adopted for the lift coefficient, although it was concluded that this parameter was much lower than the first (Figure 2.10). Through the implementation of a force sensor that allowed to determine the forces applied to the vehicle (drag, lift, and thrust), it was possible to carry some experiments. For three different wind intensities, only variations of the yaw angle were tested firstly (where the thrust equals the weight of the aircraft) and, subsequently, the same process was repeated but only for variation in the pitch angle. From Figure 2.10, it can be concluded that the wind tunnel test is not plausible for reduced wind speeds (laminar flow) since the respective curve is relatively different from the rest and presents very different results for ± 90º. Also, for this UAV, the parameter to be determined ($C_D * A$) can be considered as a function of only the yaw angle, since it is quite stable for different pitch angles.

a)



b)

Pitch Angle (α)	$V_w = 12.9$ m/s	
−10°	$c_D A = 0.0144$	$c_L A = 0.0052$
0°	$c_D A = 0.0118$	$c_L A = 0.0048$
10°	$c_D A = 0.0146$	$c_L A = 0.0033$
20°	$c_D A = 0.0140$	$c_L A = 0.0002$
30°	$c_D A = 0.0141$	$c_L A = 0.0008$

Figure 2.10. a) Evolution of $C_D * A$ with the yaw angle, for three different wind speeds. **b)** $C_D * A$ and $C_L * A$ for different pitch angles, with a wind speed at 12.9 m/s and a yaw angle of 0°. Adapted from [38].

Three years later, Felismina et al. [39] set out to elaborate the flight plan that would maximize the autonomy of the batteries of their UAV, which was designed for agricultural applications. Their approach intended to determine the various orientations of the UAV that, for the different movements of take-off (climbing) or stable flight, guaranteed the least possible drag (drag coefficient and contact area) and the lowest pressure in the battery area. For this purpose, fluid simulation software was used to replicate the tests in a wind tunnel, to estimate the values of the projected area and the drag coefficient for each vehicle's orientation studied.

In the two following years, two other important works were conducted in order to compensate for some aerodynamic effects, namely the induced drag, which affects the position tracking of the UAV when it moves at high speeds. Svacha et al. [40] formulated an algorithm that, by applying commanded forces and moments in the vehicle, made it possible to reduce the error in this tracking, as shown in Figure 2.11. This procedure also stands out for estimating the drag coefficient through the onboard accelerometer. A similar procedure was adopted by Faessler et al. [41] who, through an iterative process, determined the drag coefficient as the value that minimized the tracking error (Figure 2.11). In this case, circular and lemniscate movements of the UAV were tested, and it was confirmed that the implemented algorithm resulted in an approximation of the tracking to the actual motion.

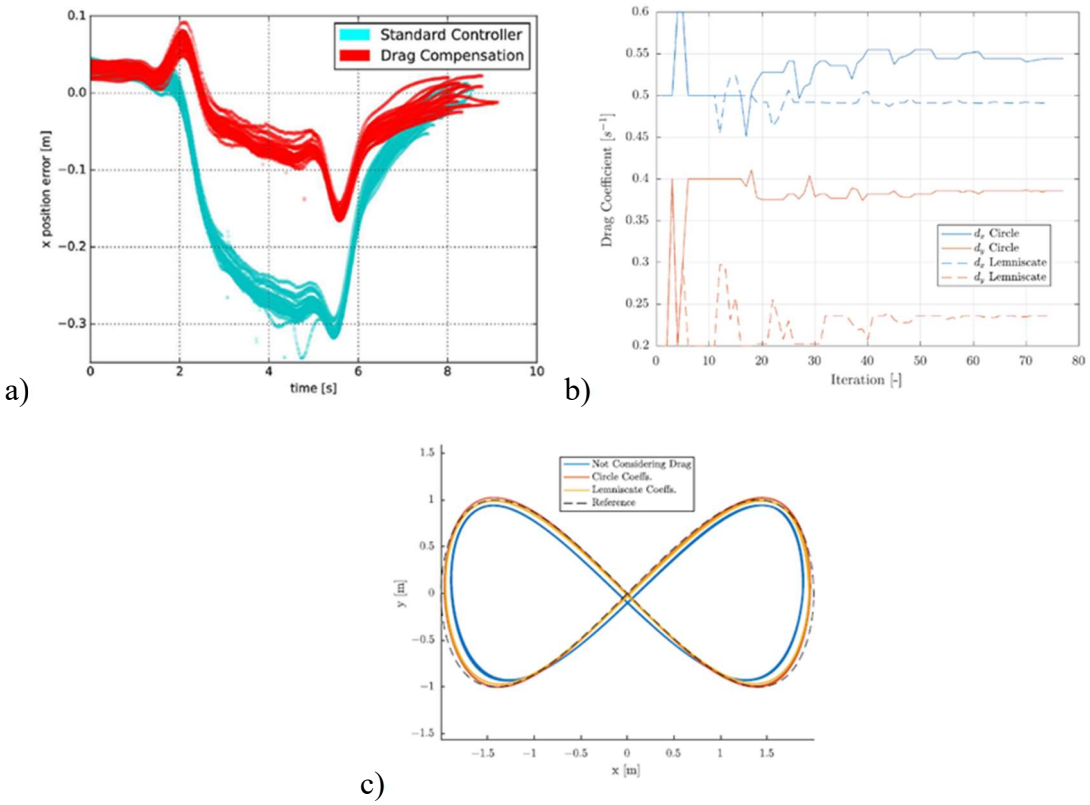


Figure 2.11. a) Tracking error with and without drag compensation. Adapted from [40]. **b)** Evolution of the drag coefficient with the iterative process in both X and Y directions, for each trajectory. Adapted from [41]. **c)** Comparison between the tracking positions in the lemniscate trajectory (with different ways of estimating the drag coefficient) with the actual movement of the UAV. Adapted from [41].

To conclude, it is highlighted the work of Götten et al. [42] of 2020 where, although UAVs are not directly addressed, a rigorous study is presented on the determination of drag in some components of an unmanned fixed-wing vehicle, namely landing gears, wheels, and a turret sensor. The measurements took place inside a wind tunnel where, with the aid of a force sensor, the drag applied to each component was determined by subtracting the force obtained for the equipment without the component under consideration, from the achieved result for the complete system. The accuracy of this approach was confirmed through software simulation. Regarding the achieved results, it was confirmed that streamlining the bodies would result in significant decreases in the drag coefficient. In the particular case of the sensor, it was also concluded that, despite its shape was similar to a cylinder, the drag applied in it was much higher than that applied in this geometric solid, due to its rotation movement. It was also deduced that a greater concern in the selection of the position and orientation of the sensor allows decreasing the drag in the equipment.

3. WORK DEVELOPMENT

This chapter is dedicated to the exhibition of the work methodology and each step that constitutes it. In tasks that involve measurements, their procedures will also be presented and discussed. In addition to that, the components added to the UAV as payload will be summarized and their main characteristics will be highlighted, thus justifying each choice instead of their competitors.

3.1. Methodology

This subchapter will briefly present the procedures and experiments to be carried out to correctly implement the intended system. The following diagram (Figure 3.1) shows not only the planning of the challenges to overcome but also the hardware used in each task. This dissertation will focus mainly on the left part of the diagram, which consists of determining the parameters related to the wind, as well as the wired communications between the UAV components and the wireless one between the UAV and the ground station, with special emphasis on the subsequent automatism of the vehicle's attitude to be aiming at the target. The right part of the diagram, related to the location of hotspots by the IR sensor, will be addressed in a complementary thesis, formulated by Gonçalo Rodrigues.

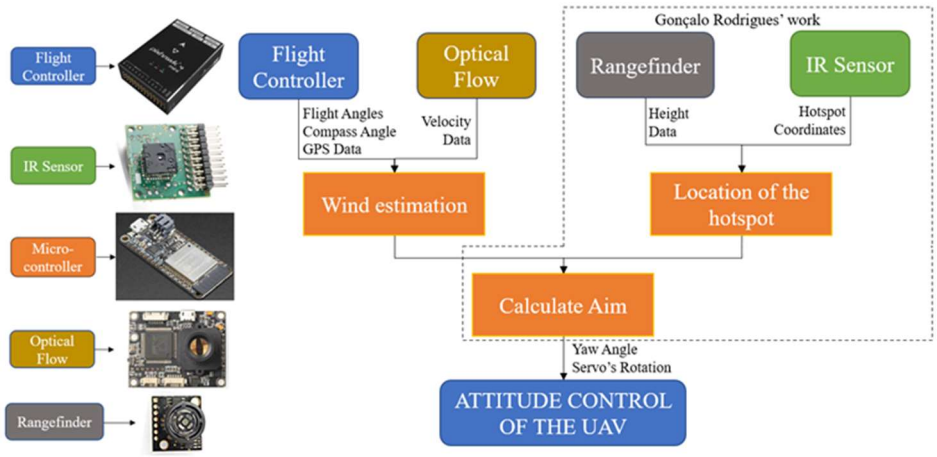


Figure 3.1. System context diagram. As shown on the left side of the figure, each color in the diagram corresponds to a different component (the Optical Flow and the GPS are connected to the Flight Controller).

All the accomplished procedures will be listed below, and a more detailed description of each one will be carried out, in order to clarify all the challenges that have arisen and what has been done to overcome them.

Communication between the microcontroller and the flight controller:

The correct definition of the parameters to obtain from the flight controller and how to ask for them is essential to the proper functioning of the system that is intended to be implemented. Thus, in this step lies the backbone of the work. For its resolution, the *MAVLink*® [7] communication protocol was used, allowing to follow a predefined message structure, which makes it easier to obtain the IMU data (particularly the pitch and roll angles), the compass angle (UAV orientation in relation to the magnetic north pole) and the GPS speed and direction. In addition, since it will be necessary to control the vehicle's attitude from the ground station, the accurate and safe data flow in the opposite direction is also required, so the implementation of a bidirectional flow is indispensable. Finally, it is relevant to point out that several data acquisition rates were tested in all experiments, in order to obtain the best compromise for the data flow in each case.

Determination of UAV aerodynamic parameters:

In the calculation of the flight vector, there is a need to estimate some aerodynamic parameters that cannot be taken directly from the flight controller, such as the projected area and the drag coefficient for a certain orientation. For the first one, a 3D-CAD software (namely the *Autodesk Inventor Professional 2021*® [43]) was used, where it was possible, from a previously designed CAD model of the system, to project the area of the latter for different directions and inclinations.

Regarding the drag coefficient, experiments were carried out in a wind tunnel, using a load cell, to determine the drag force on the equipment. Then, using Eq. (3.1), it was possible to determine C_D for each test. The tests were repeated for different orientations, in order to estimate the range of values that this parameter can assume for any real conditions.

$$C_D = \frac{2 * F_D}{\rho * A * u^2} \tag{3.1}$$

Implementation of an algorithm for wind speed and direction estimation:

In this task, it is intended to integrate all the previously determined parameters (both aerodynamic and those obtained from the flight controller) and implement an algorithm that allows estimating the local speed and direction of the wind, fundamentally based on the wind triangle. This algorithm was sought out among several studies in the field of wind estimation in UAVs (particularly those approached in Chapter 2) and selected based on efficiency and, essentially, on the proximity of the application and the payload added to the system, being later adapted for the intended project. To improve the efficiency of the model, fine-tuning and calibration techniques were used.

Implementation of a GUI for reading in real-time the values of each parameter, and for controlling the vehicle's attitude:

In order to control the UAV operation according to the values obtained for the various parameters and the user's intention, a Graphical User Interface (GUI) was implemented. The main goal of this task is to formulate an intelligible and easy-to-handle interface for any operator, with the greatest possible number of events, to improve the control of the vehicle's attitude and to allow the reading of all relevant parameters. Thence, the first challenges were to define a wireless communication protocol, to guarantee the message flow between the system and the ground station, and a generic command model to facilitate data transmission.

Several functionalities can be tested with a system of this type, such as the live displaying of the camera view, or the real-time exhibition of the flight angles and speeds values. Therefore, the range of possibilities that can be implemented is vast.

Experiments to test the validity of the system

After programming the algorithm and establishing all necessary connections, it was important to carry out the testing phase to examine whether the obtained results are satisfactory or not. Therefore, some experiments were idealized in order to verify if the correct hardware implementation and the respective connections between the various components were achieved.

With the help of a hoist, the aircraft was supported at a certain height to measure distances to a hotspot (heat gun). These tests allowed to check whether the scripts employed to read and transmit the coordinates of this hotspot were correct, and to confirm whether the various possible services in the GUI work properly.

3.2. Hardware Overview

At this point, it is relevant to present the UAV that was being used throughout the project. It is a custom-made system that weighs about 3,2 kilograms with batteries included, supports a payload of 2 kg, and has a flight autonomy of approximately 10 minutes. Another note worth mentioning is the existence of propeller guards, which guarantee a higher level of safety for the aircraft. Some illustrative images of the onboard electrical components of the vehicle are shown in Annex A.

In order to understand why the several electrical components were implemented in the UAV, some basic notions and important features of them will be introduced. Starting with the flight controller, the nuclear element of the flight and responsible for all low-level communication, a *Pixhawk® 4 Mini* [44] was used. In fact, the main advantage of this model is, as the name suggests, its high compactness (38x55x15.5 mm), which makes it ideal for small projects. This *PX4®* autopilot, developed in association with *Holybro®* and *Auterion®*, is incorporated with all types of necessary sensors for its control and supervision, such as two redundant accelerometers and gyroscopes (ICM-20689 and BMI055), a pressure sensor (MS5611), and a magnetometer (IST8310), in addition to the indispensable GPS receiver (*u-blox Neo-M8N*).

Regarding the microcontroller, an *Adafruit HUZZAH32 – ESP32 Feather®* [45] was chosen to handle the high-level communication, particularly because it represents an extremely powerful and compact solution, which are mandatory attributes for the idealized application. This 32-bit *Feather®* board was developed by *Espressif Systems®* and possesses a built-in dual-core *ESP32* chip, which allows wireless connection to be established via either *Wi-Fi* or *Bluetooth* and represents a major upgrade over the previous model (*ESP8266*), mainly due to the higher speed of its processor. Besides, it is possible to program this controller using the open-source *Arduino IDE®*, which is a familiar and comfortable environment when it comes to working with microcontrollers. Other relevant features of this

component are the existence of several digital and analog IO pins, a clock frequency of 240MHz, and the connections for different communication protocols (UART, SPI, and I2C).

