Department of Information Science and Technology

# Autonomous Obstacle Collision Avoidance System for UAVs in Rescue Operations

António Sérgio Lima Raimundo

A Dissertation presented in partial fulfillment of the Requirements for the Degree of

Master in Computer Science Engineering

Supervisor:
PhD. Pedro Joaquim Amaro Sebastião,
Assistant professor at ISCTE-IUL

Co-supervisor:
PhD. Nuno Manuel Branco Souto,
Assistant professor at ISCTE-IUL

October, 2016

# ABSTRACT

The Unmanned Aerial Vehicles (UAV) and its applications are growing for both civilian and military purposes. The operability of an UAV proved that some tasks and operations can be done easily and at a good cost-efficiency ratio.

Nowadays, an UAV can perform autonomous tasks, by using waypoint mission navigation using a GPS sensor. These autonomous tasks are also called missions. It is very useful to certain UAV applications, such as meteorology, vigilance systems, agriculture, environment mapping and search and rescue operations.

One of the biggest problems that an UAV faces is the possibility of collision with other objects in the flight area. This can cause damage to surrounding area structures, humans or the UAV itself. To avoid this, an algorithm was developed and implemented in order to prevent UAV collision with other objects.

“Sense and Avoid” algorithm was developed as a system for UAVs to avoid objects in collision course. This algorithm uses a laser distance sensor called LiDAR (Light Detection and Ranging), to detect objects facing the UAV in mid-flights. This light sensor is connected to an on-board hardware, Pixhawk’s flight controller, which interfaces its communications with another hardware: Raspberry Pi. Communications between Ground Control Station or RC controller are made via Wi-Fi telemetry or Radio telemetry.

“Sense and Avoid” algorithm has two different modes: “Brake” and “Avoid and Continue”. These modes operate in different controlling methods. “Brake” mode is used to prevent UAV collisions with objects when controlled by a human operator that is using a RC controller. “Avoid and Continue” mode works on UAV’s autonomous modes, avoiding collision with objects in sight and proceeding with the ongoing mission.

In this dissertation, some tests were made in order to evaluate the “Sense and Avoid” algorithm’s overall performance. These tests were done in two different environments: A 3D simulated environment and a real outdoor environment. Both modes worked successfully on a simulated 3D environment, and “Brake” mode on a real outdoor, proving its concepts.

***Keywords*:** UAV, object collision avoidance, distance sensing, LiDAR, 3D environment, drone, 3D vehicle.

*This page was intentionally left in blank*

# RESUMO

Os veículos aéreos não tripulados (UAV) e as suas aplicações estão cada vez mais a ser utilizadas para fins civis e militares. A operacionalidade de um UAV provou que algumas tarefas e operações podem ser feitas facilmente e com uma boa relação de custo-benefício. Hoje em dia, um UAV pode executar tarefas autonomamente, usando navegação por *waypoints* e um sensor de GPS. Essas tarefas autónomas também são designadas de missões. As missões autónomas poderão ser usadas para diversos propósitos, tais como na meteorologia, sistemas de vigilância, agricultura, mapeamento de áreas e operações de busca e salvamento. Um dos maiores problemas que um UAV enfrenta é a possibilidade de colisão com outros objetos na área, podendo causar danos às estruturas envolventes, aos seres humanos ou ao próprio UAV. Para evitar tais ocorrências, foi desenvolvido e implementado um algoritmo para evitar a colisão de um UAV com outros objetos.

O algoritmo "Sense and Avoid" foi desenvolvido como um sistema para UAVs de modo a evitar objetos em rota de colisão. Este algoritmo utiliza um sensor de distância a laser chamado LiDAR (Light Detection and Ranging), para detetar objetos que estão em frente do UAV. Este sensor é ligado a um hardware de bordo, a controladora de voo Pixhawk, que realiza as suas comunicações com outro hardware complementar: o Raspberry Pi. As comunicações entre a estação de controlo ou o operador de comando RC são feitas via telemetria Wi-Fi ou telemetria por rádio. O algoritmo "Sense and Avoid" tem dois modos diferentes: o modo "Brake" e modo "Avoid and Continue". Estes modos operam em diferentes métodos de controlo do UAV. O modo "Brake" é usado para evitar colisões com objetos quando controlado via controlador RC por um operador humano. O modo "Avoid and Continue" funciona nos modos de voo autónomos do UAV, evitando colisões com objetos à vista e prosseguindo com a missão em curso. Nesta dissertação, alguns testes foram realizados para avaliar o desempenho geral do algoritmo "Sense and Avoid". Estes testes foram realizados em dois ambientes diferentes: um ambiente de simulação em 3D e um ambiente ao ar livre. Ambos os modos obtiveram funcionaram com sucesso no ambiente de simulação 3D e o mode “Brake” no ambiente real, provando os seus conceitos.

***Palavras-chave***: UAV, colisão de objetos, deteção de distância, LiDAR, ambiente 3D, drone, veículo 3D.

*This page was intentionally left in blank*

# ACKNOWLEDGMENTS

First of all, I would like to my family for their continuous support and confidence. They played an important role on giving me the required conditions in order to accomplish this dissertation development, and for helping me getting my masters degree.

My greater acknowledgement is for Professor Pedro Sebastião for giving me the opportunity of developing this dissertation as my research mentor, and for making this dissertation and R&D project possible. I’m grateful for his continuous advice on different subject matters, such as communications methods, electronic links between components, experimental work tips and many other cases. I had all the conditions needed to develop this work, and his knowledge was outstanding, which added a relevant and considerate value to this work. Alongside with this dissertation development, I got the opportunity to participate on various external activities that were essential to show to the outside community what projects ISCTE-IUL and IT-IUL were currently developing.

I also want to thank Professor Nuno Souto for his solutions to presented problems. They were very useful and it guaranteed a continuous work flow.

A big thanks to my project teammates: Diogo Peres and Nuno Santos. They were without a doubt, an essential help on early, mid and last dissertation development stages. Together, we accomplished many goals such as successful early UAV experimental flights, connections to Raspberry Pi, and many others.

Last but not least, on this dissertation development, I also had friend support from: Ricardo Silva, Manuel Oliveira, André Glória and João Pavia. To them, I would like to give my special thanks for their contributions on this dissertation development, external activities, and personal opinions and advices. Their help increased this dissertation’s value on various aspects.

*This page was intentionally left in blank*

# Chapter 1
# INTRODUCTION

This chapter presents the concept and definition of this dissertation development, such as motivation, target goals and its research methods and questions. It is also mentioned what contributions this work provided. A visual scheme of this dissertation structure is also present.

## 1.1 Overview

Humans are always searching for a better solution to resolve day-to-day problems. There is always something that needs to be more practical, efficient, affordable or even easier to interact with human beings. Due to that fact, there was a necessity of creating something that resolve those problems in a way that can guarantee same operational functions as human operators or even new ways to do the same thing with less effort. To accomplish that, Unmanned Aerial Vehicles (UAVs) were developed to preform actions that were difficult for human beings to execute. They are different from Manned Aerial Vehicles. UAVs are remotely controlled and Manned Aerial Vehicles are locally controlled. UAVs are characterized by their size, weight and maneuver. They are very small and light-weighted when compared to real manned aerial vehicles, such as commercial travel planes or even civil aircrafts. Their initial purpose was to serve the military for executing scout and surveillance missions, and even to preform aerial attacks in Gulf War and more recent, Iraq. (Cocaud, 2007). Nowadays, UAVs are used to perform several tasks that require fast and efficient methods, such as analyzing crash sites, firefighting, search and rescue processes and so on. (Scherer, 2015) There are various types of UAVs, each one with its limitations, functionalities and purposes. In order to control those UAVs, it’s necessary to have a ground control station (GCS), where communication is established between UAV and GCS, via wireless connection. There are several ways to communicate wirelessly, which can be via Radio, Satellite or Mobile Networks. (Murilhas, 2015) They can perform autonomous actions, given by manned remote control, such as executing missions that are global positioning system (GPS)-guided through waypoints marked in a map. In a mission, user controller orders UAV to flight through pre-defined GPS coordinates. However, like every other manned control processes, error could occur. Consequently, there is a necessity to develop intelligent systems that can mitigate, or even avoid completely human-caused problems. An UAV, in order to make decisions by itself, has to be prepared to take emergency actions. Data indicators can trigger an emergency state status such as fast altitude drop, communication failure, low battery and risk of collision (for object collision avoidance).

## 1.2 Motivation

UAVs and its technology have helped global industry to increase its production in the past years. Developing solutions that can perform industry-specified tasks has increased drone global market. Industries like agriculture, energy, utilities, public service, mining, construction, real estate, news media, and film production can now rely on UAVs to improve process performance, lowering costs and risks for human beings. It's expected that the use of UAVs to have a compound annual growth rate of 19% for civilian purposes, when compared with 5% for military services, both between 2015 and 2020 (Insider, 2015).

Growth rate of civilian purposes are much greater than military ones, because UAVs (and more specifically Micro UAVs) can now be used by civilians who wish to record aerial views, such as image recording, filmmaking, racing or just for fun, at affordable costs. Initially, UAVs were only used for military purposes such as surveillance and scouting missions. Now, it's being used by governmental public services to prevent forest fires (Wall, 2011), police patrolling and border control (Spagat & Skoloff, 2014).

Search and Rescue (SAR) operations have to find a way to provide aid to a person or people that are in imminent danger as fast as possible. There are different methods to support SAR teams, and depending on the area, they can perform tasks that a human being cannot or it takes too much time to accomplish the same result. Rescue dogs can search for dead bodies, missing persons, drug smuggling; Boats rescue people from drowning, perform immigrant control, fishing control; Planes and helicopters can extinguish fires, rescue people from inaccessible ground ways (Scherer, 2015); There are so many things that SAR services and tools can do, but there are ways to do the same thing with less effort, avoiding risks for humans, using the UAVs to operate some kind of works. For example: An UAV can rescue a person from drowning in the sea by simply sending it with an attached buoy, by air, to the location of the person, throw the buoy and observe if person grabs the buoy. Then, if necessary, send a sea rescuer to help the person in need; or can extinguish fires that are too difficult to access or even inaccessible (Scherer, 2015).