The servo motor that is responsible for the movement of the nozzle is the SC-1251MG series from *Savox*® [46]. The main advantages of this component in relation to its competitors lie in its aluminum center case (to increase cooling, due to the high supported temperatures) and in the high precision and speed of response, since it is intended that the water jet precisely reaches the hotspot, which is why it is necessary the exact rotation of the servo. This actuator also provides high compactness when compared to others in the market with the same torque capacity (9 kg/cm).

The other components that operate in this project are the optical flow sensor, where the *PX4Flow*® [47] sensor was used to estimate the velocity indoors, since it was designed to work with a *Pixhawk*®, a *MB1043 HRLV-MaxSonar-EZ4* [48] rangefinder, and the *FLIR Lepton 3.5*® IR sensor [49], both selected due to their reduced size and cost but high resolution. It should be noted that the latter is coupled to a *Lepton Breakout Board V2.0*® [50] that has the SPI protocol, which is necessary for the transmission of each frame, and that it is connected to a *Raspberry Pi 4*® [51], since it was not possible to establish a direct connection between this board and the *ESP32*. Afterwards, the hotspot coordinates are read from the *Raspberry Pi*® and then transmitted to the microcontroller via serial communication – USB. For the calculation of the hotspot coordinates, scripts were adapted from a repository created for this sensor [52] which enables the generation of the video stream. Two extra scripts (Appendix A) were created to send these coordinates to the microcontroller and to stream the video to the GUI.

Finally, all that remains is to present the connections between all the components that constitute the UAV hardware, which are shown in Figure 3.2. As it can be seen, the microcontroller is indirectly connected to the optical flow (it is connected to the Flight Controller via I^2C connection) and to the IR sensor.

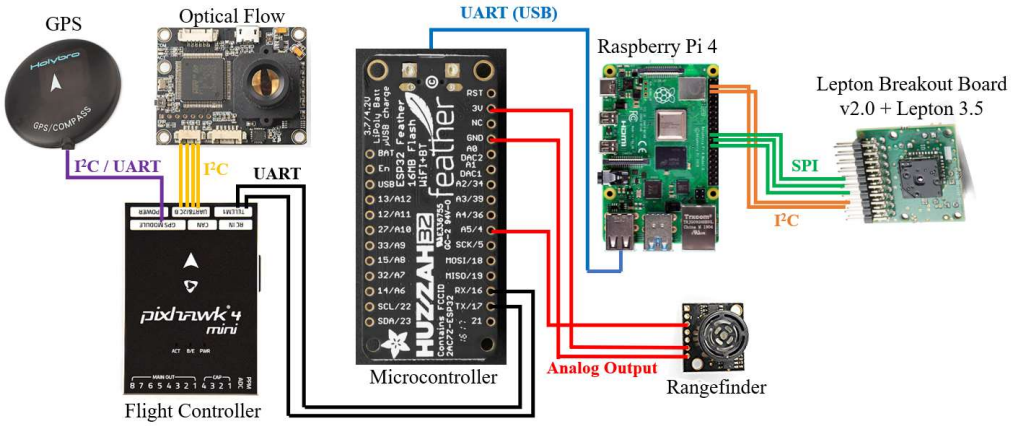


Figure 3.2. Demonstration of the connections between components.

3.3. Algorithm Development

Since the core of this work lies in the correct (local) determination of the wind components, it becomes extremely necessary to implement an algorithm that allows obtaining these results with a satisfactory precision-accessibility ratio. Furthermore, the drawback of adding elements as payload should be taken into account, so the selected algorithm must require the least amount of hardware possible. Among several methodologies found in works similar to the one intended to be implemented, the algorithm presented by Neumann and Bartholmai [23][24] was chosen, which only requires the installation of the IMU (with a built-in magnetometer) and the GPS module.

In fact, with the intention of analyzing different procedures using the same mathematical formulation, two different methods for estimating the wind were devised, both based on the wind triangle. For both, it is assumed from now on that the wind will have only a horizontal component, parallel to the ground.

1st method:

The first method focuses on an approach similar to the one proposed in the specified work [24], where the objective is to estimate locally the speed and direction of the wind, whatever the angle between its incidence and the front of the vehicle. This procedure converges with the basic idea of the project, to allow the UAV, in a continuous and autonomous way, to evaluate possible variations in the wind in a flight cycle.

This algorithm derives from the wind triangle (Figure 2.5), where the data provided in real-time by the GPS (ground vector) and the angles by the IMU, allow the calculation of the remaining vectors of the triangle. Figure 3.3 displays the flowchart of this method, where the starting parameters are highlighted in orange and the final ones in yellow.

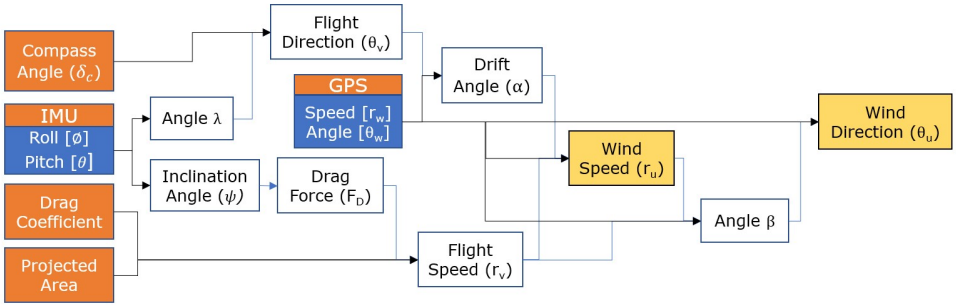


Figure 3.3. Flowchart of the 1st method algorithm.

As shown in the vehicle's force balance for an ideal hovering posture (the position that will be assumed for wind estimation, although the vehicle can't be completely stationary, due to the drift phenomenon) in the presence of wind (Figure 3.4.a)), the drag force can be calculated by trigonometry, knowing the weight and thrust vectors. Since the vertical component of the latter is opposed to the weight of the UAV, the drag force is given by Eq. (3.2). The influence of the rotors on non-zero angular speed was not considered, since it is not possible to measure it in the wind tunnel experiments.

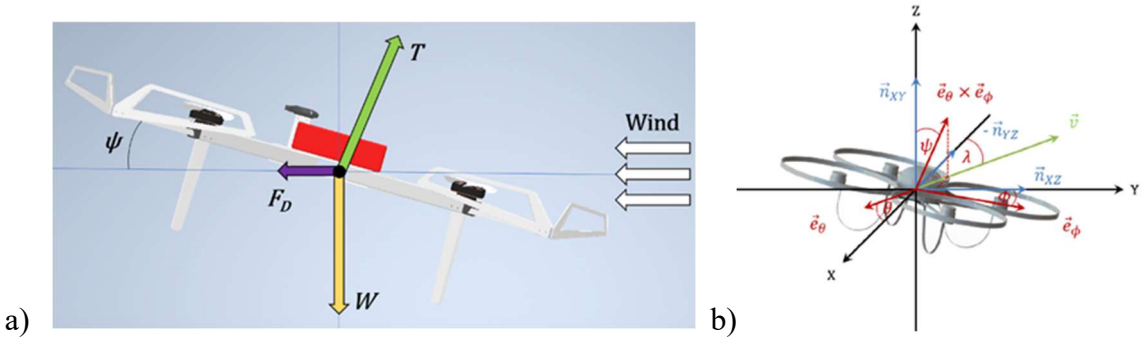


Figure 3.4. a) Force balance in the UAV, for the hovering state. **b)** Coordinate system of the UAV. Adapted from [24].

$$F_D = m \cdot g \cdot tan\,\psi \tag{3.2}$$

Naturally, in the hypothetical absence of wind, this force would not exist either, so the thrust would be vertical. The inclination angle, according to Figure 3.4.b), represents a combination of roll ($\emptyset$) and pitch (θ) angles and is obtained through the following formula, where $\vec{n}_{XY}$ constitutes the unit vector perpendicular to the ground (0,0,1):

$$\psi = cos^{-1}\left(\frac{\vec{\boldsymbol{n}}_{\boldsymbol{XY}} \cdot (\vec{\boldsymbol{e}}_{\boldsymbol{\theta}} \times \vec{\boldsymbol{e}}_{\emptyset})}{|\vec{\boldsymbol{n}}_{\boldsymbol{XY}}| \cdot |\vec{\boldsymbol{e}}_{\boldsymbol{\theta}} \times \vec{\boldsymbol{e}}_{\emptyset}|}\right) \tag{3.3}$$

This figure also shows that the vectors $\vec{e}_\theta$ and $\vec{e}_\emptyset$ are expressed, respectively, by $(\cos\theta\,, 0, -\sin\theta)$ and $(0, \cos\emptyset\,, \sin\emptyset)$, so their cross product is determined as follows:

$$\begin{aligned}\vec{\boldsymbol{e}}_{\boldsymbol{\theta}} \times \vec{\boldsymbol{e}}_{\emptyset} &= \begin{vmatrix} \hat{\boldsymbol{\imath}} & \hat{\boldsymbol{\jmath}} & \hat{\boldsymbol{k}} \\ cos\,\theta & 0 & -sin\,\theta \\ 0 & cos\,\emptyset & sin\,\emptyset \end{vmatrix} = \\ &= sin\,\theta\,cos\,\emptyset\,\hat{\boldsymbol{\imath}} - cos\,\theta\,sin\,\emptyset\,\hat{\boldsymbol{\jmath}} + cos\,\theta\,cos\,\emptyset\,\hat{\boldsymbol{k}}\end{aligned} \tag{3.4}$$

Therefore, from Eq.(3.3), it is possible to deduce the inclination angle equation in canonical form:

$$\begin{aligned}\psi &= cos^{-1}\left(\frac{(0,0,1) \cdot (sin\,\theta\,cos\,\emptyset\,, -cos\,\theta\,sin\,\emptyset\,, cos\,\theta\,cos\,\emptyset)}{1 \cdot \sqrt{(sin\,\theta)^2(cos\,\emptyset)^2 + (-cos\,\theta)^2(sin\,\emptyset)^2 + (cos\,\theta)^2(cos\,\emptyset)^2}}\right) \\ &= cos^{-1}\left(\frac{cos\,\theta\,cos\,\emptyset}{\sqrt{(sin\,\theta)^2(cos\,\emptyset)^2 + (cos\,\theta)^2}}\right)\end{aligned} \tag{3.5}$$

Knowing the vehicle's projected area and drag coefficient (A_P and C_D, respectively, which will be addressed in the next subchapter) for the calculated inclination angle, and by the definition of drag, it is now possible to estimate the flight speed:

$$r_v = \sqrt{\frac{2 \cdot F_D}{\rho \cdot A_P \cdot C_D}} \tag{3.6}$$

Looking now at the upper part of the flowchart, it is important to explain how the flight direction is obtained. The angle λ, which represents the angular distance between the front axis of the UAV (X-axis, in Figure 3.4.b), with $\theta = \zeta = 0°$), and the projection of the inclination vector in the XY plane (where Y is the axis whose angle that can be different from zero is θ, i.e.: $\emptyset = \zeta = 0°$), is determined from Eq.(3.7), according to [24].

$$\begin{aligned}\lambda &= cos^{-1}\left(\frac{\vec{\boldsymbol{n}}_{\boldsymbol{YZ}} \cdot (\vec{\boldsymbol{e}}_{\boldsymbol{\theta}} \times \vec{\boldsymbol{e}}_{\emptyset})_{XY}}{|\vec{\boldsymbol{n}}_{\boldsymbol{YZ}}| \cdot |(\vec{\boldsymbol{e}}_{\boldsymbol{\theta}} \times \vec{\boldsymbol{e}}_{\emptyset})_{XY}|}\right) \\ &= cos^{-1}\left(\frac{(1,0,0) \cdot (sin\,\theta\,cos\,\emptyset\,, -cos\,\theta\,sin\,\emptyset\,, 0)}{\sqrt{(sin\,\theta\,cos\,\emptyset)^2 + (-cos\,\theta\,sin\,\emptyset)^2 + 0^2}}\right) = \\ &= cos^{-1}\left(\frac{sin\,\theta\,cos\,\emptyset}{\sqrt{(sin\,\theta)^2(cos\,\emptyset)^2 + (cos\,\theta)^2(sin\,\emptyset)^2}}\right)\end{aligned} \tag{3.7}$$

In order to estimate the flight direction, all that remains is to take the compass angle (δ_c) received from the IMU, and use one of the following formulas, depending on whether the inclination angle is to the left of the X-axis or not. Mathematically, this condition is analyzed by verifying if the projection of the inclination angle in the XY plane ($(\vec{e}_\theta \times \vec{e}_\emptyset)_{XY}$) has the same direction as the one considered positive for the Y-axis ($\vec{n}_{XZ}$).

$$\theta_v = \begin{cases} 360° - \lambda + \delta_c, & \vec{\boldsymbol{n}}_{\boldsymbol{XZ}} \cdot (\vec{\boldsymbol{e}}_{\boldsymbol{\theta}} \times \vec{\boldsymbol{e}}_{\emptyset})_{XY} < 0 \quad (left) \\ \lambda + \delta_c, & otherwise\ (aligned\ or\ right) \end{cases} \tag{3.8}$$

After performing the previous calculations and reading the data provided by the GPS, two of the three wind triangle vectors are already known. Considering the drift angle (α) as the angular distance between θ_w (ground vector angle) and θ_v, the wind speed is given by the following equation, where r_w represents the ground vector speed:

$$r_u = \sqrt{{r_v}^2 + {r_w}^2 - 2 \cdot r_v \cdot r_w \cdot cos\ \alpha} \tag{3.9}$$

The angle β of the wind triangle, which will be needed to obtain the wind direction, can be determined using the following formula:

$$\beta = cos^{-1}\left(\frac{{r_v}^2 - {r_w}^2 - {r_u}^2}{-2 \cdot r_w \cdot r_u}\right) \tag{3.10}$$

Finally, the wind direction is calculated using the following system of equations, and the expression to be used will depend on the flight direction:

$$\theta_u = \begin{cases} \theta_w + 180° + \beta, & \theta_v \in [\theta_w + 180°, \theta_w] \\ \theta_w + 180° - \beta, & otherwise \end{cases} \tag{3.11}$$

Appendix B shows prints that demonstrate the correct implementation of the algorithm in the microcontroller. For this purpose, a comparison between its results and those obtained in a *Microsoft Excel*® sheet where the same calculation is performed is carried out, for the same input parameters.