Companies like Facebook and Amazon are investing massively in drone solutions. For example: Facebook launches a solar-powered drone to provide internet access in remote zones (Hern, 2015); Amazon recently launched a new prototype drone capable of delivering small order packages that hovers vertically to a height of approx. 120 m, and then flies horizontally for up to approx. 24 km (Murgia, 2015).

This dissertation is a module part of a Research & Development project team, supported by IT-IUL Instituto de Telecomunicações @ ISCTE-IUL, which is a Portuguese research and development non-profit organization in the field of telecommunications ("IT - Instituto de Telecomunicações," n.d.). The main motivation to develop the work presented in this dissertation was my participation in the project for building a full system for controlling and monitoring a 3D vehicle, which is a hybrid unmanned vehicle that can be controlled on air (UAV – Unmanned Aerial Vehicles), ground (UGV – Unmanned Ground Vehicles) and water surfaces (USV – Unmanned Surface Vehicles). This dissertation fulfills a part of project's security module, which aims mainly on avoiding UAV crashes by making autonomous decisions; Waypoint navigation through guided missions; Object collision avoidance maneuvers, using an algorithm that can "see and avoid" objects.

## 1.3 Goals

This dissertation's main goal is to develop an UAV system to perform search and rescue operations autonomously and almost risk-free for humans and for the UAV itself. To accomplish that, a subdivision of the main goal was made in three distant modules:

- Guided missions through pre-defined GPS provided waypoints: In this module, an UAV is developed to perform autonomous missions given by a human operator;
- Human error avoidance: In this module, we control user behaviors. Such behaviors can cause UAV to crash, which consists risk for humans being hit by an UAV. To prevent that, user controls are monitored to evaluate if an UAV is being instructed to hit a solid object by accident. If that occurs, a smart background "sense and avoid" algorithm take control and perform a pre-configured action;
- Object collision avoidance system: This is the main module of this dissertation. It improves previous one, in order to perform the same tasks, but avoiding collision autonomously with visible objects through a laser sensor. Human interaction reduces significantly, because there is no need to worry about possible objects that may appear in the way.

By achieving these goals, the present solution can be the solution for a variety of problems that can simplify the use of UAVs for search and rescue operations, and consequently save the life of people.

## 1.4 Research Questions & Methods

In order to proceed with this research, some questions have to be raised. And those questions can be answered by solutions presented in 1.3 (Goals). These solutions can provide working methods to resolve existing problems. Questions raised can be the following:

- How to develop solutions that can be cost-effective, efficient and with low risk for human beings, to perform Search and Rescue operations?
- By developing those solutions, which way should be followed in order to achieve dispensable human actions? What actions can be disposed in order to perform same tasks and operations?
- Dispensing human actions can cause other unresolved problems. Is it really necessary? If so, how to solve those problems?
- Will this solution be doable to implement? Will that help Search and Rescue teams to achieve same goals as human-allocated tasks?

To achieve the main goal specified in chapter subsection 1.3 (Goals), there is the necessity to do experimental research methods, by exploring quantitative variables and generate statistical data. By analyzing this data, it simplifies conclusion formulations and if possible, to make major or minor improvements. In this way, we can Test, Evaluate, Improve and Test again.

With this data, it is possible to conclude if it’s doable develop the purposed solutions, and to realize if it is really more efficient than human resource to perform search and rescue operations.

## 1.5 Contributions

The research made to write this dissertation has led to many contributions that consisted of curricular research demonstrations and event participations, and other side-activities. These side-activities were not directly related to this dissertation’s main goals of development, but the acquired knowledge during research made those activities realization possible. All these contributions were very important, useful and essential to this dissertation’s elaboration, because it was possible to enrich this research and to show third-parties what this team develops, and what solutions can bring to the community, societies and companies. The list below describes all curricular and non-curricular activities that helped this dissertation elaboration and third-parties with their needs.

- *Photogrammetry using an UAV-attached camera*. This outdoor activity consisted in photographing Mosteiro da Batalha's facade in Alcobaça, Portugal, with a camera attached on a UAV, user controlled. The R&D team was invited by DGPC, Portuguese cultural and patrimony department, and FAUL, one of the Portuguese architecture colleges.
- *Encontro Ciência 2016*. 2016 Science Meeting who took place at CCL, which is the Lisbon Congress Centre, from days July 4th to July 6th of the present year. This event consisted in showing this team's research work at the event's IT – Instituto de Telecomunicações stand.
- *European Researchers' Night 2016*. This event is a European Commission initiative which aims to raise citizens awareness for the importance of science, citizen quality of life and its impact on society development ("Noite Europeia dos Investigadores," n.d.). Took place at National Natural History and Science Museum, Lisbon on the 30th September of the present year. This event consisted in showing this team's research work at the event's IT stand.
- *ISCTE-IUL Carbon 0%*. This activity consisted on planting various trees on Montejunto's sierra. This initiative focused on environment sustainability subjects in reducing human ecological footprint. The R&D team designed several UAV flight tests to record plantation moments, considering closer and farther UAV positions in order to capture and record different view angles.
- Workshops on *How to build an UAV*. The R&D team gave three workshops on how to build a drone from scratch, using custom parts, explaining all component features and showing all steps necessary to make an UAV fly. These workshops took place on the dates and locations described below:
    - *ISCTE-IUL Academy Days*, from March 31st to April 1st. The academy days consisted on various workshops for high-school students from 10th to 12th grade and future college students;
    - *Electronic Laboratories week at Universidade da Beira Interior*. Took place at Universidade da Beira Interior (UBI) on June 30th of the present year. It was

supported by “Ciência Viva”, Portuguese agency for science and technology culture.

- *Drone’s Week at ISCTE-IUL*. This week consisted on various activities including presentations, speakers and workshops, all related with drones or UAVs. Took place at ISCTE-IUL from days July 18th to July 22nd of the present year and it was supported by “Ciência Viva”, Portuguese agency for science and technology culture.

- *UAV-g 2017 – International Conference on Unmanned Aerial Vehicles for Geomatics*: A paper is being written for submission on this conference that takes place in Bonn, Germany from September 4th to September 7th of 2017;
- *ICUAS'17 - The 2017 International Conference on Unmanned Aircraft Systems*: A paper is being written for submission on this conference that takes place in Miami Marriott Biscayne Bay, Miami, Florida, United States of America, from June 13th to June 16th of 2017;

All this events and activities gave an important step in this dissertation’s development, on how to improve work done with shared experiences between event and activity attendants.

## 1.6 Dissertation Structure

This subsection describes how this dissertation is structured. There are 7 chapters, and each one includes the following:

- Chapter 1 – Introduction: Present chapter;
- Chapter 2 – State of Art: Describes all research made before the development of this dissertation’s work;
- Chapter 3 – Unmanned Aerial Vehicle: Explains how to build an actual UAV from scratch explaining all parts and component features;
- Chapter 4 – Control and Communication: Describes how UAV’s control and communications are made between user and aircraft;
- Chapter 5 – Sense and Avoid Algorithm: Explains steps made to achieve main goals: develop an autonomous object collision avoidance system for UAVs;
- Chapter 6 – Simulation and Experimental Results: Shows results from different test environments and algorithm’s performance evaluation on goal achieving;

- Chapter 7 – Conclusion and Future Work: This dissertation conclusion and what can be made to improve work done.

To complement the chapters' contents, there is a set of "Annexes" where detailed information about some components or its features can be found. The diagram of blocks depicted on figure 1.1 describes the main aspects considered in this dissertation. The dashed arrows mean that an UAV can only output one of the parallel shape options, *i.e.*, GCS or Radio controlled, or UAV goes left, right or changes to Brake flight mode. The blue arrow pointing to "UAV" shape means that Raspberry Pi and Pixhawk composes the main components to control the UAV.

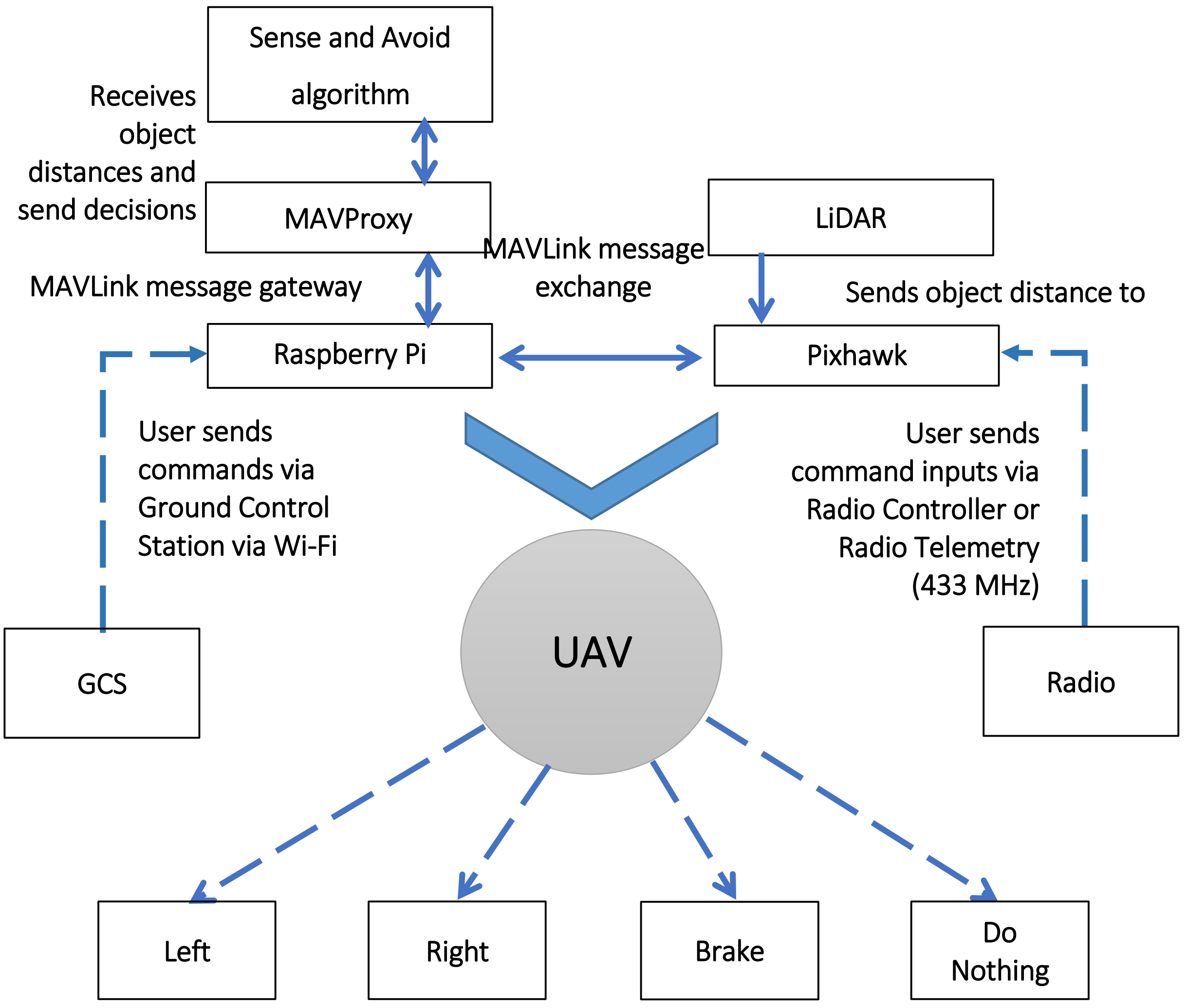

Sense and Avoid algorithm
Receives object distances and send decisions
MAVProxy
MAVLink message gateway
MAVLink message exchange
LiDAR
Sends object distance to
Raspberry Pi
Pixhawk
User sends commands via Ground Control Station via Wi-Fi
User sends command inputs via Radio Controller or Radio Telemetry (433 MHz)
GCS
UAV
Radio
Left
Right
Brake
Do Nothing

Figure 1.1 – Diagram of blocks showing the UAV actions, links and behaviors.

# Chapter 2
# STATE OF ART

This chapter includes all research done on early dissertation development stages, involving goal relative subjects, such as different UAV types, different types of communication, possible component technologies and differences between them.

The Unmanned Aerial Vehicles (UAVs) have a wide range of applications. Companies have developed hardware components in order to make an UAV fly with maximum stability and performance. Software projects were made to communicate with hardware components, in order to give a user-friendly interface, to read data and easily communicate with an UAV.

## 2.1 Unmanned Aerial Vehicles

Unmanned Aerial Vehicles (UAVs) are aircrafts controlled by a manned ground control station, which can send and receive commands between GCS and UAVs with the absence of a present pilot on an UAV.

### 2.1.1 UAV Types

There are various types of UAVs, and each one has its own operation processes, functionalities and purposes (Austin, 2011):

HALE (High Altitude Long Endurance) – These vehicles fly at high altitudes (15 km) and have great autonomy (24 hours of flight time). Their main purpose is to preform long-range (trans-global) military surveillance missions, reconnaissance and if well-armed, aerial attacks. They are controlled from fixed ground control stations;

MALE (Medium Altitude Long Endurance) – Similar to HALE, these vehicles operate at short ranges and fly at low altitudes (5 km – 15 km). They are used by military forces as well;

Tactical UAVs or TUAVs – Simpler and smaller than HALE and MALE, they operate at shorter ranges (100 km – 300 km). Used by the military from naval or ground bases.

Mini UAVs or MUAVs – Smaller than TUAVs, these light-weighted UAVs (20 kg of maximum weight) have even shorter ranges (30 km) and are now being used for some civilian purposes.

Micro UAVs – Also called as MAVs, these vehicles are smaller and lighter than any of above, and can perform actions that no other UAV can. Requires a slow flight with slow direction-changing movements. These UAVs are commonly used for civilian purposes, because they are affordable, easily controllable and can perform simple actions.

In this dissertation, we are going to focus on Micro Air Vehicles (MAVs).

### 2.1.2 UAV Structural Types

There are some different structural types of MAV, each one has its way of taking off, landing, aerodynamics and usability:

Fixed-wing: These UAVs have some benefits that doesn't exist in any other UAV types. They benefit from their designed two-winged aerodynamics. Their wings help the wind flow through the UAV. Because of that, they can achieve greater air speeds and glide without losing too much height. This leads to a much less power consumption and that allows much more autonomy for long range flights. Fixed-wing UAVs lift and land horizontally, and they can't hover in a certain position like other UAVs. Due to increased range, these structural type of UAV is most used by the military (Scherer, 2015).

Multi-rotor: Unlike fixed-wing UAVs, these multi-rotor UAVs are capable of hovering, standing still in the air. They do not have wings. Their behavior is similar to a helicopter, but with more propellers. The most common multi-rotor UAVs are: Quadcopters, Hexacopters and Octacopters. They lift off and land vertically. Compared with fixed-wing UAVs, they fly at lower speeds but have more action control. For being a multi-rotor UAVs, they consume much more power and have a shorter autonomy (Scherer, 2015).

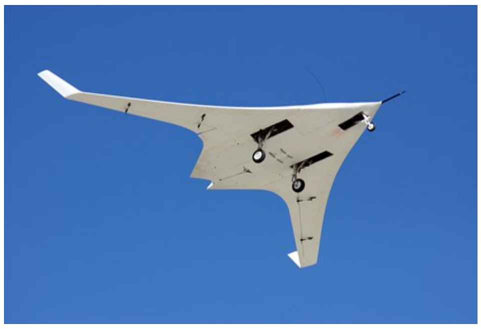


Figure 2.1– Fixed-wing UAV example (Wen, 2013).

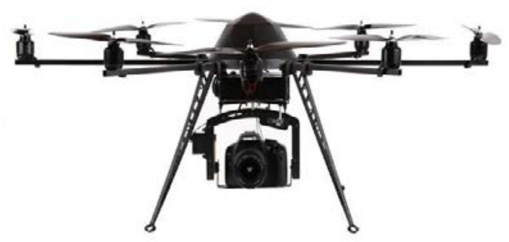


Figure 2.2 – Multi-rotor UAV example (Mortimer, 2012).

## 2.2 Companion Hardware

In order to control hardware components, read data and send commands, an UAV needs to be equipped with an on-flight board control. That component is described as a flight controller. To provide autonomous tasks, extended wireless communication features and on-board situation control, an UAV needs additional hardware component: a mini-computer. Both flight controller and mini-computer are connected internally through serial connection and they exchange data between them. These components are categorized as Companion Hardware.

### 2.2.1 Raspberry Pi

Raspberry Pi is a credit-card sized micro-computer, developed in 2012 at the University of Cambridge's Computer Laboratory, Runs Linux as operating system with the option to have a graphical user interface, and provide some useful inputs, such as USB ports, HDMI port, RJ45 port, audio port and GPIO (General Purpose Input / Output) connectors. These GPIO connectors allows developers to attach other hardware components, such as sensors, motors, leds and displays.

Raspberry Pi can be used for a variety of things, such as a personal computer, a web server, or a device controller where the setup steps are simple. The operating system is downloaded to an SD card, then user configures some minor variables, such as operating system language, keyboard layout and drive setup management. After that, additional packages, such as compilers and libraries, can be added (Brock, Bruce, & Cameron, 2013).

It’s a powerful computer, and when compared to desktop increased-sized solutions, it’s relatively cheap. This companion hardware it’s really useful for this dissertation because of its size, price and included functionalities (Saraiva, 2015).

### 2.2.2 Flight Controller

Some UAV flight controllers use hardware configurations based on existing projects and 3DR Pixhawk is one of them. ArduPilot is a full-featured open source project, licensed by GNU General Public License version 3 (GPLv3) that supports “from conventional airplanes, multirotors, and helicopters, to boats and even submarines” (Ardupilot Dev Team, n.d.-a). ArduPilot supports several types of UAV’s such as fixed wings, multi-rotors and UGVs (Unmanned Ground Vehicles). Each type is configured in a single firmware file, which is installed on the Pixhawk main processor. This flight control board has some embedded sensors to provide useful data to Ground Control Station (GCS): a gyroscope, an accelerometer, a barometer and a magnetometer. (Saraiva, 2015) It has the option to add a Global Position System (GPS) sensor to provide location data, such as latitude, longitude, altitude and speed. To perform guided missions, a GPS module is required in order to specify waypoints. This autopilot is fully programmable and can have First-Person View (FPV) camera gimbal support and control, Radio Controlled (RC) channel inputs and other sensors. The built-in hardware failsafe uses a separate circuit to transfer control from the RC system to the autopilot and back again. This prevents crashes by safely land on the ground  (Bin & Justice, 2009). In this dissertation, a 3DR Pixhawk flight controller was used with ArduPilot available firmware. In order to transfer data to Raspberry Pi, Pixhawk uses an USB connection.

## 2.3 Control and Communication

In order to control an UAV, it’s necessary to have intermediary components between UAV itself and piloting user. Such components are essential for piloting because they determine the behavior of an UAV.

### 2.3.1 Radio Controller

Piloting an UAVs can be quite a challenge for novice pilots, because an UAV can fly in various 3D space directions: left and right, to the front and back, up and down. The simplest and fastest way to start controlling an UAV is using a radio controller. A radio receiver on the UAV is needed to receive radio frequency (RF) data. This data is transmitted by a radio transmitter, controlled by the user, to control UAV’s attitude.

### 2.3.2 Ground Control Station

A Ground Control Station (GCS) is an essential UAV monitoring and controlling tool. It provides user relevant flight data, read by flight controller sensors. Can be used to track down UAV location by reading GPS data, perform autonomous tasks, calibrate sensors, pre-flight tests, and so on. It’s a required tool for ground operation tasks, used before, during and after UAV flights.

There are various Ground Control Stations software available for Windows and Linux operating system environments, such as Mission Planner, APM Planner and QGroundControl. In order to send and receive data between UAVs and GCS, both must have telemetry equipment, such as

Wi-Fi or Radio Frequency transmitters and receivers. This data is encrypted and sent by a common message type via MAVLink protocol.

### 2.3.3 MAVLink Message Protocol

MAVLink is a header-only message protocol that uses group of messages to transmit data between the UAV and GCS. It is designed to be reliable, fast and safe against transmission errors. It was first released in 2009 by Lorenz Meier under the LGPL license (Murilhas, 2015; Scherer, 2015). Each message is byte-encrypted with sensor related content, which can be interpreted by Ground Station Control or by Raspberry Pi, which will serve as a message intermediary between Pixhawk and GCS. Commands like take off, raise or decrease altitude (throttle increase or decrease, respectively) are sent by GCS and interpreted by Pixhawk to perform desired action.