2nd method:

The alternative method aims to reduce the uncertainties associated with the determination of the aerodynamic parameters, since it is based on the automatic orientation of the vehicle so that it faces the wind (yaw angle equal to zero). Hence, the only possible

inaccuracies regarding the projected area or the drag coefficient will be related to the system's inclination. However, this procedure goes against the project's idea of estimating the wind locally, so that the UAV can be reoriented autonomously according to the instantaneous variations in the direction and intensity of the wind. Therefore, since this method only allows to obtain one wind direction and intensity per flight cycle, it will only be used to compare results with the first method to test its validity.

Figure 3.5 shows the flowchart for obtaining the wind direction using this method. Firstly, the vehicle is placed in the zero-yaw orientation, then entering a loop that increases this angle up to 180º (for symmetry reasons). In each cycle, the roll angle is read and compared to its minimum value, being the variable that saves the yaw angle also updated when a new minimum roll is found. At the end of the loop, the yaw angle obtained represents the wind direction, which must be related later to the compass angle to get the wind direction.

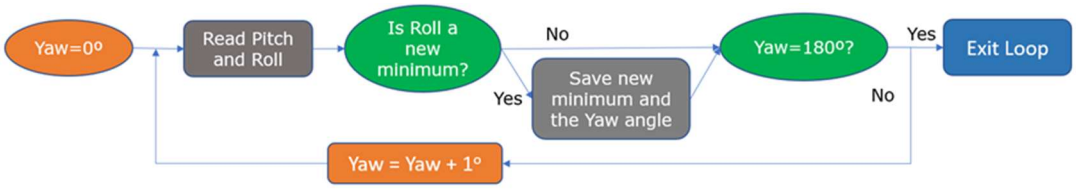


Figure 3.5. Flowchart of the 2nd method cycle.

Regarding the wind intensity, the calculation procedure is similar to that of the first method, but the fact that the aerodynamic parameters are calculated for a given inclination with yaw equal to zero stands, increasing (in theory) the accuracy of the method.

3.4. Aerodynamic Parameters

In this subchapter, the procedures that were performed to obtain the projected area and the drag coefficient will be presented. Fundamentally, it should be noted that, for both variables in research, the best correlation between accuracy and accessibility was sought, since it is impracticable to estimate these parameters for all inclinations and directions (yaw angle) that the UAV can assume.

In fact, the more measurements were taken, the more time would be spent, which might not even lead to practical benefits in terms of increasing the reliability of the calculation. In the particular case of the drag coefficient, it is also important to take into account the fact that it is determined in wind tunnel experiments, which represents an inconvenience to carry out a significant number of measurements. Therefore, for each parameter, increments for the inclination and yaw angles were stipulated, and calibration functions were subsequently formulated.

3.4.1. Projected Area

Since it is necessary to know the area of the UAV facing the wind to determine the drag coefficient, the first parameter to be calculated is the projected area. For this purpose, a CAD model of the vehicle was used and, by projecting its geometry in the *Autodesk Inventor Professional 2021®* [43] software, the area was measured for each increment of the inclination and yaw angles.

At an early stage, in order to check if the direction of the UAV has a significant influence on the evolution of the projected area with the increase of the inclination, an increment of 45º in the yaw angle was used. Since the vehicle is symmetrical about two axes (X and Y), the application of this increment leads to measurements for only three yaw angles (0º, 45º, and 90º were considered).

Regarding the increment for the inclination angle, a value of 2º was adopted, since it is considered that it allows to accurately approximate the evolution of the projected area, without causing long durations in the total measurements. Additionally, the measurements were only made up to a maximum pitch of 45º, as it is considered that this is the highest inclination that the UAV will assume in the idealized application.

To facilitate the computation, some assumptions were made, namely:

- The area for a given inclination angle (combination of roll and pitch angles) of the system is equivalent to that of a similar inclination angle, but considering only the pitch angle different from zero ($\emptyset = 0^{o}$);
- Some components were removed from the assembly, such as the microcontroller and the targeting system, because they have geometries that are extremely difficult to project and

it was considered that their presence would not significantly influence the result obtained for the area;

- In the measurements for 0º and 90º yaw angles, the symmetry of the UAV was explored and only half of its geometry was projected, with the value of the area of this surface being subsequently doubled.

Thus, in the CAD software, planes with the desired slopes were generated and, using the "*Project Geometry*" command, the areas of the projected surfaces were measured. Figure 3.6 shows an example of a projection of the vehicle geometry for each yaw angle, as well as the measurement of its area.

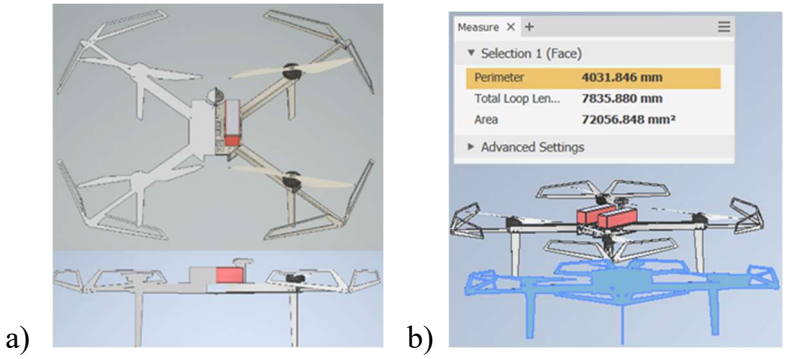


Figure 3.6. Projection of the UAV's area for a yaw angle of: **a)** 0º (inclination of 45 º) and 90 º (inclination of 0 º). **b)** 45 º (inclination of 0 º), with the measurement of the area.

After performing all the proposed measurements, the following graph was plotted, which shows the evolution of the projected area of the UAV with the increase of its inclination, for each examined yaw angle. As expected, these curves are increasing as the vehicle's body emerges as opposed to its frontal face.

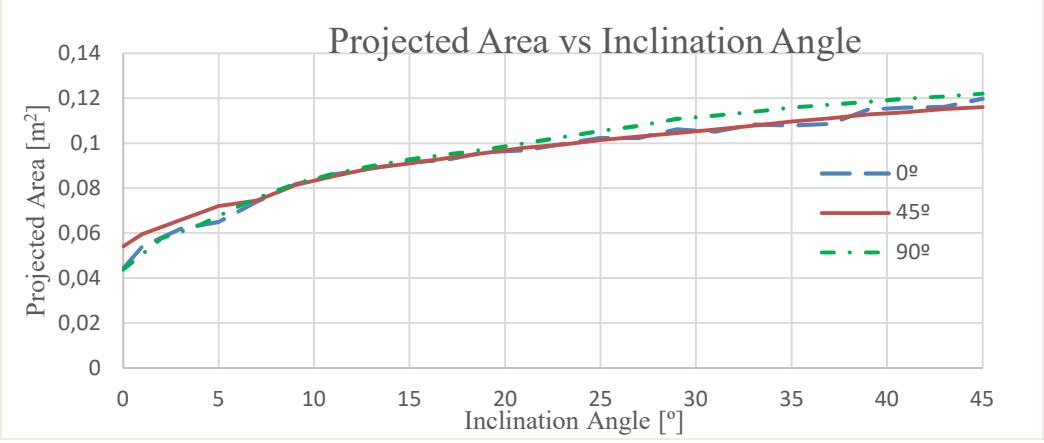


Figure 3.7. Evolution of the projected area of the UAV with the increase of the inclination angle, for three different directions (yaw angle): 0º, 45º and 90º.

The main difference between the three graphs lies in the calculations for the lower inclination angles since, contrary to what happens for yaw angles of 0º and 90º, the propellers and their guards do not obstruct each other two by two for small inclinations when yaw equals 45º, thus increasing the total area. For angles greater than 5°, this effect is no longer noticed, and all curves show a similar growth, so no further measurements were taken.

Finally, for each of the curves, a third-degree polynomial trend line was generated, all with a coefficient of determination (R^2) values greater than 0.99. These will be used in the next subsection to calculate the drag coefficient, and in the algorithm to estimate the wind vector. For this last task, in order to know when to use each of the equations, the relationship between the roll and pitch angles will be used. That is, the equation of zero-yaw will be applied when the roll angle is less than half of the pitch (system aligned with the wind), that of yaw equal to 90º in the inverse situation (system perpendicular to the wind), and that of 45º in any other scenario.

3.4.2. Drag Coefficient

After determining the projected area, the next step is to estimate the UAV's drag coefficient. For this purpose, a procedure similar to the one presented above was used, namely concerning the assumptions and directions (yaw angle) of the tested UAV (0º, 45º, and 90º). However, in this case, no computational analysis tools (such as CFD software) were used, but experiments were carried out in a wind tunnel (at the Industrial Aerodynamics Laboratory – LAI, in Coimbra). Therefore, due to the greater difficulty in carrying out the experiments (not only physical, but also in terms of access and cost), the increment used in vehicle inclination must necessarily be higher than that previously chosen in the area measurements. So, 5º was selected instead of 2º, with tests from zero inclination to a maximum of 40º (imposed by the geometry of the developed structure, so that the UAV would not touch the ground). Also, since the batteries have not yet arrived when the tests were carried out, and considering the influence of their physiognomy on the aerodynamic behavior of the aircraft, styrofoam blocks were used to simulate these components.

To carry out these experiments, a metallic support was developed that allowed the vehicle to be balanced at its center of gravity and placed over a *JR3 PCI*® Force-Torque

sensor [53], used to determine the moments caused by the wind effect on the aircraft. The variation in inclination between tests was provided by a plate created for this purpose, whose slope could be modified using a ball screw, and was measured with the aid of an inclinometer. Additionally, foam pads were used to dampen vibrations in the system. To remove the wind effect on these structures from the desired calculation, they were also tested alone in the wind tunnel, with the results of these experiments being later subtracted from those of the complete system, thus obtaining the wind force only in the UAV.

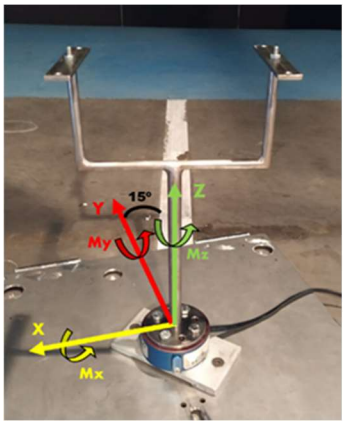
Since the goal of the experiments was only to measure the effect of the wind on the UAV, it became necessary to reset the load cell between each measurement, so that it would not consider the variations in the effect of the vehicle's weight on each axis of the sensor, with the increase in inclination. However, the reset had to be carried out in circumstances of total absence of wind, so that this phenomenon could be properly estimated. Although the most straightforward solution to this problem is, of course, to turn off the tunnel between each measurement, this procedure entails very high energy costs, given the huge number of tests necessary to reproduce. Thus, it was decided to build a wooden box that could enclose the entire structure, so that the sensor software could be reset.

The methodology followed in each experiment was as follows:

1. Turn on the tunnel, with a wind speed of 5m/s (the maximum value that did not induce visible vibrations in the system). The thickness of the boundary layer – 0,1 m [54] – is much smaller than the ground distance from the UAV's center of gravity, so it was neglected;
2. Set the vehicle’s inclination;
3. Isolate the structure from the wind, placing the box around it;
4. Reset the Force-Torque sensor;
5. Remove the box, and read the values indicated in the cell software (since it does not allow recording data for further processing, ten screenshots were taken in each test, and their average was calculated).

Since there is an arm between the wind force and the sensor, the type of load to be analyzed is the moment measured by the cell, which will later be transformed into the intended force. However, after the construction of the structure that supported the UAV, a slight inaccuracy was discovered in it regarding the alignment of the cell with the vehicle. As can be seen in Figure 3.8.a), in a situation where the latter is perfectly aligned with the

wind, the cell presents a deviation of 15º (a value that remains for the three sets of measurements), due to an erroneous generation of the threaded holes that allow the vehicle support to be fixed to the tilting plate. Therefore, the calculation method is slightly more complex than it would be if the cell were aligned with the vehicle (a representation of this is shown in Figure 3.8.b).

a) 

b) 

Figure 3.8. a) Deviation of the tilting plate in relation to the wind. **b)** Force diagram in the vehicle.

Assuming that, due to its symmetry along the XZ and YZ planes, the vehicle's center of gravity (the point where the wind force is exerted) is at its central point for all yaw-inclination conjugations and that the moment around the Z-axis is always negligible (values always vary in sign, for the same test), it can be considered that only moments around the X and Y axes will act in the system. Thus, by obtaining these values from the cell software, it is possible to determine the two components of the wind force perpendicular to these two axes ($F_{y_{cell}}$ and $F_{x_{cell}}$) by dividing them by the respective arms. Then, by trigonometry (considering the 15º offset of the cell in relation to the wind), it is necessary to estimate the components of the latter according to the wind direction (F_y and F_x, respectively), and the sum of these corresponds to the force applied by the wind on the structure (F_{wind} in Figure 3.9).