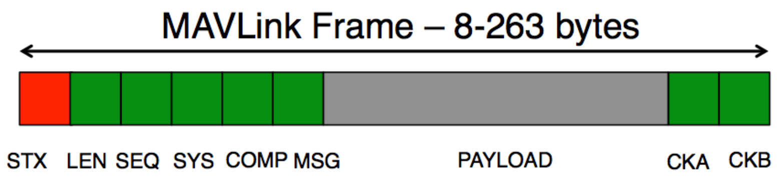


Figure 2.3 – MAVLink Message Structure (“MAVLink Micro Air Vehicle Communication Protocol - QGroundControl GCS,” n.d.).

## 2.4 Distance Sensing

Distance sensing can be useful for many UAV applications, such as ground altitude measurement, UAV terrain shape following, object detection and environment mapping. The use of distance sensors on an UAV extends its features for user determined purposes. There are various types of distance sensors, and each one has its features, restrictions and limitations. The sections 2.4.1 to 2.4.4 describe researched distance sensors, their applications and differences.

### 2.4.1 Light Detection and Ranging (LiDAR)

A Light Detection and Ranging (LiDAR) sensor is a measurement technology, which is based on a precise measurement of the time delay between the transmission of a pulsed optical laser light signal and its reception (Duh, n.d.). This allows UAV to produce environment mapping, by analyzing objects that are close or far from UAV current position. This sensor produces two types of light signals: Reference signal fed from the transmitter and a received signal reflected from the target. The time delay between these two signals is calculated through a signal processing method, known as correlation. These correlation process accurately calculates the time delay, which is translated into distance based on the speed-of-light constant ("LIDAR - Technology and System Hardware Overview," n.d.). Each LiDAR for each application works at different electromagnetic wavelengths. Meteorology LiDARs usually work on ultraviolet (250 nm) and infrared (1500 nm to 2000 nm) wavelengths. For terrestrial mapping, at near-infrared wavelengths (Duh, n.d.). The figure 2.4 shows an example of the electromagnetic spectrum and where terrestrial LiDARs wavelengths are located.

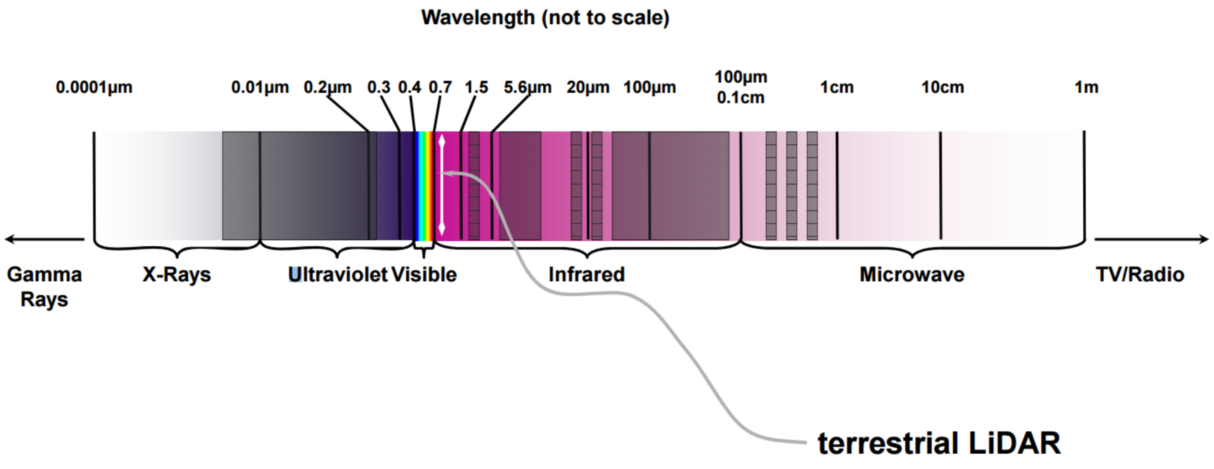


Figure 2.4 – Electromagnetic spectrum and LiDAR wavelength comparison (Duh, n.d.).

For pulsed LiDAR lasers, to obtain object distances and range resolutions, the equations are (2.1) and (2.2) are used:

$$R = c\frac{t}{2} \tag{2.1}$$

where $R$ means the distance, $c$ the speed of light value (~299,792,458 m/s) and $t$ the time delay between transmitting and receiving the light pulse (in ns), and

$$\Delta R = c\frac{\Delta t}{2} \tag{2.2}$$

where $\Delta R$ is the range resolution, $c$ the speed of light and $\Delta t$ resolution of time measurement (in ns) (Duh, n.d.).

Range resolution is not the same thing as accuracy. Accuracy is the difference of measured distances between values read by LiDAR and real and true distance values. For example, if

LiDAR measures a distance of 1,01 m from an object or surface and the real distance value is 1 m, the LiDAR reading accuracy is 1 cm (Value read - Real value = LiDAR accuracy). Range resolution is, for example, on environment mapping, a LiDAR sensor can generate a graph called "point cloud". And each point means a read distance. Range resolution is the difference between points on a point cloud graph and for a terrain elevation model, "it's the accuracy of the elevation", or by other words, how far is the model measurements are from the real terrain elevation (Schmid, Hadley, & Wijekoon, 2011).

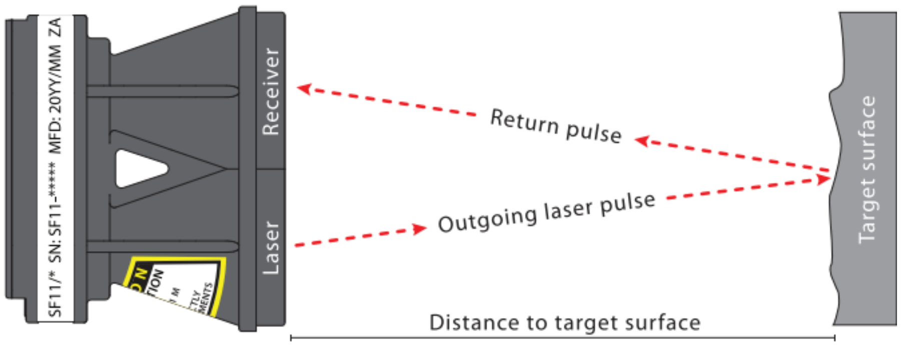

SF11/* SN: SF11-***** MFD: 20YY/MM ZA
Receiver
Laser
Return pulse
Outgoing laser pulse
Target surface
Distance to target surface

Figure 2.5 – LIDAR explanation (Lightware, 2016).

### 2.4.2 Sound Navigation and Ranging (SoNAR)

SoNAR is a measure technology based on sound waves. This sensor's principle is the measurement of "bouncing acoustic waves" that hit objects, and by measure its time delay between sound transmission and echo reception of those acoustic waves, it's possible to determine object distance. A SoNAR has various applications, but it's commonly used on ships "to measure the depths" of water surfaces, animals or objects (Rouse, 2011). A SoNAR sensor is composed by "an acoustic wave pulse generator, a transducer for transmitting acoustic waves in narrow beams, an acoustic pickup, a set of amplifiers and a delay timer" (Rouse, 2011). Sound waves propagation speed varies according to the environment. A SoNAR sensor can be more useful on water environments other than outdoor environments, because sound waves propagation speed on water is faster than outdoor environment. The distance from an object is measured by sound waves echo time delay. The figure 2.6 explains how a SoNAR sensor works.

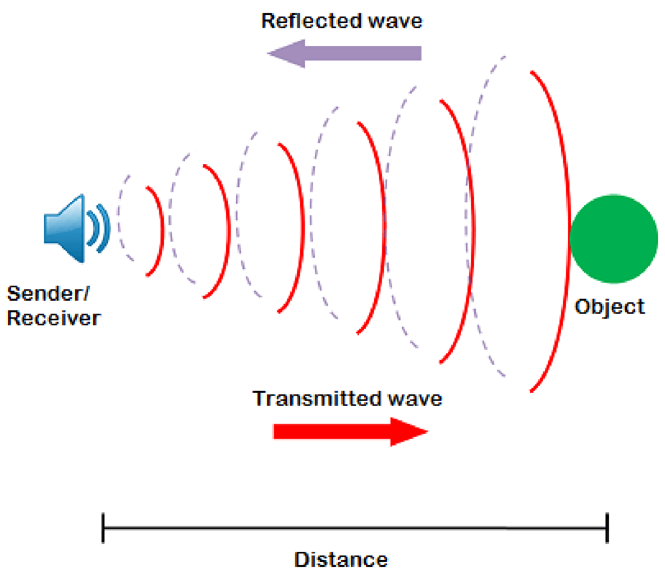


Figure 2.6 – SoNAR principle (Radio-Electronics, 2014).

SoNAR sensors are commonly used in fishing ships in order to locate fish shoals. It is also used on submarines to detect objects and other vessels. On medicine field, SoNAR sensors are used to detect diseases or to monitor unborn child. Radio waves are reflected differently according to different body parts or organs (Scholastic Inc, n.d.).

### 2.4.3 Radio Detection and Ranging (RaDAR)

RaDAR is another measurement technology based on radio waves. It has the same principle of and LiDAR SoNAR sensors, which consists in detecting objects based on the reflected emitted signals. RaDAR sensors have an antenna, a transmitter and a receiver. The transmitter generates the radio wave. When that wave hits an object, it “bounces” back. The reflected signal is detected and classified as radio echo by the antenna. Then, the receiver amplifies that signal to increase its strength. The radio waves travels at speed of light. RaDAR sensors are used for both military and civilian purposes. Commonly, it’s used in planes and ships in order to locate other ships or aircrafts and to prevent collisions. RaDAR sensors are also used for meteorology purposes, such as weather predictions. And when combined with LiDAR sensors, this hybrid solution can help studying storm directions and hurricane locations. A Doppler RaDAR is used to determine storm directions, and it’s “based on the principle of the Doppler effect”. The figure 2.7 explains the Doppler Effect on reflected radio waves.

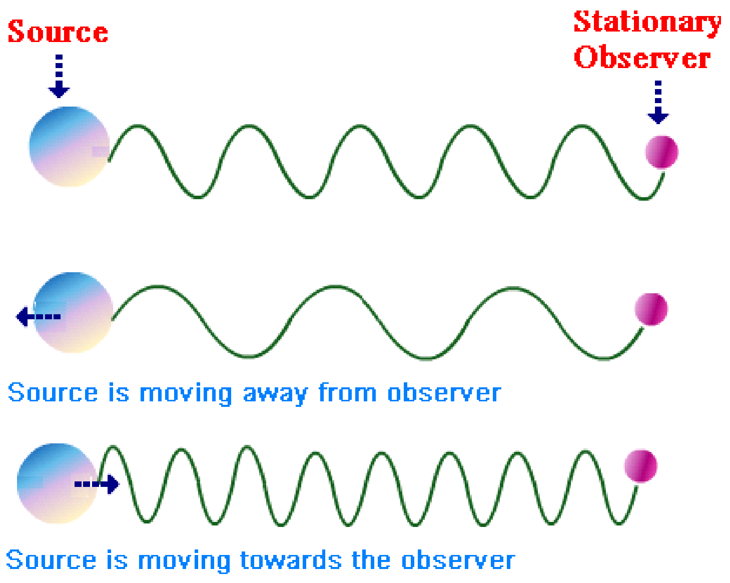

Source
Stationary Observer
Source is moving away from observer
Source is moving towards the observer

Figure 2.7 – Doppler effect (NCS Pearson, 2016).

The Doppler Effect explains that the "frequency of a wave changes as the source of the wave moves toward or away from the receiver". This means that reflected radio waves with different frequencies can determine the source or stationary observer direction by analyzing reflected waves. The Doppler RaDAR sensors are also used by police cars to determine the car speeds. (Scholastic Inc, n.d.).

### 2.4.4 Distance sensors comparison

The three researched distance sensing technologies have its own features, restrictions and limitations. Choosing the right sensor depends on its purpose of measurement. Some known companies use different distance measurement sensors in order to accomplish same goals: object sensing and avoiding. Google company and its self-driving car uses a LiDAR sensor as primary sensor to "see" the area around the car for autonomous driving. Google's current solution on object-aware cars can sense objects from 30 m to 200 m (CleanTechnica, 2016). Tesla company uses RaDAR sensors on its autopilot featured cars, once this, sensors can detect objects on harsh environments such as snow, rain and dust while a LiDAR could have some problems in these conditions (Davies, 2016). The Annex D explains differences between these three sensors. In this dissertation, the use of a LiDAR sensor was chosen for having a long distance measurement, a light weight and an affordable price. In order to perform SAR operations, an UAV must be capable of "seeing" objects with high accuracy. Choosing a LiDAR sensor was the solution found to meet the requirements of this dissertation's main goals.

# Chapter 3 UNMANNED AERIAL VEHICLE

In this chapter, all necessary components description for a successful UAV building are present. Common parts and electrical components explained, what role they have on an UAV, alongside with technical explanations and pre-flight calibrations.

## 3.1 UAV Composition – Common Parts

In order to perform this dissertation experimental tests, there was a need to build a custom multirotor-typed UAV. The subsections 3.1.1 to 3.1.5 define which materials and components are commonly used to build an UAV with described parts for an initial and base configuration.

### 3.1.1 Frame

For every custom built UAV, this part must be the chosen first (Benson, 2015). There are various types of multirotor frames from bicopter (two rotor UAV) to octacopter (eight rotor UAV) (O. Liang, 2004). The frame is the UAVs chassis that is used to support all payload installed in an UAV. It's composed by the "arms, landing gears, rotor mounts and central plates" (Saraiva, 2015). The frame choice is an important one because each frame type has different rotor numbers. A tricopter frame appropriated for three-rotor UAV and an octacopter for eight-rotor UAV. More rotors mean a "higher energy consumption but also leads to a higher flight stability" (Saraiva, 2015). Frames can also have different sizes. The frame size is measured by its diameter, which is normally comprehended between 350mm and 700mm (Benson, 2015).

There are two multirotor frame configurations: flat and coaxial. Flat configuration means that every arm can only support one rotor. If the frame has four arms, its meant for a quadcopter, which is a four rotor UAV. Coaxial configuration means that every arm can have a secondary rotor, above the initial one. A four arm frame in a coaxial configuration can support two rotors: the first one installed like flat frame configuration, and the secondary installed below primary rotor. For example, in a coaxial frame configuration, a three arm frame makes a hexacopter and a four arm frame makes an octacopter. For quadcopters, hexacopters and octacopters, there are two more possible frame arm configurations: 'X' and '+'. This is determined what style is desired by choosing how frame center-to-front direction can be presented. In this dissertation, a flat configuration and 'X' quadcopter-typed UAV was used (O. Liang, 2004).

The figure 3.1 represents the difference between 'X' and '+' configurations:

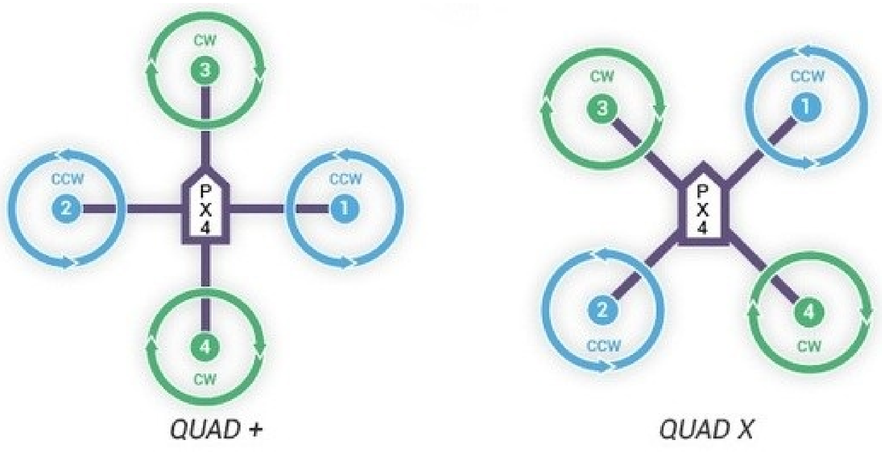


Figure 3.1 – Differences between 'X' and '+' configurations (Ardupilot Dev Team, 2014a).

### 3.1.2 Rotors

The main function of rotors is to lift the "frame off the ground, hover" and by slightly adjusting angular velocity of rotors, an UAV can fly on every space direction ("How to Build a Drone - A Definitive Guide For Newbies," n.d.; Luukkonen, 2011). A rotor is composed by its casing, shaft, which sizes can vary. Rotors can have different types, which can be brushed or brushless. The figure 3.2 shows the style difference between rotor types.

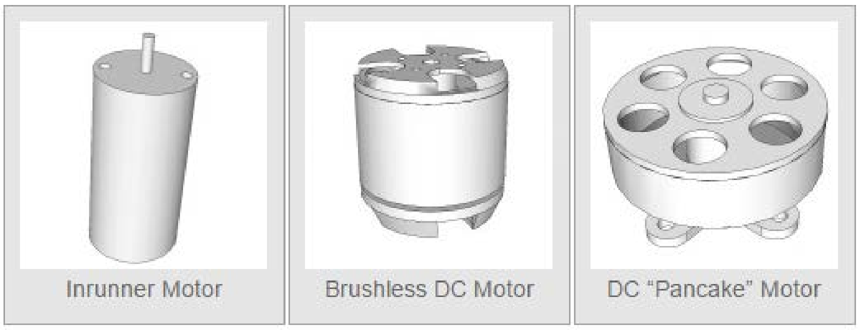


Figure 3.2 – Different types of rotors (Benson, 2015).

Brushed rotors "spin the coil inside a case with fixed magnets mounted around the outside of the casing". Brushless rotors work the other way around: "the coils are fixed either to the outer casing or inside the casing while the magnets are spun" (Benson, 2015). In the UAV market, it's commonly used brushless DC motors.

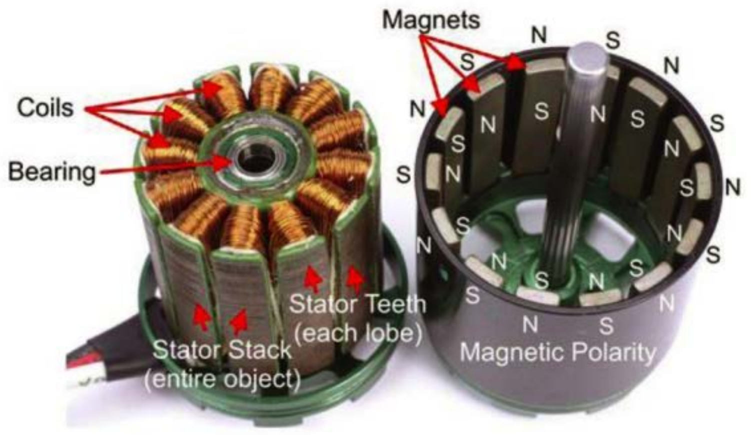


Figure 3.3 – Brushless DC motor (Kwan, 2016).

These types of rotors can have different sizes, measured by rotor height and diameter. Rotation clockwise or counter-clockwise, rotation speeds also vary between sizes, and are measured by KV rating. The KV value of a rotor specify “how fast it will rotate for a given voltage” (Benson, 2015). Rotor speeds are measured in rotations per minute (or rpm). Low KV rotor values (comprehended between 500 and 1000) are used to increase UAV in-air stability. High KV rotor values (1000 and 2500) are used for racer and acrobatic UAVs. For most multirotor UAVs, low KV rotor values are preferred. If a rotor has a KV value of 800 (or 800 rpm per volt), for a given input voltage of 11.1V, it will spin at: $800 \times 11{,}1 = 8880$ rpm (Benson, 2015).

The rotors are mounted on every arm end of the frame, with rotor shaft pointed up. For a coaxial frame configuration, the secondary rotor is mounted with rotors’ shaft pointed down. There are advantages and disadvantages about using a coaxial configuration. It can save space, make frame foldable or even land safely a rotor fails, but has a 10 to 20% efficiency loss because the lower rotor is “just turning in already sped-up air” (O. Liang, 2004).