Finally, all that remains is to assess which arms to consider when calculating the forces from the moments obtained by the sensor. Based on Figure 3.9, which shows the pose of the system for a given inclination of the structure, it is possible to understand how these distances evolve as this inclination increases. Focusing on the load cell's X and Y axes, it is easy to see that, unlike the latter, whose angle relative to the tunnel floor varies accordingly

to the inclination of the structure, the X-axis is always parallel to the ground. Hence, the forces perpendicular to these axes and responsible for the moments estimated by the sensor ($F_{x_{cell}}$ and $F_{y_{cell}}$) are at different distances from the center of gravity of the body. While the force perpendicular to the Y axis ($F_{x_{cell}}$) is always at a distance from it equal to the support arm ($a = 0.248\ m$), the arm corresponding to $F_{y_{cell}}$ decreases with inclination, and is given by the equation $b = a * \cos(\sigma)$, where α corresponds to the inclination of the structure.

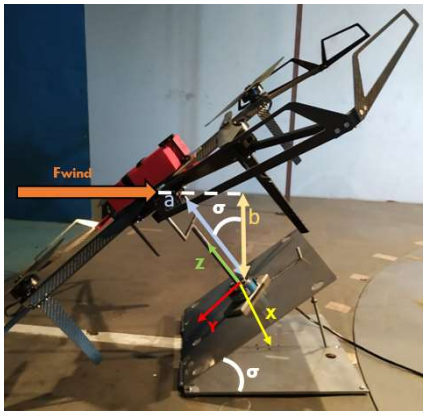


Figure 3.9. Representation of the arms of the forces acting on the vehicle, when tilted.

At this stage, the description of the elaboration of the experiments is concluded. In the fourth chapter, the results of this methodology will be presented and analyzed, and the validation of the model will be discussed.

3.5. *MAVLink®*

Since the creation of unmanned vehicles and systems, there is a need to improve communication between these and the ground control station. Globally, the three best-known messaging protocols for establishing these connections are *UAVCan®* [55], *UranusLink®* and *MAVLink®* [7]. Although the first two are lightweight and open-source tools, they are not as scalable and reliable as the third, which makes the latter the most used and accepted protocol when it comes to unmanned systems.

The first version of *MAVLink®* [56] was published by Lorenz Meier in 2009, and since then it has covered a wide range of programming languages and autopilots, and facilitated communication not only between unmanned systems and GCS but also between different machines and even with the various individual components of them. This protocol

operates a serialization (conversion to a sequence of bytes) of the messages to a binary format that mobilizes a very small overhead size, so the overall message size will be lighter than that of other protocols. Transmission can be carried out via serial telemetry, *Ethernet*, or *Wi-Fi*, and its reliability is guaranteed by automatically detecting when packet losses occur.

Figure 3.10 represents the message header of the latest version [57] (2.0 – the one that was used in this project) of *MAVLink®,* which was published in 2017 with the main purpose of increasing the variety of messages that can be sent, as *MAVLink®* 1.0 only allowed the definition of 255 (8 bits) distinct, as well as their reliability and security. Each message is divided into 10 or 11 fields (depending on the respective values) that make it possible to identify and recognize the message, as well as the devices that send and receive it.

Figure 3.10. *MAVLink®* 2.0 packet header.

The first one ("*Start-Of-Text*") concerns the beginning of a new message and prepares the receiver to analyze the remaining bytes. As for the second and fifth bytes, the payload length and the message sequence number, respectively, are indicated, the latter being extremely necessary to detect loss of information. Among these, it is important to highlight the existence of two flags, exclusive to *MAVLink®* 2.0, which contain information that may or may not influence the structure of the package, depending on whether it appears in the third ("*Incompatibility Flags*") or in the fourth ("*Compatibility Flags*") byte.

The next two fields consist of the IDs of the system (vehicle or GCS) and its component (IMU, camera, GPS, etc) that sends the message, while the eighth field involves the identification of the type (or ID) of the specific message that is being sent, such as vehicle speed or orientation. One of the upgrades from the first to the second version of the protocol lies on increasing the size of the latter field from 1 to 3 bytes, thus enabling the definition of 16777215 different messages (even customizable ones), instead of the 255 possible in *MAVLink®* 1.0, each one being generated in a separate header file [58].

After analyzing all the above fields, the receiver can read the payload to obtain all the information contained in the message. This is the part of the packet that has the longest length and can even reach 255 bytes of information. The "*Checksum*" field indicates the end of the header and guarantees that the message has not changed during its transmission.

Finally, it remains to address the last field shown in Figure 3.10 ("*Signature*"), which is included in the package when the "*Incompatibility Flag*" is set to 0x01 and which aims to increase the security of the message. This 13-byte field was added to the header when *MAVLink*® 2.0 was launched, and incorporates information about the link/channel where the packet was sent, a timestamp that allows the message to be discarded if it is older than the one previously received or if it has taken too long to arrive, and a signature related to a secret key that makes it possible to verify whether the received message matches the one sent or not.

In this project, the messages exchanged between the vehicle and the GCS will be divided into two distinct groups, corresponding to the two directions in which the communication can be established. The first group corresponds to the messages that are intended to be sent by the UAV to the GCS, namely the "*Heartbeat*" [59] (to ensure that the system is alive and receiving requests correctly) and the parameters representing the current status of the vehicle (such as the attitude [60] and the velocity components [61][62]). On the other hand, it is important to send some commands to the UAV, namely related to the control of its translation and rotation movements, and the speed with which it executes them [63].
Another note worth mentioning is the use of the *QGroundControl*® [64] software as a GCS, as it is an open-source environment that allows full configuration for the *PX4*® [65] autopilot and is very user-friendly. The employment of this autopilot was mainly due to the use of the *PX4Flow*® optical flow sensor which, despite working correctly with other autopilots (such as the *Ardupilot*® [66]) was initially conceived to be integrated with the *PX4*®, which resulted in better data transmission between this sensor and the flight controller.

3.6. Ground Control Station

The last step in the work formulation consists in the development of a GUI, whose purpose is to provide the user with a direct and intelligible communication with the UAV, so that the user can control the vehicle's operation as easy as possible. Therefore, and taking into account that the microcontroller is connected by wires to the other components of the UAV, the need to establish a communication between the ground station and the microcontroller emerges. Since the microcontroller selected for the project has a built-in *Wi-*

Fi module, a communication via *Wi-Fi* was implemented, considering that it provides faster and more secure data flows than those provided by a *Bluetooth* connection, as well as a greater connection range.

3.6.1. Communication between Ground Station and Microcontroller

For an *ESP32*® microcontroller, there are three different ways to set up a wireless communication with a client: Station Mode, Access Mode, and a combination of both. The contrast between the first two is explained in Figure 3.11, where it is shown that the Station Mode presupposes the connection of both the microcontroller and the client to an external *Wi-Fi* network (for example one created by a router), while the Access Mode allows the generation of a microcontroller's own network, enabling a direct connection with users. Since this project is intended for forest applications, the latter was used.

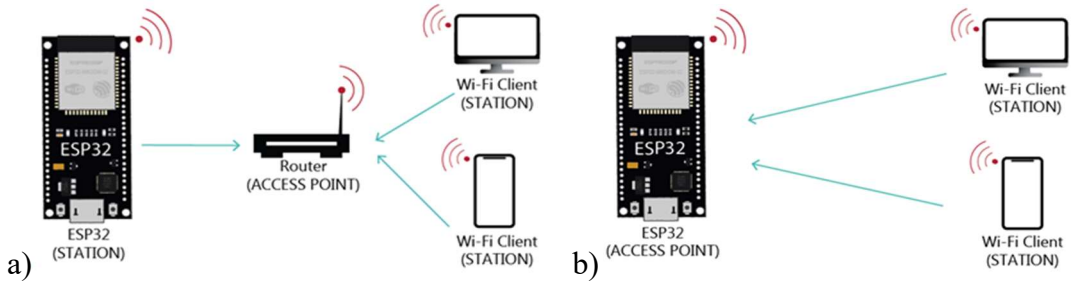


Figure 3.11. a) Station Mode. **b)** Access Mode. Adapted from [67].

In order to make the communication between the GUI and the microcontroller possible, sockets were used, which correspond to data structures that allow understanding how the connection is made (the communication protocol employed), which machines exist on both sides (their IP's), and the port numbers of the applications, the amount of memory available to communicate, which data will flow and in which format (binary, text, etc.) and whether they will be streamed or not, etc. The data transmission consists of sending a specific character by the GCS as a request (*ASCII* encoding), which allows a specific response from the microcontroller to be received. For example, the continuous sending of the "!" character causes the microcontroller to respond by sending the flight angles, the UAV's velocity, ground distance and orientation, and the wind speed and direction, with all parameters being separated by a space character (" ").

Regarding the communication protocol, two different formats (TCP and UDP) were implemented in order to test which one is the most suitable at a general level and for each type of data. The Transmission Control Protocol (TCP) is a connection-oriented transport layer that is ideal for sending and receiving messages and commands that are more critical for the correct performance of the task (such as the start and end events), since it does not cause loss of information and guarantees the data flow in the correct order (streaming). However, these aspects, combined with the error detection (recovery of lost data) and the bidirectional communication of this protocol, lead to a lower flow rate, which represents its main disadvantage compared to other transport layers.

On the other hand, the User Datagram Protocol (UDP) appears as the opposite of the previous protocol, since there is a high speed in the data flow due to the fact that there is no guarantee of the reception of messages by the server, as well as the respective order. Furthermore, there is no need to pre-establish a connection between client and server, since each data packet is independently sent to the endpoint of the transmission (datagram). Thus, this protocol is used in applications where the flow speed is more important than the loss of some information, such as, for example, the image obtained by cameras, where the highest possible transmission speed is desired, even if this results in the loss of a pixel or a frame.

Both protocols work based on the Internet Protocol (IP), as the latter is responsible for addressing the messages to the destination, using its IP address. TCP and UDP can be used simultaneously in applications with different data to be transferred and, consequently, with different objectives in each data flow, in order to increase the communication efficiency.

3.6.2. Graphical User Interface

Figure 3.12 shows the GUI developed for controlling and reading the parameters of the UAV. It has several features that, in the figure, are organized in blocks that serve a particular purpose. The first one represents the establishment of the communication between the GCS and the vehicle, through the IP and port of the microcontroller and a *Button* that allows switching between UDP (default) and TCP protocols. Other controls present in this block are a *CheckBox* that, when activated, allows the start of the data stream, a *ComboBox* that allows changing the data flow rate and a *Button* to make it possible to record the stream in a text file.

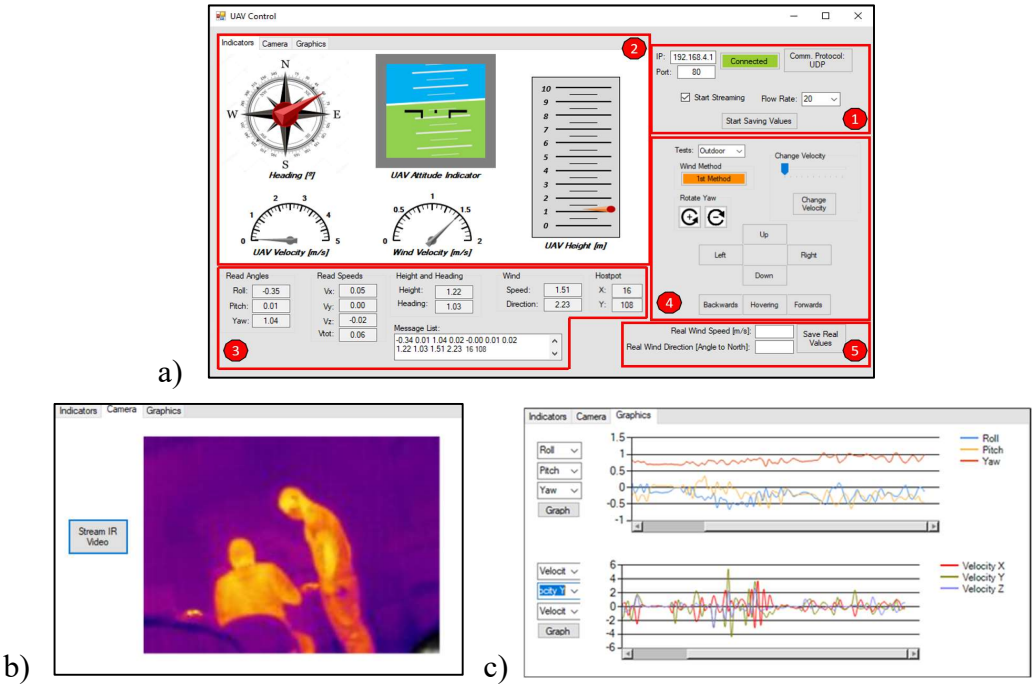


Figure 3.12. a) Graphical User Interface, with the Indicators tab. **b)** Camera tab. **c)** Graphics tab. Blocks: 1 - Initialization of communication between UAV and GCS; 2 – *TabPages* for viewing parameter variation; 3 – Data exhibition; 4 – Vehicle's control; 5 – Wind inputs.

Block number two consists of a set of *TabPages*, each displaying the collected data differently. The first tab (Figure 3.12.a)) displays some common aviation indicators, such as velocities, orientation and vehicle-to-ground distance, while the second and third (Figure 3.12.b) and Figure 3.12.c)) show the image obtained by the IR sensor and graphs of the evolution of the various parameters over time, respectively. The values used in this block will be the ones exhibited in the *TextBoxes* of the third one. In this last group, there is also a list of messages sent by the microcontroller to the GCS, where both the data obtained and the status of the process can be expressed (through messages like "*Taking off*").

Finally, the last two blocks concern inputs in the system, either in the UAV or in the interface itself. The fourth represents an alternative to the vehicle's radio control, through *Buttons* that induce translation along the three axes, rotation around the yaw angle (direction of movement) and the variation of the UAV's velocity. The "*Hovering*" *Button* allows the vehicle to stabilize at a certain point in space, a phenomenon that often requires adapting the aircraft attitude according to the current situation. On the other hand, in the last block there are two *TextBoxes* where the real values of wind speed and orientation (obtained, for example, by meteorological stations) can be stored, to allow the comparison with the values calculated by the algorithm in the graphs shown in Figure 3.12.c).