### 3.1.3 Propellers

The propellers are the top-end part of all rotors. In order to lift an UAV, propellers create downwards air flow allowing enough strength to take-off. To choose the right propellers, it’s important to check their diameter size and pitch values. For most UAV configurations, propellers with two blades it’s commonly used. Smaller the propeller, smaller the inertia created, which can be adequate for acrobatic flights because it’s “easier to slow down and speed up” the blades. On the contrary, bigger propellers are used for a more stable flight because takes more time to quick speed-ups and slow-downs. (Benson, 2015; "How to Build a Drone - A Definitive Guide For Newbies," n.d.)

There are two types of propellers, clockwise and counter-clockwise, which design varies according to rotation direction. Clockwise propellers must be installed on clockwise rotation rotors and vice-versa for counter-clockwise propellers. This verification is very important because a mistaken installation of propellers can lead to an instant flip at UAV's take-off. For example, a counter-clockwise propeller mounted on a clockwise rotor generates upward air flow, *i.e.*, instead of pushing air downwards, it pushes air upwards.

In order to achieve thrust, propellers must have a certain pitch value to push air downwards at a certain rotor speed. The pitch value is related to a propeller's "angle of attack", which means the angle value a propeller "attacks" the air. A higher pitch value means a higher angle between the blades and propellers diameter center, a therefore a higher angle of attack (Benson, 2015) (Bristeau, Martin, Salaün, & Petit, 2009). The figure 3.4 represents a comparison between a high and low pitch propeller and how can it can make a difference when taking-off an UAV.

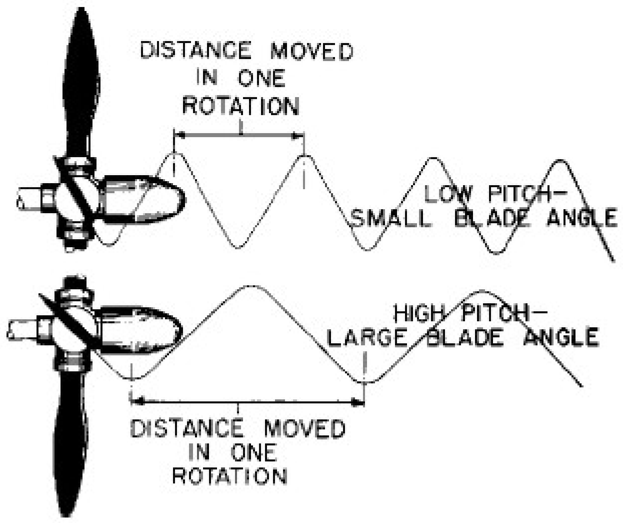

DISTANCE MOVED IN ONE ROTATION
LOW PITCH— SMALL BLADE ANGLE
HIGH PITCH— LARGE BLADE ANGLE
DISTANCE MOVED IN ONE ROTATION

Figure 3.4 – High and low pitch values propellers differences according to distance moved ("Constant Speed Propellers," n.d.).

### 3.1.4 Electronic Speed Controllers

An Electronic Speed Controller (ESC) is the bridge between rotors and flight controller. Controls rotor speed and direction. An ESC is known for transforming electric current from battery that outputs Direct Current (DC) to Alternating Current (AC). The UAV's rotors only accept AC type of current. The signal is pulse-modulated, described as Pulse-Width Modulation (PWM) which means that "the pulse has always the same maximum and minimum amplitude and the duty cycle is the only variation" (Murilhas, 2015). An ESC has maximum current and tension values, so the right ESC must be chosen accordingly with rotor's maximum current draw and voltage input. For example, if rotor's specification has a maximum current draw at top speed of 24 amperes (A), the ESC attached to this rotor must have a maximum current draw of 24A or above. It is recommended that ESCs always have a threshold between rotors maximum current draw of rotors and ESC itself (Marinescu, 2013). This means that for the same rotor, a 30A ESC should the right choice to make. Connected and controlled by flight controller, it determines how much current should provide to a single rotor, and therefore, controlling rotor's rotation speed. A single ESC controls a single rotor. The connections made between ESCs outputs and rotors inputs controls rotor's rotation direction (Benson, 2015) (Saraiva, 2015). Some ESCs have a Battery Eliminator Circuit (BEC) component to convert input voltage to 5V which can power a RC receiver in order to, for example, test the RC input and Rotor/ESC response.

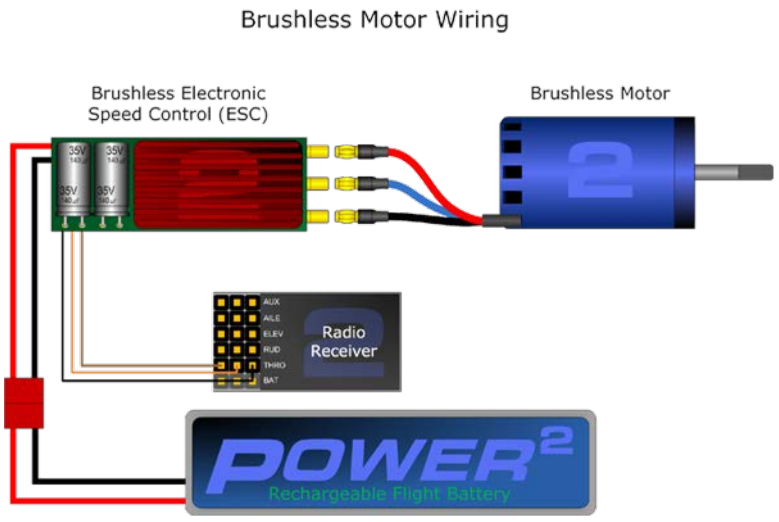


Figure 3.5 – ESC wiring example ("Single Engine ESC Wiring | 2BFly," n.d.).

### 3.1.5 Power Supply

In order to fly an UAV, an energy source is needed. For most UAV uses, lithium polymer batteries are used (LiPo batteries), and they are known for their high power capacities and low pack weights. These LiPo batteries are composed by cells. Each cell has a nominal voltage value of 3.7V. And these cells are normally connected in series, but they can also be connected in parallel to achieve a desired power output (Patel, Patel, Faldu, & Dave, 2013) (Schneider, 2012). For example, if a battery has three cells connected in series, it's called a 3S LiPo battery. LiPo batteries are categorized by its power capacity and C-rating. Power capacity is measured in miliamperes per hour (mAh). Higher power capacity value means longer flights, but also an increase in weight payload values. And increased weight values means more to lift, resulting in an increased power consumption, affecting flight times at the same time (G. C. T. Liang, 2015) (Marinescu, 2013). C-rating is the battery discharge level and measures how quick the battery can output power. If a 5800mAh battery has a discharge rate of 25C, which means 25 times battery capacity (C), the battery can output up to a maximum current draw of 145A ($25 \times 5800 = 145$). LiPo batteries have also a burst current C-rating. Measuring the rotors current draw at mid throttle and full throttle it's important to verify if a certain battery is appropriated for the UAV (Marinescu, 2013) (Patel et al., 2013). To measure a LiPo battery, a cell voltage meter is needed. A fully charged battery means all cells must measure 4.2V. A discharged battery measures approximately 3.74V-3.75V (Salt, n.d.).

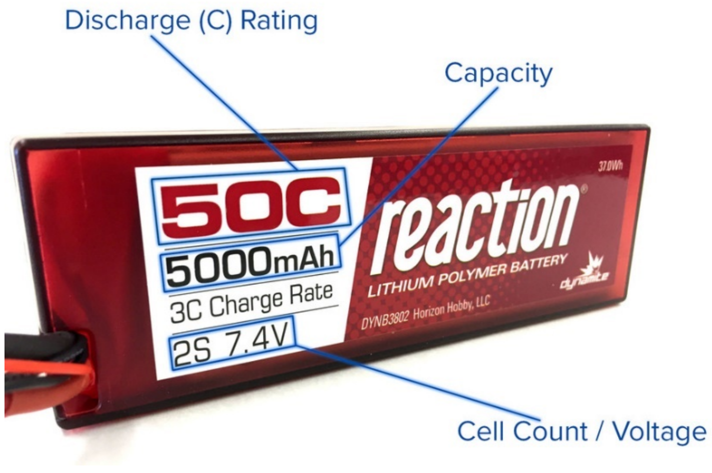


Figure 3.6 – LiPo battery example (Schneider, 2012).

To ensure power to all ESCs and rotors, there is the need to distribute battery power. In order to accomplish that, a Power Distribution Board is needed (PDB). Some frame plates have an embedded PDB. The PDB example figure 3.7 demonstrates how to connect ESCs and other components.

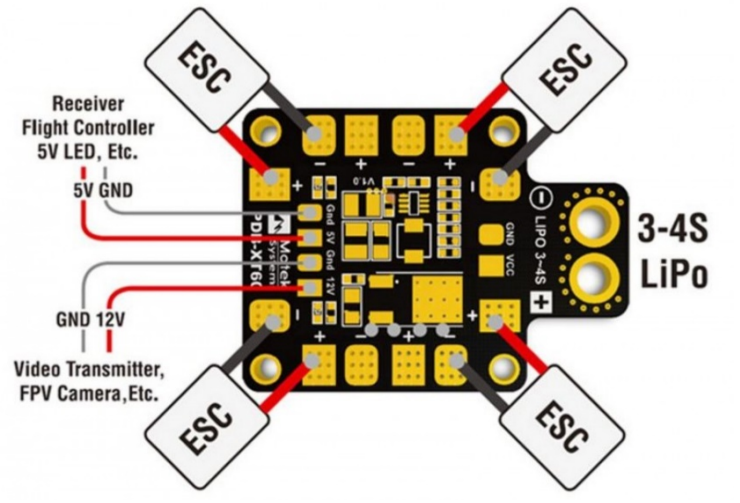


Figure 3.7 – PDB example (BuzzHobbies, 2016).

### 3.1.6 Set-up Model

In this dissertation, a quadcopter-typed UAV was used, with a ‘X’ mounted orientation and hexacopter plates (with embedded PDB) for an increased middle frame space. This personalized configuration allows extra components addition. The common parts used were:

- 2 unit. of Hexacopter frame plates;
- 4 unit. of Low-KV rotors (350KV);
- 4 unit. of ESCs with 20A of maximum current draw;
- 4 unit. of 1345 Carbon fiber propellers;
- 1 unit. of Battery 6S with a capacity of 10000mAh.

## 3.2 UAV Composition – Flight Controller

The flight controller is responsible for mostly all UAV actions and behaviors. In this dissertation, a 3DR Pixhawk was used. Pixhawk is an "independent, open-hardware project" to provide UAV autopilot features commonly used for civil, industrial and military purposes ("Pixhawk Autopilot - Pixhawk Flight Controller Hardware Project," 2016). Running an internal real-time operating system called NuttX (BYOD, 2015), Pixhawk is known to be an improved and updated version of ArduPilot Mega (APM) (Saraiva, 2015) and it has the perfect conditions to meet this dissertation goals. The table 3.1 shows some Pixhawk hardware specifications:

| | |
|---|---|
| **CPU** | 168 MHz Cortex |
| **RAM** | 256 KB |
| **Flash memory** | 2 MB |
| **Sensors** | 3D Accelerometer, Gyroscope, Barometer, Magnetometer |

Table 3.2.1 – Pixhawk hardware specifications (BYOD, 2015).

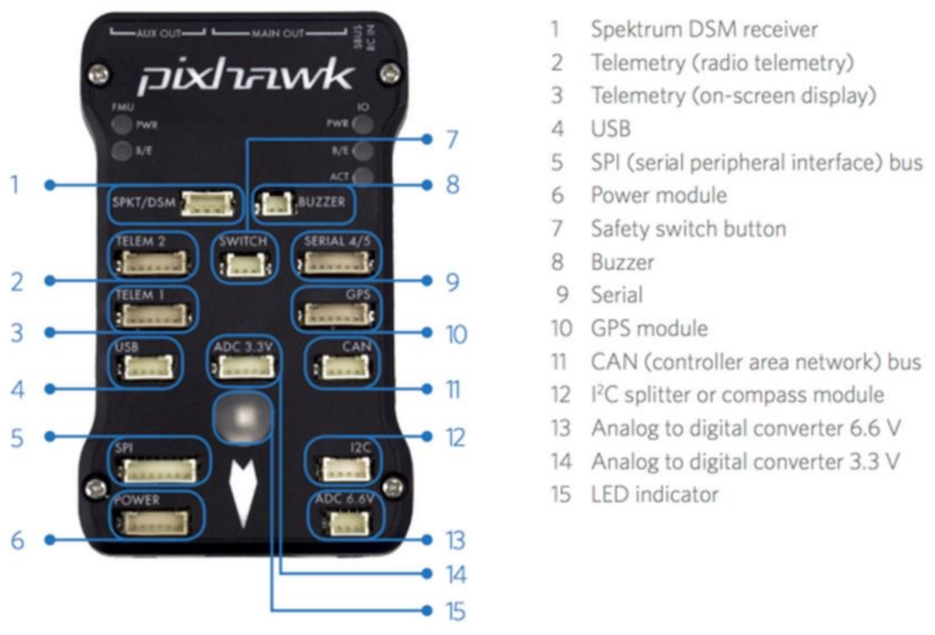

AUX OUT
MAIN OUT
SBUS
RC IN
pixhawk
FMU
PWR
B/E
IO
PWR
B/E
ACT
SPKT/DSM
BUZZER
TELEM 2
SWITCH
SERIAL 4/5
TELEM 1
GPS
USB
ADC 3.3V
CAN
SPI
I2C
POWER
ADC 6.6V
1 Spektrum DSM receiver
2 Telemetry (radio telemetry)
3 Telemetry (on-screen display)
4 USB
5 SPI (serial peripheral interface) bus
6 Power module
7 Safety switch button
8 Buzzer
9 Serial
10 GPS module
11 CAN (controller area network) bus
12 I²C splitter or compass module
13 Analog to digital converter 6.6 V
14 Analog to digital converter 3.3 V
15 LED indicator

Figure 3.8 – Pixhawk input options ("Unmanned Hawk Autopilot Kit," n.d.).

### 3.2.1 Firmware

Every flight controller has an internal firmware. This firmware is responsible for UAV control and to provide user relevant data read from internal and external sensors. Each firmware is configured accordingly to UAV type. A quadcopter configured firmware is different from a fixed-wing UAV. The firmware used was ArduCopter, in its 3.3.3 version for Pixhawk. It's an open-source firmware that allows UAVs to preform autonomous tasks and missions, real time data and control, UAV altitude hold, hovering stabilization and other features. This firmware, totally pre-configured for quadcopters, requiring no programming at all, includes various flight modes to perform certain tasks or to achieve UAV desired behavior (Saraiva, 2015). The Annex A is related with available and experimental tested flight modes.

### 3.2.2 Attitude

An UAV can fly in every direction by controlling 3 different axes: Longitudinal, lateral and vertical axis. Vertical axis movements represent upwards and downwards UAV's movement, called Throttle movement. Left or right movements are represented by UAV's rotation on longitudinal axis, also called Roll movement. And lateral rotations axis represents Pitch movements, or UAV's dislocation to the front or to the back. Yaw movements means UAV rotations on vertical axis, and it can be clockwise or counter clock-wise rotations. Every change on Roll, Yaw, Pitch or Throttle can be also called as an Attitude change. The figure 3.9 describes visually what axis are controlled by Roll, Pitch and Yaw movements (Kwan, 2016).

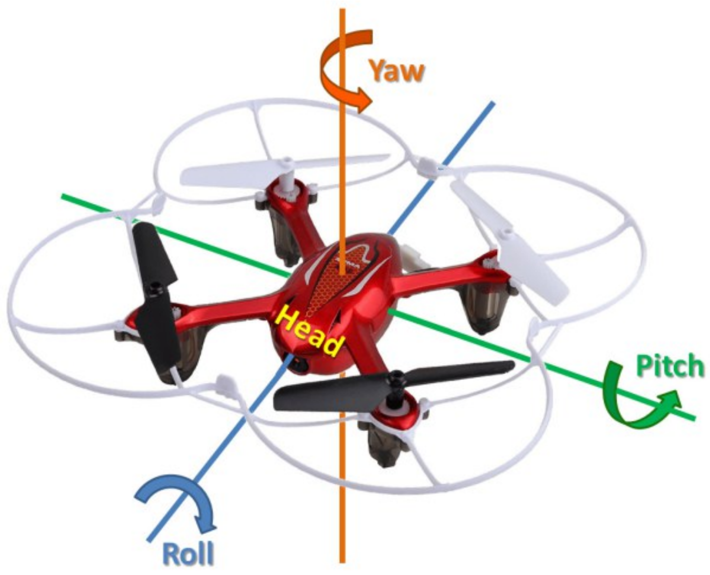


Figure 3.9 – UAV's possible 3D space movements ("Drone Controls," n.d.).

### 3.2.3 Internal Sensors

Pixhawk’s flight controller has internal sensors in order to maintain a self-stabilizing flight. These internal sensors are within a small electronic board called Inertia Measurement Unit (IMU). Pixhawk’s IMU board has a 3-axis gyroscope, which measures UAV’s angle of rotation and a 3-axis accelerometer, measuring linear accelerations. The IMU board is responsible for keeping an UAV stable in the air, by reading its sensors data and sending ESCs signals to change rotors speeds, increasing or decreasing, in order to achieve a horizontal balance on longitudinal and lateral axis (Benson, 2015) (Kwan, 2016). These sensors need calibration in order to set default IMU parameters when an UAV is standing still on the ground before flying for the first time.

For maintaining current UAV altitude, a barometer is needed in order to analyze changes on atmospheric pressure, since it changes when UAVs altitude also changes above sea level (Benson, 2015). The combination of the IMU and barometer readings can assure a self-stabilizing hover flight on non-windy environments.

To calculate UAV’s current heading, a compass or magnetometer is used. By using earths’ magnetic field, a compass can determine which direction is UAV’s heading.

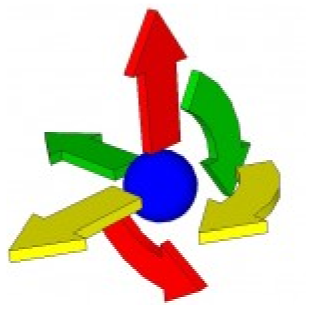


Figure 3.10 – IMU possible readings on UAV’s behavior changes (Benson, 2015).

### 3.2.4 External Sensors

External sensors add Pixhawk’s flight controller useful features in order to improve UAV’s flights and to achieve certain goals. These sensors are not embedded within Pixhawk’s main board and therefore connections have to be made.

#### 3.2.4.1 Global Positioning System

In order to perform autonomous tasks, such as waypoint-guided missions or to save home location, a GPS sensor is needed to give accurate reading about UAV’s current position in 3D space. With GPS, the user can track down UAV’s current position in real time. Also, it gives a more accurate reading about UAV’s current altitude above sea level, and on windy environments, GPS readings improve UAV’s current position and altitude holding. GPS sensor it’s indispensable for outdoor UAV flights, since failsafe mechanisms and other autonomous tasks relies almost completely on GPS readings.

#### 3.2.4.2 Light Detection and Ranging

A Light Detection and Ranging sensor, or a LiDAR sensor is commonly used for environment mapping and altitude precise measurements. It’s useful for an UAV to maintain a predefined above ground altitude in order to perform precise hoverings or automated UAV landings (Lightware, 2016). In this dissertation, a laser altimetry LiDAR was used. There are various LiDAR altimeters that are compatible with Pixhawk’s flight controller. LIDAR-Lite v2 and SF11/C LiDAR, produced by PulsedLight and Lightware respectively, are good choices for UAV distance measure purposes. In Annex C, table C.1 represents a comparison between LIDAR-Lite and SF11/C LiDAR, by stating full specifications of both LiDAR altimeters.

For this dissertation, it was used SF11/C LIDAR altimeter for future work usages. For having higher distance measurement than LIDAR-Lite, SF11/C LIDAR can also be used for other applications that require a high distance measurement value, such as environment mapping. Its maximum distance measurement guaranteed a safe distance when “reading” and locating objects in sight. This LiDAR sensor was the chosen solution to fly an UAV outdoors and to perform search and rescue operations.

Pixhawk’s flight controller is prepared to read LiDAR data in order to perform tasks described above if this sensor is mounted down-pointed below UAV’s frame’s infra-plate structure. In this dissertation, the sensor was placed on UAV’s upper-frame plate and is front-faced, matching UAV’s front (also called “nose”) heading. Pixhawk’s above ground altitude holding features using LiDAR sensors were disabled to prevent unwanted UAV behaviors.