4. EXPERIMENTAL TESTS

After the presentation of the methodology followed during the wind tunnel tests to calculate the drag coefficient of the UAV, the respective results will be exposed and discussed in this chapter. Since, due to shipping delays associated with the *COVID-19* pandemic, the batteries ordered for the vehicle only arrived about a week before the deadline for submission of this dissertation, it was not possible to confirm the full validation of the project before it, namely the tasks that required the UAV's flight, such as its control through the interaction with the GUI and local estimation of the wind. However, in order to demonstrate the correct functioning of the remaining blocks of the GUI and the connections between the various components of the system, an experiment carried out using these requirements will be briefly referenced.

4.1. Drag Coefficient Experiments

Regarding the wind tunnel experiments for estimating the drag coefficient, it is important to define the mathematical procedure for calculating the wind force and, subsequently, the UAV drag coefficient, from the M_x and M_y readings by the sensor, for the structure with and without the vehicle. First, the dimensionless results were converted into the SI units of moment (N.m) through a cell calibration curve, and then divided by the respective arm, thus resulting in the $F_{x_{cell}}$ and $F_{y_{cell}}$ forces. Additionally, the 15º deviation of the Y-axis of the cell in relation to the wind was taken into account, and F_x and F_y were calculated, which, added together, result in the total wind force on the structure. To determine how much of this force is reflected only in the aircraft, the force felt only in the base-support set was subtracted from this total. As expected, Figure 4.1 shows that this force continually increases as the vehicle's inclination also increases, regardless of the UAV's orientation. This variation between the orders of magnitude of 1.10 N (zero inclination) and 1.50 N (40º) is mainly due to the increase in the projected area that offers resistance to the wind.

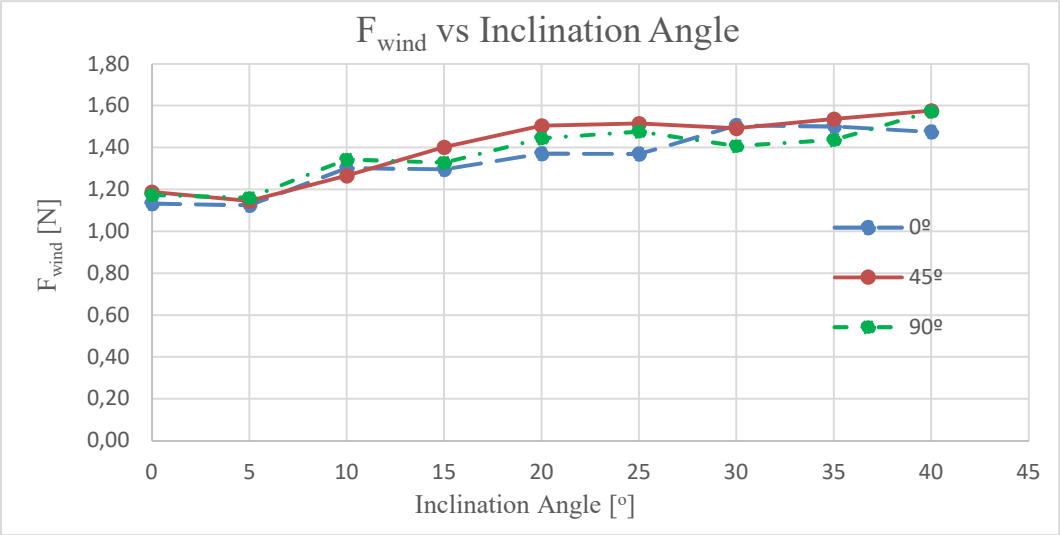


Figure 4.1. Evolution of the drag force on the UAV with the increase in inclination, for different directions.

The next step consists in taking the results of this force and the projected area already determined in the previous chapter and estimating the curve of the evolution of the drag coefficient with the increase in inclination. To this end, for the slopes tested in the wind tunnel, the drag coefficient equation, Eq.(3.1), was applied, where A corresponds to the projected area, ρ to the air density, and u to the wind speed (5 m/s). In this equation, a value for ρ of 1,222 kg/m^3 was assumed, since the relative error between this and the real ones for altitudes between 0 m and 50 m (maximum height of the UAV) is extremely small (as demonstrated in Gonçalo Rodrigues' dissertation).

Based on that formula, the following graph was plotted, which expresses a decrease in the UAV drag coefficient with an increase in its inclination. Contrary to what would be expected, as this type of vehicle normally offers less aerodynamic resistance in the situation of zero inclination, the UAV designed in this project appears to behave differently, with drag coefficient values between 1.39 and 1.62 for this last scenario and only about 0.9 at the maximum slope tested.

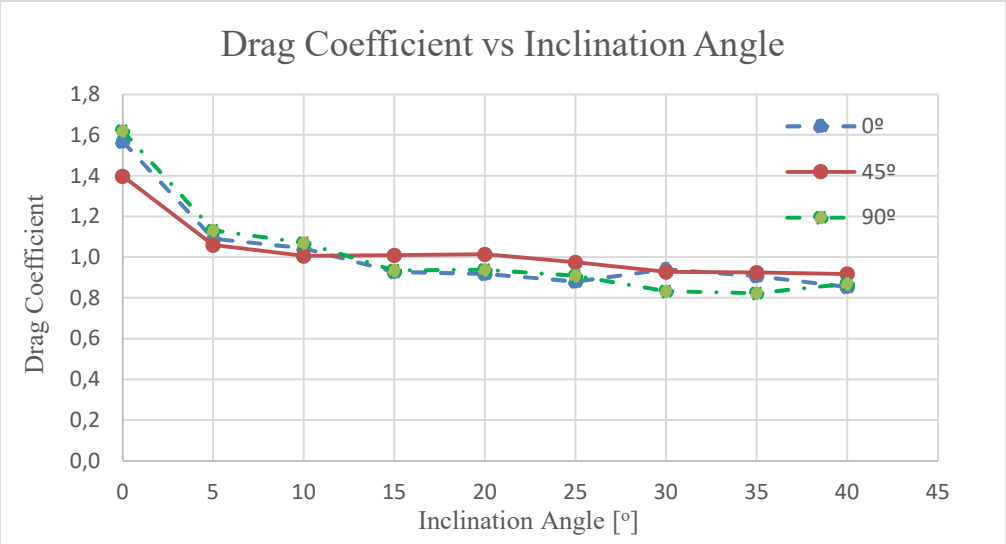


Figure 4.2. Evolution of the drag coefficient with the increase in inclination, for different directions.

Beyond the fact that the effect of propeller rotation was not addressed in this calculation and that only ten values were read per test (which does not allow the total elimination of noise and the most correct determination of the real value), this phenomenon can be explained by the introduction of some components that make the body shape more streamlined as it is more inclined, such as prop guards. These elements contribute to a significant increase in the viscous component of drag due to the effect of shear stresses acting on the vehicle surface, for low slopes. Additionally, the fact that the UAV's arms are extremely thin contributes to the separation of the flowing air being especially reduced as its inclination increases, thus reducing the pressure drag component.

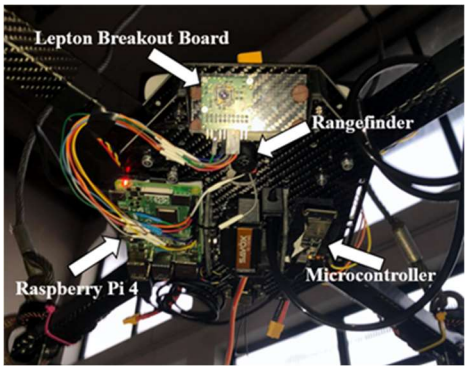
To conclude, and similar to the procedure followed for the projected area, a cubic polynomial trend line (with a coefficient of determination – R^2 – greater than 0.9) was generated for each of the curves, so that they could be included in the algorithm running in the microcontroller to determine the speed and direction of the wind. It is important to highlight that, ideally, additional tests were planned which, with the vehicle in flight and knowing the direction and intensity of the incident wind, consisted of measuring these parameters through the designed algorithm, to subsequently apply bias and fine-tuning techniques to increase its accuracy. However, the late arrival of the batteries made it impossible to carry out these measurements in time for the delivery of this dissertation, something that would significantly reduce the errors and uncertainties associated with not measuring the effects of rotor rotation for different rotation speeds in the drag coefficient tests.

4.2. Connection experience within the system

Despite the impossibility of flying the vehicle in time for the submission of this dissertation, the reliable connection between its various components was verified. As seen in Figure 3.12.a), which exposes the GUI designed to control the UAV, any movement of the flight controller (whether rotation or translation) can be demonstrated in it, through *TextBoxes* and indicators. This characteristic makes it possible to conclude that not only the *Wi-Fi* communication between the microcontroller and the GCS, but also all the connections between the various constituents of the system payload, are correctly established.

In order to test this communication in an actual application, an experiment was carried out whose main objective was to measure the distances from the vehicle to the hotspot of the thermal image obtained by the IR sensor. For this purpose, a hoist located at LAI, Coimbra, was used to hold the UAV at certain distances from the ground and from a powered heat gun. While the results of this experiment will be studied in Gonçalo Rodrigues' dissertation, this reference seeks to highlight, in addition to the wireless connection between the vehicle and the GUI, the correct implementation of the communication between the IR sensor breakout board with the *Raspberry Pi*®, and between the latter and the microcontroller (Figure 4.3.a)).

In fact, as shown in Figure 4.3.b), it is possible to obtain the video stream from the IR sensor, as well as the hot spot coordinates determined on the *Raspberry Pi*®, which demonstrates that the system is responding accurately to this task. The functionality of generating a text file with a list of values obtained between consecutive clicks of the "*Start Saving Values*" button was also explored, which allowed a more precise treatment of the data by subsequently eliminating some noise in the measurements.

a) 

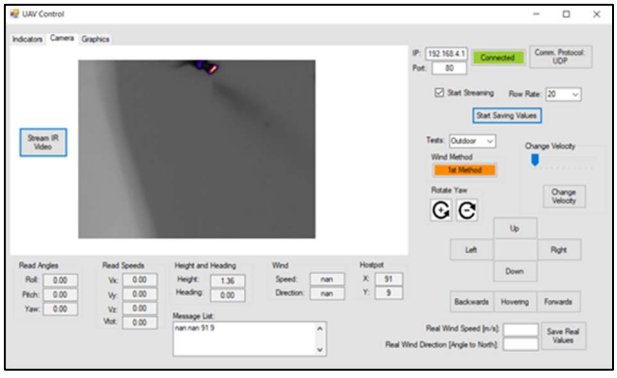
b) 

Figure 4.3. a) Hotspot Experiment Setup; **b)** Video stream and hotspot coordinates in real-time, in GUI.

5. CONCLUSIONS

This project, as a whole, was conceived to explore a weakness in an industry that, despite being urgent, still does not have the advances that current scientific knowledge would allow. The possibility, in addition to monitoring a fire in real-time, of the UAVs being the first agents of fire extinction emerges as a current and future need to protect the planet and the lives that inhabit it. For this, it is necessary to design a system that is capable of detecting a hotspot and throwing a water jet in its direction, considering the local wind effect, with this last task being the main goal of this dissertation.

The work began with the selection of an algorithm for determining the wind with a UAV within the scientific literature, and its subsequent adaptation to the outlined project. The algorithm developed by Neumann and Bartholmai [24] was fundamentally chosen based on a relationship between its efficiency and the amount of payload added to the system. The proposed methodology assumes only the need to read the flight angles and GPS data, and to measure the projected area and the drag coefficient (using CAD software and experiments in wind tunnels, respectively) of the vehicle for different inclinations. Taking into account the impracticality associated with carrying out measurements for all infinitesimal variations in the vehicle's direction and inclination, polynomial calibration functions were formulated.

While the evolution obtained for the projected area as a function of the slope can be considered with high levels of precision, the same may not have happened for the drag coefficient curve, largely due to the fact that the readings were taken through ten sets of values per test (the Force-Torque sensor’s software used does not provide the ability to record data over a period of time) and, essentially, the influence of rotor rotation for different speeds has not been considered. Ideally, this imprecision would be faded through fine-tuning techniques, when estimating the local wind with the vehicle in flight, and comparing it with the real value obtained by an anemometer, for further improvement of the model. However, the major obstacle to the project's realization arose, related to a delay in the delivery of the batteries ordered for the UAV, which made it unfeasible for it to be tested in flight. In fact, this impediment inhibited the possibility of testing the entire

system in time for the submission of this dissertation, both in terms of determining the wind and controlling the UAV to direct the water jet to the hotspot.

Nevertheless, it was possible to confirm the reliability of the remaining work carried out, namely that related to communications between components and between the vehicle and the GCS (via *Wi-Fi*). Naturally, the main focus was on understanding the message protocol to be exchanged with the flight controller (*MAVLink*®), in order to receive the necessary data to proceed with the wind estimation algorithm, but also to be able to control the vehicle's movement. On the more negative side, it is only important to highlight the need to change the system payload initially foreseen, due to the fact that it was not possible to establish a reliable communication between the IR sensor breakout board (*FLIR Lepton 3.5*®) and the microcontroller (*Adafruit ESP32 Feather*®), so a *Raspberry Pi 4*® was eventually added to the setup.

In addition, a GUI was designed in order to control the UAV and view some common aviation indicators, so that the user can have a more intelligible perception of its flight. Parameters such as vehicle inclination, orientation, and speed are displayed on their own indicators, being even possible to plot graphs that show their variation over time, and the video stream obtained by the sensor can be exhibited on its own tab.

Regarding the future work in this field and for this particular project, in addition to carrying out the tests to assess the reproducibility of the model, some procedures can be added to this method to improve its efficiency. Given the immensity of utilities that the *MAVLink*® protocol provides to the user, the local wind calculation can be the target of calibration techniques using, for example, readings of the thrust values sent to the motors. By knowing the speed at which they rotate, it becomes easier not only to estimate the vehicle's drag coefficient, but also the intensity and direction of the wind itself. On the other hand, and naturally, an increase in the number of measurements of the projected area and the drag coefficient, for different directions and orientations, will contribute to greater precision of the generated trendlines, reducing inaccuracies in the final calculations. If necessary, to compare the results obtained in the wind tunnel experiments, CFD simulation can be employed and a procedure similar to the one for the projected area can be performed.

It is also possible to deepen the variety of functionalities provided by the GCS software, namely the possibility of planning missions, and incorporating them into the

implemented algorithm. For example, it would be interesting to plan routes for the aircraft to fly and, if the IR sensor detects temperature values above a certain value, stop that route and proceed autonomously to its extinction.

In short, even though the system has not been fully tested, it is believed to have reached an interesting solution to the identified problem. In fact, considering that what remained to be tested in this dissertation (wind estimation algorithm) was adapted from another proven scientific work, and that the remaining tasks were successfully completed, it is plausible to assume that the system will produce adequate results. Thus, a future resizing of the prototype to a higher scale can be considered, in order to bring a system of this type closer to a solution for direct firefighting.

BIBLIOGRAPHY

[1] "A Brief History of Firefighting." https://www.popularmechanics.com/technology/gear/reviews/g1442/a-brief-history-of-firefighting/?slide=7 (accessed Dec. 06, 2020).

[2] "Thermite Robot." https://www.howeandhowe.com/civil/thermite (accessed Oct. 06, 2020).

[3] "TAF 20." https://www.crisis-response.com/comment/blogpost.php?post=221 (accessed Oct. 06, 2020).

[4] G. M. Garfin *et al.*, "Chapter 25 : Southwest. Impacts, Risks, and Adaptation in the United States: The Fourth National Climate Assessment, Volume II," Washington, DC, 2018. doi: 10.7930/NCA4.2018.CH25.

[5] "Allied Market Research." https://www.alliedmarketresearch.com/firefighting-drone-market-A06280

[6] "Arduino IDE." [Online]. Available: https://www.arduino.cc/en/software

[7] "MAVLink." https://mavlink.io/en/

[8] "Microsoft Visual Studio." [Online]. Available: https://visualstudio.microsoft.com/

[9] J. J. Roldán-Gómez, E. González-Gironda, and A. Barrientos, "A survey on robotic technologies for forest firefighting: Applying drone swarms to improve firefighters' efficiency and safety," *Applied Sciences (Switzerland)*, vol. 11, no. 1, pp. 1–18, Jan. 2021, doi: 10.3390/app11010363.

[10] H. González-Jorge, J. Martínez-Sánchez, M. Bueno, and P. Arias, "Unmanned aerial systems for civil applications: A review," *Drones*, vol. 1, no. 1. MDPI AG, pp. 1–19, Dec. 01, 2017. doi: 10.3390/drones1010002.

[11] "DJI Matrice 210 V2." https://www.dslrpros.com/dji-matrice-210.html (accessed Dec. 18, 2020).

[12] "Aerones Firefighting UAV." https://www.redbull.com/ng-en/aerones-fire-drone (accessed Dec. 18, 2020).

[13] "BEHA M1-AT Air Tanker." https://www.wearefinn.com/topics/posts/faradair-announces-firefighting-air-tanker-variant-of-beha-m1/ (accessed Feb. 24, 2020).

[14] R. Rysdyk, "UAV PATH FOLLOWING FOR CONSTANT LINE-OF-SIGHT."

[15] A. Mokhtari and A. Benallegue, "Dynamic Feedback Controller of Euler Angles and Wind parameters estimation for a Quadrotor Unmanned Aerial Vehicle."

[16] IEEE Robotics and Automation Society., *2005 IEEE International Conference on Robotics and Automation (ICRA) : Barcelona, Spain : 18-22 April, 2005.* IEEE, 2005.

[17] H. J. Palanthandalam-Madapusi, A. Girard, and D. S. Bernstein, "Wind-field reconstruction using flight data," in *Proceedings of the American Control Conference*, 2008, pp. 1863–1868. doi: 10.1109/ACC.2008.4586763.

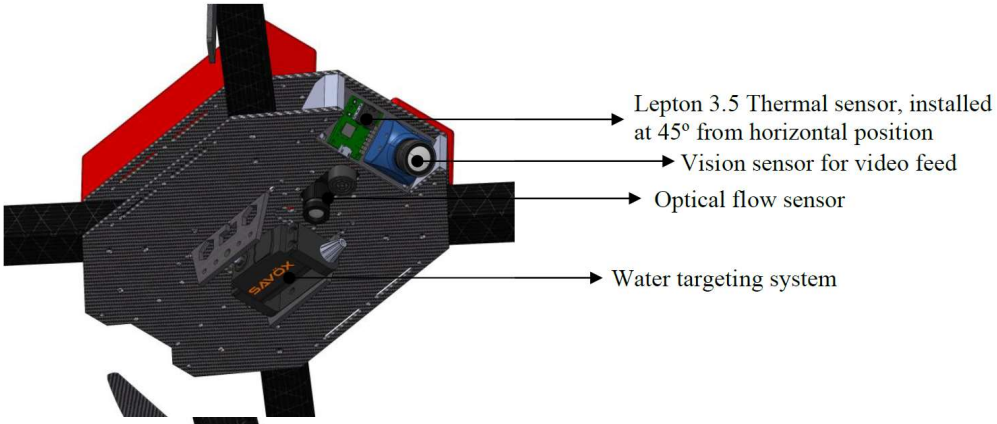
[18] A. van den Kroonenberg, T. Martin, M. Buschmann, J. Bange, and P. Vörsmann, "Measuring the wind vector using the autonomous mini aerial vehicle M2 AV," *Journal of Atmospheric and Oceanic Technology*, vol. 25, no. 11, pp. 1969–1982, 2008, doi: 10.1175/2008JTECHA1114.1.

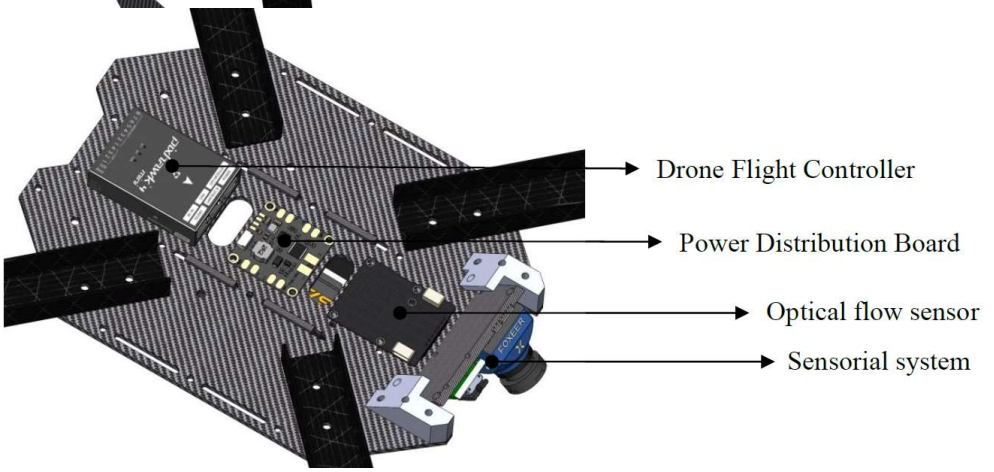
[19] B. Scholarsarchive, A. Felipe, and R. Perez, “Real-Time Wind Estimation and Video Compression Onboard Miniature Aerial Vehicles BYU ScholarsArchive Citation,” 2009.
[20] W. L. Chan, C. S. Lee, and F. B. Hsiao, “Real-time approaches to the estimation of local wind velocity for a fixed-wing unmanned air vehicle,” *Measurement Science and Technology*, vol. 22, no. 10, 2011, doi: 10.1088/0957-0233/22/10/105203.
[21] S. Mayer, G. Hattenberger, P. Brisset, M. O. Jonassen, and J. Reuder, “A ‘no-flow-sensor’ Wind Estimation Algorithm for Unmanned Aerial Systems,” 2012. [Online]. Available: http://paparazzi.enac.fr/wiki/Overview
[22] P. Neumann, M. Bartholmai, J. H. Schiller, B. Wiggerich, and M. Manolov, “Micro-drone for the characterization and self-optimizing search of hazardous gaseous substance sources: A new approach to determine wind speed and direction,” in *ROSE 2010 - 2010 IEEE International Workshop on Robotic and Sensors Environments, Proceedings*, 2010, pp. 1–6. doi: 10.1109/ROSE.2010.5675265.
[23] P. P. Neumann, S. Asadi, A. J. Lilienthal, M. Bartholmai, and J. H. Schiller, “Autonomous gas-sensitive microdrone: Wind vector estimation and gas distribution mapping,” *IEEE Robotics and Automation Magazine*, vol. 19, no. 1, pp. 50–61, Mar. 2012, doi: 10.1109/MRA.2012.2184671.
[24] P. P. Neumann and M. Bartholmai, “Real-time wind estimation on a micro unmanned aerial vehicle using its inertial measurement unit,” *Sensors and Actuators, A: Physical*, vol. 235, pp. 300–310, Nov. 2015, doi: 10.1016/j.sna.2015.09.036.
[25] R. T. Palomaki, N. T. Rose, M. van den Bossche, T. J. Sherman, and S. F. J. de Wekker, “Wind estimation in the lower atmosphere using multirotor aircraft,” *Journal of Atmospheric and Oceanic Technology*, vol. 34, no. 5, pp. 1183–1191, May 2017, doi: 10.1175/JTECH-D-16-0177.1.
[26] M. Marino, A. Fisher, R. Clothier, S. Watkins, S. Prudden, and C. S. Leung, “An Evaluation of Multi-Rotor Unmanned Aircraft as Flying Wind Sensors,” 2015.
[27] J. Y. Wang, B. Luo, M. Zeng, and Q. H. Meng, “A wind estimation method with an unmanned rotorcraft for environmental monitoring tasks,” *Sensors (Switzerland)*, vol. 18, no. 12, Dec. 2018, doi: 10.3390/s18124504.
[28] N. Vasiljević *et al.*, “Wind sensing with drone-mounted wind lidars: Proof of concept,” *Atmospheric Measurement Techniques*, vol. 13, no. 2, pp. 521–536, Feb. 2020, doi: 10.5194/amt-13-521-2020.
[29] K. A. Adkins, C. J. Swinford, P. D. Wambolt, and G. Bease, “Development of a sensor suite for atmospheric boundary layer measurement with a small multirotor unmanned aerial system,” *International Journal of Aviation, Aeronautics, and Aerospace*, vol. 7, no. 1, pp. 1–14, 2020, doi: 10.15394/IJAAA.2020.1433.
[30] W. Thielicke, W. Hübert, and U. Müller, “Towards accurate and practical drone-based wind measurements with an ultrasonic anemometer,” *Atmospheric Measurement Techniques Discussions*, pp. 1–29, 2020, doi: 10.5194/amt-2020-258.
[31] A. Woo, P. Poulin, and A. Fournier, “A Survey of Shadow Algorithms,” *IEEE Computer Graphics and Applications*, vol. 10, no. 6, pp. 13–32, 1990, doi: 10.1109/38.62693.

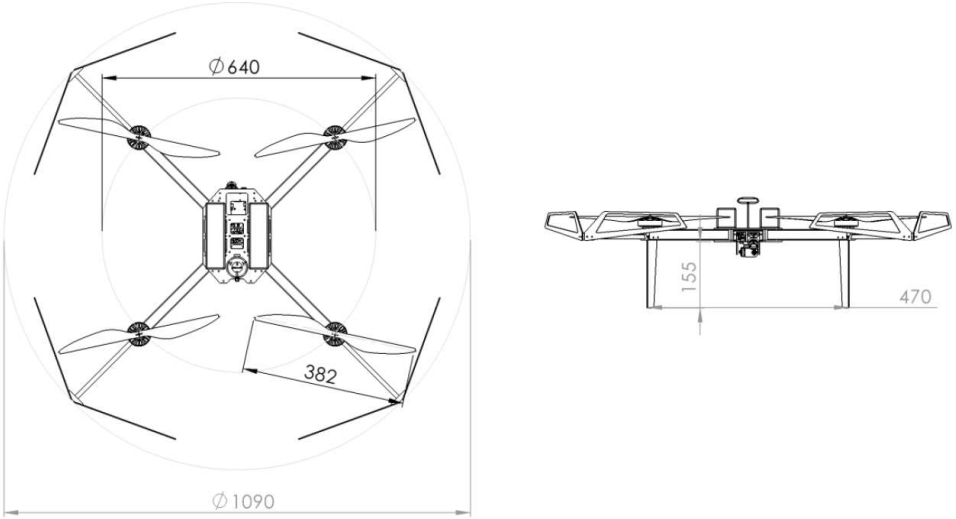
[32] D. J. Brown and G. T. Vickers, “use of projected area distribution functions in particle shape measurement,” 1998.
[33] P. Sagnes, “Using multiple scales to estimate the projected frontal surface area of complex three-dimensional shapes such as flexible freshwater macrophytes at different flow conditions,” *Limnology and Oceanography: Methods*, vol. 8, no. SEPT, pp. 474–483, Sep. 2010, doi: 10.4319/lom.2010.8.474.
[34] O. Ben-Yaacov, E. Edlerman, and P. Gurfil, “Analytical technique for satellite projected cross-sectional area calculation,” *Advances in Space Research*, vol. 56, no. 2, pp. 205–217, Jul. 2015, doi: 10.1016/j.asr.2015.04.004.
[35] T. Volz, R. Schwaiger, J. Wang, and S. M. Weygand, “Comparison of three approaches to determine the projected area in contact from finite element Berkovich nanoindentation simulations in tungsten,” in *IOP Conference Series: Materials Science and Engineering*, Nov. 2017, vol. 257, no. 1. doi: 10.1088/1757-899X/257/1/012013.
[36] E. Kljuno and A. Catovic, “Estimation of projected surface area of irregularly shaped fragments,” *Defence Technology*, vol. 15, no. 2, pp. 198–209, Apr. 2019, doi: 10.1016/j.dt.2018.08.012.
[37] H. A. Israr and M. N. Dahalan, “Estimation of Lift and Drag Characteristics of UTM Elang-1 UAV Low Velocity Crushing of CFRP Laminates View project Hybrid composite of E-glass/flax on tension, flexural and water absorption properties View project,” 2008. [Online]. Available: https://www.researchgate.net/publication/220024745
[38] F. ; Schiano *et al.*, “Towards Estimation and Correction of Wind Effects on a Quadrotor UAV,” 2014, doi: 10.3929/ethz-a-010286793.
[39] R. Felismina, M. Silva, A. Mateus, and C. Malça, “Study on the aerodynamic behavior of a UAV with an applied seeder for agricultural practices,” in *AIP Conference Proceedings*, Jun. 2017, vol. 1836. doi: 10.1063/1.4981989.
[40] J. Svacha, K. Mohta, and V. Kumar, *Improving Quadrotor Trajectory Tracking by Compensating for Aerodynamic Effects*. 2017. doi: 10.0/Linux-x86_64.
[41] M. Faessler, A. Franchi, and D. Scaramuzza, “Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories,” *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, Apr. 2018, doi: 10.1109/LRA.2017.2776353.
[42] F. Götten, M. Havermann, C. Braun, M. Marino, and C. Bil, “Wind-tunnel and CFD investigations of UAV landing gears and turrets – Improvements in empirical drag estimation,” *Aerospace Science and Technology*, vol. 107, Dec. 2020, doi: 10.1016/j.ast.2020.106306.
[43] “*Autodesk Inventor Professional 2021®*.” [Online]. Available: https://www.autodesk.pt/products/inventor/overview?term=1-YEAR&tab=subscription
[44] “*Pixhawk®* 4 Mini.” https://docs.px4.io/master/en/flight_controller/pixhawk4_mini.html (accessed Dec. 21, 2020).
[45] “*Adafruit HUZZAH32 – ESP32*.” https://www.adafruit.com/product/3405 (accessed Dec. 22, 2020).
[46] “*Savox Servo SC1251MG*.” https://www.savox-servo.com/Servos-c-1338/Coreless-Motor-c-1348/Savox-Servo-SC-1251MG-Digital-Coreless-Motor-Metal-Gear/ (accessed Dec. 22, 2020).

[47] "*PX4FLOW* Smart Camera." https://docs.px4.io/v1.12/en/sensor/px4flow.html#px4flow-smart-camera
[48] "MB1043 HRLV-MaxSonar-EZ4." https://www.maxbotix.com/ultrasonic_sensors/mb1043.htm
[49] "FLIR Lepton 3.5." https://groupgets.com/manufacturers/teledyne-flir/products/lepton-3-5
[50] "*FLIR Lepton® Breakout Board* v2.0." https://www.flir.com/products/lepton-breakout-board-v2.0/
[51] "*Raspberry Pi* 4." https://www.raspberrypi.com/products/raspberry-pi-4-model-b/
[52] "raspberrypi_video." https://github.com/groupgets/LeptonModule/tree/master/software/raspberrypi_video
[53] "*JR3 PCI* Force Torque Sensor." https://docs.quanser.com/quarc/documentation/jr3_pci_force_torque_sensor_block.html
[54] A. D. Ferreira and M. R. M. Fino, "A wind tunnel study of wind erosion and profile reshaping of transverse sand piles in tandem," *Geomorphology*, vol. 139–140, pp. 230–241, Feb. 2012, doi: 10.1016/j.geomorph.2011.10.024.
[55] "*UAVCAN*." https://uavcan.org/
[56] "*Mavlink* V1." https://github.com/mavlink/c_library_v1
[57] "*Mavlink* V2." https://github.com/mavlink/c_library_v2
[58] "Common library." https://mavlink.io/en/messages/common.html
[59] "Heartbeat message." https://github.com/mavlink/c_library_v1/blob/master/minimal/mavlink_msg_heartbeat.h
[60] "Attitude message." https://github.com/mavlink/c_library_v2/blob/master/common/mavlink_msg_attitude.h
[61] "Velocities message." https://github.com/mavlink/c_library_v2/blob/master/common/mavlink_msg_local_position_ned.h
[62] "Optical Flow message." https://github.com/mavlink/c_library_v2/blob/master/common/mavlink_msg_distance_sensor.h
[63] "Controlling Drone." https://github.com/mavlink/c_library_v2/blob/master/common/mavlink_msg_command_int.h
[64] "*QGroundControl*." http://qgroundcontrol.com/
[65] "PX4." https://px4.io/
[66] "*Ardupilot*." https://ardupilot.org/
[67] "Access mode vs Atation mode." https://randomnerdtutorials.com/esp32-access-point-ap-web-server/

ANNEX A – UAV ASSEMBLY AND DIMENSIONS







APPENDIX A - ALGORITHMS

Algorithm 1 – Send Hotspot Coordinates from *Raspberry Pi 4®* to the Microcontroller (*Python*);

Algorithm 2 – Stream the IR video from the *Raspberry Pi 4®* to the GUI (*Python*);

Algorithm 3 – Receive the IR video stream in the GUI (*C#*);

Algorithm 4 – 1st method for Wind Estimation (*C++*);

Algorithm 5 – 2nd method for Wind Estimation (*C++*);

Algorithm 6 – *MAVLink®* communication setup (*C++*);

Algorithm 7 – UAV's attitude control (*C++*);

Algorithm 8 – *Wi-Fi* connection setup in the microcontroller (*C++*);

Algorithm 9 – *Wi-Fi* connection setup in the GUI (*C#*);

Algorithm 10 –Receiving data and control UAV's attitude in the GUI (*C#*);

Algorithm 1 – Send Hotspot Coordinates from *Raspberry Pi 4®* to the Microcontroller (*Python*);

```
1.  #!/usr/bin/env python3
2.  import serial
3.  import time
4.  if __name__ == '__main__':
5.      ser = serial.Serial('/dev/ttyUSB0', 57600, timeout=1) #Define the Port and
    Baudrate
6.      ser.flush()
7.      while True:
8.          with open('valores') as f:
9.              contents = f.read()     #Read the content of the file
10.             print(contents)         #to which the coordinates are being printed
11.             ser.write((str(contents)+"\n").encode('utf-8'))    #Send it to the
    microcontroller
12.             time.sleep(0.1)
```

Algorithm 2 – Stream the IR video from the *Raspberry Pi 4®* to the GUI (*Python*):

```
1.  #!/usr/bin/env python3
2.
3.  import socket
4.  import cv2
```

```
import time

serversocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  #TCP
serversocket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

serversocket.bind(("192.168.4.3", 9999))  #Search in this IP and port

serversocket.listen(5)
print ('server started and listening')

try:
    while 1:
        (clientsocket, address) = serversocket.accept()  #Accept the client
        print ("connection found!")

        img = cv2.imread('testImage.jpeg')  #Use OpenCV to read the image

        clientsocket.sendall(img)  #Send the image

        clientsocket.close()  #Close the socket
except:
    clientsocket.close()
```

Algorithm 3 – Receive the IR video stream in the GUI (*C#*);

```
private void connectVideoSocket()
        {
            VideoSocket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);  //TCP socket
            IPEndPoint hostIpEndPoint = new
IPEndPoint(IPAddress.Parse("192.168.4.3", 9999);  //Define IP and port
            VideoSocket.Connect(hostIpEndPoint);
        }
        private void btnStreamVideo_Click(object sender, EventArgs e)
        {
            if (videostream == 0) {   //Change the variable value each time a
click is done
                videostream = 1;
            }
            else {
                videostream = 0;
            }

            int j = 0;
            for (j = 1; j < 6; j++) {        //5 tries to connect
                try
                {
                    while (videostream == 1)    //While videostream=1, connect
the socket, wait 0.15s and receive the array
                    {
                        connectVideoSocket();
                        wait(150);
                        int dataSize;
                        int y, x, t = 0, g = 0;

                        Bitmap bmp2 = new Bitmap(160, 120);

                        dataSize = 0;
                        byte[] b = new byte[VideoSocket.ReceiveBufferSize];
                        dataSize = VideoSocket.Receive(b);
```

```
                        if (dataSize > 0)
                        {
                            for (y = 0; y < 120; y++)
                            {
                                for (x = 0; x < 160; x++)
                                {                                   //Assign the read
value to the color of the respective pixel
                                    bmp2.SetPixel(x, y,
Color.FromArgb(Convert.ToInt32(b[t + 2]), Convert.ToInt32(b[t + 1]),
Convert.ToInt32(b[t])));
                                    t = t + 3;     //The array has 3 different
values for each pixel (RGB)
                                }
                            }
                        }
                        picBoxSensorIR.Image = bmp2;    //assign the Bitmap value
to the Picture Box.
                        picBoxSensorIR.SizeMode = PictureBoxSizeMode.Zoom;

                        VideoSocket.Close();
                        wait(10);
                    }
                }
                catch (Exception)
                {
                    MessageBox.Show("Impossible to connect! Try again later");
                }
                wait(100);      //wait 0,1s between each try
            }
            picBoxSensorIR.Image = null;
        }
```

Algorithm 4 – 1st method for Wind Estimation (*C++*);

```
void method_1() {
  float mass;
  float air_density;
  float roll;
  float pitch;
  float compass_angle;
  float ground_vel;
  float ground_ang;

  ground_ang=atan(speedx/speedy);

     float den1 = sqrt(pow(sin(pitch),2) * pow(cos(roll), 2) + pow(cos(pitch),
2));
     float inclin_angle;
     if (den1 != 0) {          //To prevent indeterminate equations
     inclin_angle = acos(cos(pitch)*cos(roll)/den1);
     }

     float drag_force = 9.81 * mass * tan(inclin_angle);

  if (abs(pitch)>2*abs(roll)) {       //If the pitch angle is bigger than the
double of roll, use the equations for yaw=0º
    proj_area = ProjArea_0(ToDegrees(inclin_angle));
    drag_coef = DragCoef_0(ToDegrees(inclin_angle));
  }
  else if (abs(roll)>2*abs(pitch)) {   //If the roll angle is bigger than the
double of pitch, use the equations for yaw=90º
```

```
25.     proj_area = ProjArea_90(ToDegrees(inclin_angle));
26.     drag_coef = DragCoef_90(ToDegrees(inclin_angle));
27.   }
28.   else {    //Else, use the equations for yaw=45º
29.     proj_area = ProjArea_45(ToDegrees(inclin_angle));
30.     drag_coef = DragCoef_45(ToDegrees(inclin_angle));
31.   }
32.
33.   float flight_speed = sqrt(abs(2*drag_force / (air_density * proj_area *
    drag_coef)));
34.
35.   float den2 = sqrt(pow(sin(pitch), 2) * pow(cos(roll), 2) + pow(cos(pitch),2) *
    pow(sin(roll), 2));
36.
37.   float lambda;
38.   if (den1 != 0) {
39.   lambda = acos(cos(roll)*sin(pitch) / den2);
40.   }
41.
42.   float flight_angle;
43.   if (-sin(roll) * cos(pitch) < 0)  {
44.      flight_angle = 2 * PI - lambda + compass_angle;
45.    }
46.   else {
47.    flight_angle = lambda + compass_angle;
48.   }
49.
50.   if (flight_angle>2*PI) {       /* maybe a function*/
51.   flight_angle = flight_angle - 2 * PI;
52.   }
53.
54.   float alpha;
55.   if (abs(ground_ang-flight_angle) < PI) {        //needs to be between 0 and
    180º
56.     alpha = ground_ang - flight_angle;
57.   }
58.   else {
59.     alpha=2*PI-abs(ground_ang-flight_angle);
60.   }
61.
62.   wind_speed = sqrt(pow(flight_speed, 2) + pow(ground_vel, 2) - 2 * flight_speed
    * ground_vel * cos(alpha));
63.
64.   float den3 = -2 * ground_vel * wind_speed;
65.
66.   float beta;
67.   if (den3 !=0) {
68.   beta = acos((pow(flight_speed, 2) - pow(ground_vel, 2) - pow(wind_speed, 2)) /
    den3);
69.   }
70.
71.   if (flight_angle < ground_ang || flight_angle > PI + ground_ang) {
72.      wind_angle = flight_angle + PI + beta;
73.       }
74.    else {
75.      wind_angle = flight_angle + PI - beta;
76.        }
77.
78.   if (wind_angle > 2 * PI)  {
79.       wind_angle = wind_angle - 2 * PI;
80.    }
81. }
```

```
82.
83. float ProjArea_0(float x) {
84.   float y = 0.00000162*pow(x,3) - 0.00014125*pow(x,2) + 0.00469925*x +
    0.04737158;
85. }
86.
87. float DragCoef_0(float x) {
88.   float y = -0.0000467*pow(x,3) + 0.00353*pow(x,2) - 0.0836*x + 1.527;
89.   return y;
90. }
91.
92. float ProjArea_45(float x) {
93.   float y = 0.00000089*pow(x,3) - 0.00008639*pow(x,2) + 0.00345142*x +
    0.05565395;
94.   return y;
95. }
96.
97. float DragCoef_45(float x) {
98.   float y = -0.00002985*pow(x,3) + 0.002192*pow(x,2) - 0.05145*x + 1.3502;
99.   return y;
100.       }
101.
102.       float ProjArea_90(float x) {
103.         float y = 0.00000117*pow(x,3) - 0.00011483*pow(x,2) + 0.00450221*x +
    0.04750834;
104.         return y;
105.       }
106.
107.       float DragCoef_90(float x) {
108.         float y = -0.00003391*pow(x,3) + 0.00282*pow(x,2) - 0.07675*x + 1.56433;
109.         return y;
110.       }
```

Algorithm 5 – 2nd method for Wind Estimation (*C++*);

```
1.  void method_2() {
2.
3.   float roll_min=3.14;
4.   float yaw_maxwind;
5.
6.    for (int yang=0; yang<179; yang = yang + 1){   //Loop to increase the desired
    yaw angle in each step
7.
8.      change_yaw(yang, -1, 0);  // send to the yang -> change_yaw(desired angle,
    direction of rotation, real value or relative offset)
9.      delay(2000);    //Wait 2 seconds to make it reach the desired position
10.     Mav_Request_Data();
11.     comm_receive();                         //Read Roll and Yaw
12.      if (abs(roll) < abs(roll_min)) {
13.      roll_min = roll;           //If the new roll is smaller than the minimum,
    update roll_min
14.      yaw_maxwind = yang;        //And also the yaw_maxwind
15.      }
16.   }
17.
18.   change_yaw(yaw_maxwind, -1, 0);    //Send to the yaw to start the calculation
19.
20.   Mav_Request_Data();   // Read roll, pitch, compass, and GPS speed and angle
21.   comm_receive();
22.
23.   //From now on, the algorithm is similar to method1!
```

```
24. }
```

Algorithm 6 – *MAVLink*® communication setup (*C++*);

```
1.  void pack_message() {
2.
3.    unsigned long previousMillisMAVLink = 0;     // will store last time MAVLink
    was transmitted and listened
4.    unsigned long next_interval_MAVLink = 1000;  // next interval to
    count
5.    const int num_hbs = 5;               // # of heartbeats to wait before
    activating STREAMS from Pixhawk. 60 = one minute.
6.    int num_hbs_pasados = num_hbs;
7.
8.    // MAVLink config
9.    /* The default UART header for your MCU */
10.   int sysid = 1;                   // PX
11.   int compid = 2;                // The component sending the message
12.   int type = 2;                  //  MAV_TYPE_QUADROTOR;
13.
14.   // Define the system type, in this case a Quadcopter -> on-board controller
15.   uint8_t system_type = 0;                  // MAV_TYPE_GENERIC;
16.   uint8_t autopilot_type = 12;           // MAV_AUTOPILOT_PX4;
17.
18.   uint8_t system_mode = 0;    // MAV_MODE_PREFLIGHT ///< Booting up
19.   uint32_t custom_mode = 0;     /// Custom mode, can be defined by user/adopter
20.   uint8_t system_state = 3;    // MAV_STATE_STANDBY; ///< System ready for flight
21.
22.   mavlink_message_t msg;          // Initialize the required buffers
23.   uint8_t buf[MAVLINK_MAX_PACKET_LEN];
24.
25.    // Pack the message
26.   mavlink_msg_heartbeat_pack(1,2, &msg, type, autopilot_type, system_mode,
    custom_mode, system_state);
27.
28.   // Copy the message to the send buffer
29.   uint16_t len = mavlink_msg_to_send_buffer(buf, &msg);
30.
31.   // Send the message with the standard UART send function
32.   unsigned long currentMillisMAVLink = millis();
33.   if (currentMillisMAVLink - previousMillisMAVLink >= next_interval_MAVLink) {
34.     previousMillisMAVLink = currentMillisMAVLink;
35.
36.     Serial1.write(buf, len);
37.
38.     //Mav_Request_Data();
39.     num_hbs_pasados++;
40.     if(num_hbs_pasados>=num_hbs){
41.      Mav_Request_Data();  //Request streams from Pixhawk
42.      num_hbs_pasados=0;
43.     }
44.    }
45. }
46.
47. void Mav_Request_Data()
48. {
49.   mavlink_message_t msg;
50.   uint8_t buf[MAVLINK_MAX_PACKET_LEN];
51.
52.   const int  maxStreams = 3;   //Stream just the messages we want
```

```
53.   const uint8_t MAVStreams[maxStreams] = {MAV_DATA_STREAM_EXTENDED_STATUS,
    MAV_DATA_STREAM_POSITION, MAV_DATA_STREAM_EXTRA3};
54.   const uint16_t MAVRates[maxStreams] = {0x01, 0x01, 0x03};
55.
56.   for (int i=1; i < maxStreams+1; i++) {
57.     mavlink_msg_request_data_stream_pack(1, 2, &msg, 0, 0, MAVStreams[i],
    MAVRates[i], 1);
58.     uint16_t len = mavlink_msg_to_send_buffer(buf, &msg);
59.     Serial1.write(buf, len);
60.   }
61. }
62.
63. void comm_receive() {
64.
65.   mavlink_message_t msg;
66.   mavlink_status_t status;
67.
68.   while(Serial1.available()>0) {  // Read meanwhile there is data from the FC
69.
70.     uint8_t c = Serial1.read();
71.
72.     if(mavlink_parse_char(MAVLINK_COMM_0, c, &msg, &status)) {
73.
74.       switch(msg.msgid) {     //Handle each message
75.
76.         case MAVLINK_MSG_ID_HEARTBEAT:  // #0:Heartbeat
77.           {
78.             Serial.println("Flight Controller HeartBeat");
79.           }
80.           break;
81.
82.          case MAVLINK_MSG_ID_ATTITUDE:  // #30
83.           {
84.             mavlink_attitude_t attitude;
85.             mavlink_msg_attitude_decode(&msg, &attitude);
86.             roll=attitude.roll;
87.             pitch=attitude.pitch;
88.             yaw=attitude.yaw;
89.           }
90.
91.           case MAVLINK_MSG_ID_OPTICAL_FLOW_RAD: //#106     //Get velocities, if
    in indoor tests
92.           {
93.             mavlink_optical_flow_rad_t optical_flow_rad;
94.             mavlink_msg_optical_flow_rad_decode(&msg, &optical_flow_rad);
95.             if (outdoor=false) {
96.               groundspeed=sqrt(pow(optical_flow_rad.opt_mx,2)+pow(optical_flow_ra
    d.opt_my,2))/*?*/;
97.               groundangle=atan(optical_flow_rad.opt_my/optical_flow_rad.opt_mx);
      ////////////////////////////////atan em graus ou rad?
98.             }
99.           }
100.
101.                case MAVLINK_MSG_ID_LOCAL_POSITION_NED: //#32   //Get
    velocities, if in outdoor tests.
102.                {
103.                  mavlink_local_position_ned_t local_position_ned;
104.                  mavlink_msg_local_position_ned_decode(&msg,
    &local_position_ned);
105.                  if (outdoor==true) {
106.                  speedx = local_position_ned.vx;
107.                  speedy = local_position_ned.vy;
108.                  speedz = local_position_ned.vz;
```

```
                    ground_vel=sqrt(pow(speedx, 2)+pow(speedy, 2)+pow(speedz, 2));
                    }
                  }
              }
            }
          }
```

Algorithm 7 – UAV's attitude control (*C++*);

```
void change_yaw(float heading, float directioning, float is_absolute_ang) {
//Code to control the UAV's orientation
  mavlink_message_t msg_send;
  uint8_t buf2[MAVLINK_MAX_PACKET_LEN];

  mavlink_msg_command_int_pack(255, 1, &msg_send, 1, 1, 0, MAV_CMD_CONDITION_YAW
/*115*/, 0, 0, heading, 5 /*deg/s*/, directioning, is_absolute_ang, 0, 0, 0);

  uint16_t len2 = mavlink_msg_to_send_buffer(buf2, &msg_send);
  Serial1.write(buf2, len2);
}

void move_UAV(float x, float y, float z) {    //Code to move the UAV
  mavlink_message_t msg_send;
  uint8_t buf[MAVLINK_MAX_PACKET_LEN];

 mavlink_msg_set_position_target_local_ned_pack(255, 1, &msg_send, 0, 1, 1,
MAV_FRAME_LOCAL_OFFSET_NED, 0b0000111111111000 /*only positions enabled*/, x, y ,
z, 0, 0, 0, 0, 0, 0, 0, 0);

  uint16_t len = mavlink_msg_to_send_buffer(buf, &msg_send);
  Serial1.write(buf, len);
}
```

Algorithm 8 – *Wi-Fi* connection setup in the microcontroller (*C++*);

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiAP.h>
#include <WiFiUdp.h>
#include <Udp.h>

const char* ssid = "UAV_WiFi";
const char* password = "123456789";

WiFiServer server(80);    //Port 80 - TCP
WiFiClient client;
WiFiUDP Udp;
const int udpPort = 80;  //Port 80 - UDP

void setup() {
  Serial.begin(9600);

  Serial.print("Setting AP (Access Point)…");
  WiFi.softAP(ssid, password);    //Create the Access Point
  IPAddress IP = WiFi.softAPIP();

  Udp.begin(udpPort);
  delay(1000);  //To be able to use ADC pins on the microcontroller
```

```
24. }
25.
26. void loop() {
27.
28.   client = server.available();   // Listen for incoming clients - TCP
29.   int packetSize = Udp.parsePacket();   //Variable that checks if there is a UDP
    package
30.
31. //Each of these conditions begins a transmission dependind on each protocol, and
    sends the package
32. //str_reply varies with the request from the user
33.   if (client) {
34.       client.print(str_reply);      //TCP CODE
35.       client.stop();
36.   }
37.   else if (packetSize) {
38.     Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());   //UDP CODE
39.     Udp.println(str_reply);
40.     Udp.endPacket();
41.   }
```

Algorithm 9 – *Wi-Fi* connection setup in the GUI (*C#*);

```
1.  public void initiate_connection(string esp32_ip, string esp32_port, int
    protocoltype) //this void receives the IP and the Port of the microcontroller,
    and also the desired protocol, and establishes the connection
2.          {
3.              string msg;
4.              int port;
5.
6.              if (protocoltype == 1)
7.              {
8.                  protype = ProtocolType.Udp;
9.                   socktype = SocketType.Dgram;
10.             }
11.             else if (protocoltype == 2)
12.             {
13.                 protype = ProtocolType.Tcp;
14.                 socktype = SocketType.Stream;
15.             }
16.             try
17.             {
18.                 s = new Socket(AddressFamily.InterNetwork, socktype, protype);
19.                 msg = esp32_ip;
20.                 IPAddress adafruit_remote_IP = IPAddress.Parse(msg);
21.                 port = int.Parse(esp32_port);
22.                 IPEndPoint IP_Endpoint_remote = new
    IPEndPoint(adafruit_remote_IP, port);
23.                  s.Connect(IP_Endpoint_remote);
24.             }
25.             catch (SocketException se)
26.             {
27.                 MessageBox.Show(se.Message);
28.             }
29.         }
30.
31.         private string send_command_adafruit(string msg)  //This string allows to
    send a command to the microcontroller, and receive its response
32.         {
33.             byte[] sent_data;
34.             byte[] received_data = new byte[256];
```

```
35.
36.             int number_of_received_bytes;
37.
38.             try
39.             {
40.                 if (s.Connected)
41.                  {
42.                      sent_data =
    System.Text.ASCIIEncoding.ASCII.GetBytes(msg);  //It uses the ASCII encoding
43.                      s.Send(sent_data);
44.                      number_of_received_bytes = s.Receive(received_data);
45.                      msg = System.Text.Encoding.ASCII.GetString(received_data, 0,
    number_of_received_bytes);
46.                      return msg;
47.                  }
48.                 else
49.                 {
50.                     MessageBox.Show("Application Message: Socket NOT connect,
    verify your cables and conections.");
51.                     return "-1";
52.                 }
53.             }
54.             catch (SocketException se)
55.             {
56.                 MessageBox.Show(se.Message);
57.                 return "-1";
58.             }
59.         }
```

Algorithm 10 – Receiving data and control UAV's attitude in the GUI (*C#*);

```
1.  while (checkBox1.Checked == true)
2.              {
3.                  answer = send_command_adafruit("!"); //sending a '!' asks the
    microcontroller for all the parameters, which will be saved in the variable
    'answer' separated by " "
4.                  txtMessageList.AppendText(answer);
5.                  txtMessageList.AppendText("\r\n");
6.
7.                  for (i = 1; i < 14; i = i + 1)       //Loop to read the full
    stream, and save each parameter separately
8.                  {
9.                      len_ans = answer.Length;
10.                     index_of_S = answer.IndexOf(" ");
11.
12.                     if (index_of_S > -1)
13.                     {
14.                         parameter[i] = answer.Substring(0, index_of_S);
15.                         answer = answer.Substring(index_of_S + 1, len_ans -
    index_of_S - 1);
16.                     }
17.                     else
18.                     {
19.                         if (len_ans > 0)
20.                         {
21.                             parameter[i] = answer;
22.                             answer = "";
23.                         }
24.                         else parameter[i] = "ERROR";
25.                     }
```

```
                }
               //Print each parameter in the respective TextBox
            txtReadRoll.Text = parameter[1];txtReadPitch.Text = parameter[2];
            txtReadYaw.Text = parameter[3]; txtVx.Text = parameter[4];
            txtVy.Text = parameter[5];  txtVz.Text = parameter[6];
            txtVtot.Text = parameter[7]; txtHeight.Text = parameter[8];
            txtCompassAngle.Text = parameter[9]; xtWindSpeed.Text = parameter[10];
            txtDirection.Text = parameter[11]; txtHotX.Text = parameter[12];
            txtHotY.Text = parameter[13];
}
```

APPENDIX B - VERIFICATION OF THE WIND ESTIMATION ALGORITHM

Microsoft Excel® Sheet:

Inputs		
GPS		
r_w	0.25	[m/s]
θ_w	3.2	[rad]
IMU		
Roll [Ø]	0.1	[rad]
Pitch [θ]	0.02	[rad]
Compass Ang. [δc]	0.01	[rad]
C_D	1.20546	
Ap	0.07012	[m^2]

Constants	
g =	9.81 m/s^2
m =	3.2 kg
ρ =	1.222 kg/m^3

Intermediate Steps		
Variable		Degrees
λ =	1.374017	78.72536
ψ =	0.101954	5.841555
θ_v=	4.919169	
F_D =	3.211687	
r_v =	7.885682	
α =	1.719169	98.5011
β =	1.391226	79.71136

Final Results		
Variable		Degrees
r_u =	7.926496	
θ_u =	0.38635	22.13623

Results in the algorithm implemented in the microcontroller:

```
float mass=3.2;
float air_density = 1.222;
float proj_area;
float drag_coef;

float roll=0.1;
float pitch=0.02;
float compass_angle=0.01;
float ground_ang=3.2;
float ground_vel=0.25;
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
wind speed=7.93
wind angle=0.39
```