### 3.2.5 Calibrations

In order to fly an UAV for the first time, some pre-flight calibrations must be made. The most important sensors for calibration are the accelerometer and compass / magnetometer. These internal sensors (compass can be an external sensor if GPS is used), as said before, are the main reason for UAV's on-air stabilization. In order to calibrate these sensors, a Ground Control Station is needed to enter calibration options. Radio transmitter calibration is also needed. The Mission Planner GCS software was used to perform pre-flight calibrations mentioned above. Each calibration option links to a sensor (except for RC calibration). The subsections 3.2.5.1 to 3.2.5.3 describes what steps should be made to perform a successful sensor and RC calibration.

#### 3.2.5.1 Accelerometer Calibration

In order to calibrate UAV's accelerometer in Mission Planner GCS, the user must go to "Initial Setup" > "Mandatory Hardware", and then select "Accel Calibration" from the menu on the left and click on "Calibrate Accel". Then a step-by step window pops up, and by following the instructions, user must grab the UAV and perform this steps, one by one, as requested (Ardupilot Dev Team, n.d.-b):

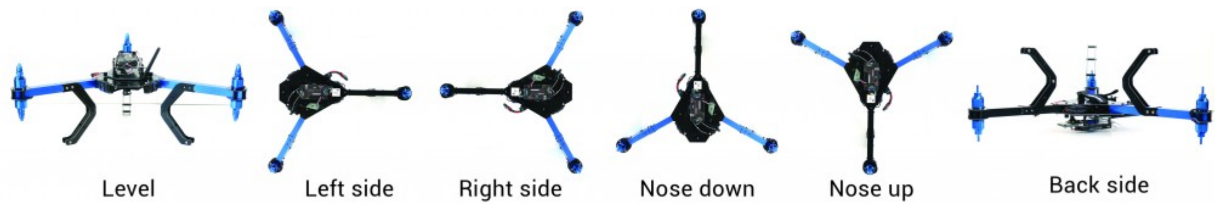


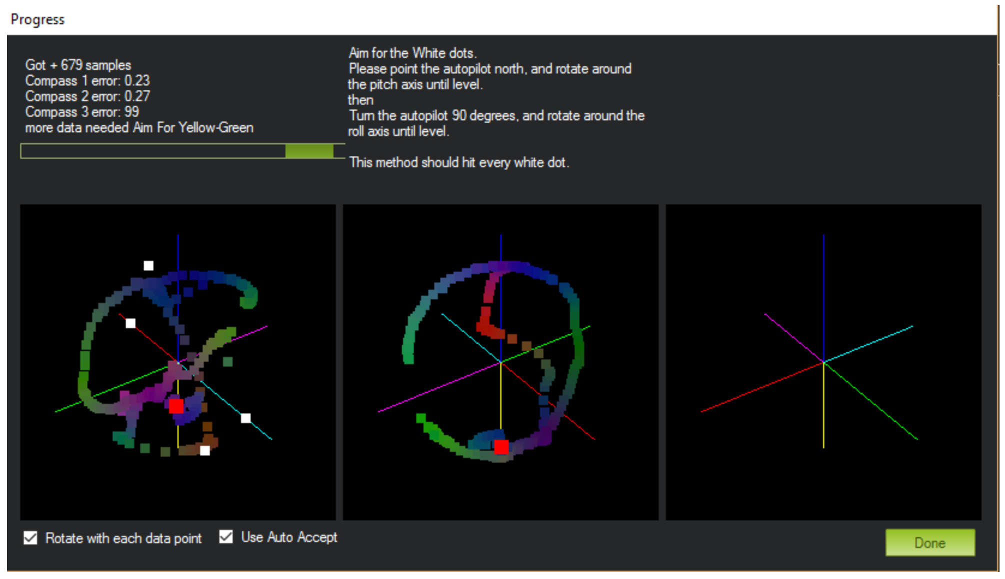
Figure 3.11 – All UAV positions required for successful calibration (Ardupilot Dev Team, n.d.-b).

When all steps described on the figure 3.11 were made, a successful calibration message should appear at the end of the process.

#### 3.2.5.2 Compass Calibration

Compass calibration can also be made on Mission Planner GCS. Under "Initial Setup" > "Mandatory Hardware", there is an option called "Compass". By selecting UAV's flight controller pre-defined options (which in this case, it's Pixhawk) and by checking "Obtain declination automatically" and "Automatically learn offsets" checkboxes, the user must proceed to calibration by clicking in "Live Calibration" button (Ardupilot Dev Team, n.d.-c).

The image 3.12 shows the next pop-up window that should appear right after clicking on "Live Calibration" button.


Progress

Got + 679 samples
Compass 1 error: 0.23
Compass 2 error: 0.27
Compass 3 error: 99
more data needed Aim For Yellow-Green

Aim for the White dots.
Please point the autopilot north, and rotate around the pitch axis until level.
then
Turn the autopilot 90 degrees, and rotate around the roll axis until level.

This method should hit every white dot.

Figure 3.12 – Mission Planner's compass "Live Calibration" window (Ardupilot Dev Team, n.d.-c).

On this window, the user needs to perform the exactly same steps as mentioned on Figure 3.11, *i.e.*, rotating on every UAVs axis and angles, to acquire calibration samples. When enough samples were acquired, the compass calibration ends and save its preferences.

#### 3.2.5.3 RC Controller Calibration

Radio controller calibration is also an important step before flying an UAV for the first time. This calibration sets maximum and minimum limits of RC channels for user's current RC transmitter. To perform RC calibration, also on Mission Planner GCS, the user must go to "Initial Setup" > "Mandatory Hardware" and then select "Radio Calibration". If radio transmitter and receiver were bond, green bars should move when the user also move RC transmitter sticks. In order to start calibration, the user must click on "Calibrate Radio" button. Now, the user must move all sticks to its maximum and minimum position, for every UAV possible behaviors (Yaw, Pitch, Roll, Throttle) and change switches to high and low positions of active additional channels. When all of the steps mentioned before were made, the user must click again on "Calibrate Radio" in order to save maximum and minimum RC transmitter limits for a determined UAV.

## 3.3 UAV Composition – Raspberry Pi

Raspberry Pi is the intermediary companion hardware between user's command inputs from a Ground Control Station and Pixhawk's flight controller. It is running Raspberry Pi's Raspbian operating system version. Raspbian is a Debian-based Linux operation system, fully optimized for Raspberry Pi's hardware, developed by Raspberry Pi Foundation. It has some pre-installed tools such as Java SE Platform, which is useful to run this dissertation "Sense and Avoid" algorithm. (Raspberry Pi Foundation, 2012a) The chapter 5 of this dissertation explains what features "Sense and Avoid" algorithm has and how it will behave when facing objects in UAVs current trajectory path. A Raspberry Pi 2 Model B was used and it's mounted on UAV's upper frame plate. This Raspberry Pi's version has 1 GB of RAM, 900 MHz quad-core CPU processor and 4 USB ports (Raspberry Pi Foundation, 2012b). Some USB ports were used to connect Pixhawk, to exchange MAVLink data between background algorithm and Pixhawk, and a Wi-Fi dongle to connect to a GCS. It requires 5V powered by a mini-USB cable. This credit-card sized computer was chosen to perform this dissertations' main goals because of its size, performance, big hobbyist internet community and for being a low-cost solution (Murilhas, 2015).

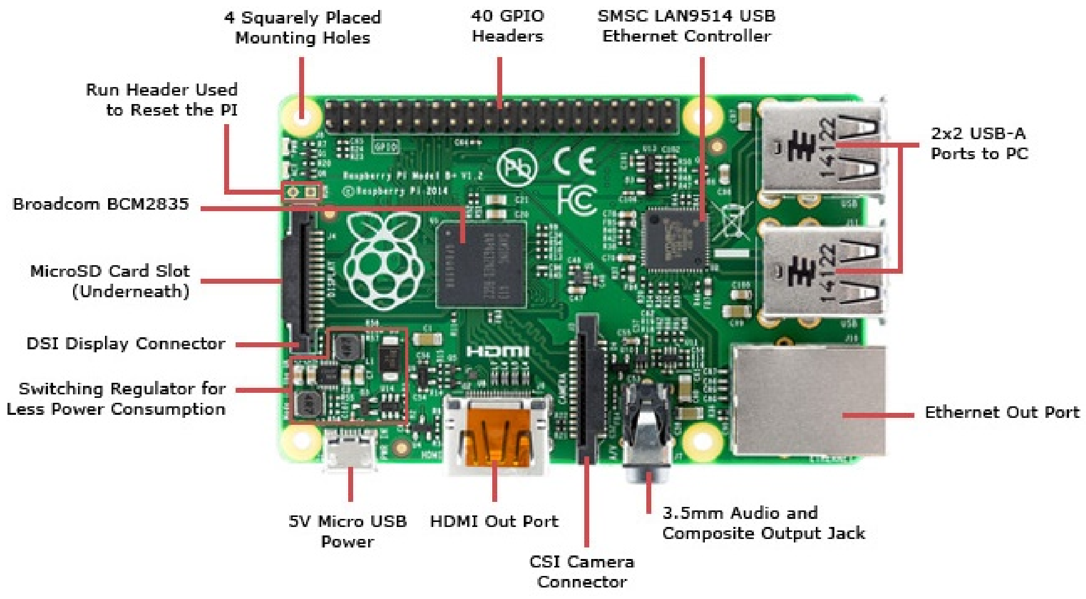

4 Squarely Placed Mounting Holes
40 GPIO Headers
SMSC LAN9514 USB Ethernet Controller
Run Header Used to Reset the PI
2x2 USB-A Ports to PC
Broadcom BCM2835
MicroSD Card Slot (Underneath)
DSI Display Connector
Switching Regulator for Less Power Consumption
Ethernet Out Port
5V Micro USB Power
HDMI Out Port
3.5mm Audio and Composite Output Jack
CSI Camera Connector

Figure 3.13 – Raspberry Pi inputs and outputs diagram ("Raspberry Pi Circuit Note," n.d.).

## 3.4 Connections & Final Build

Pixhawk flight controller will be installed on the inferior plate. After UAV full building, it's difficult to reach Pixhawk input connections, and to avoid that, some essential components need to be connected before mounting upper frame plate. The figure 3.14 shows the main connections to Pixhawk.

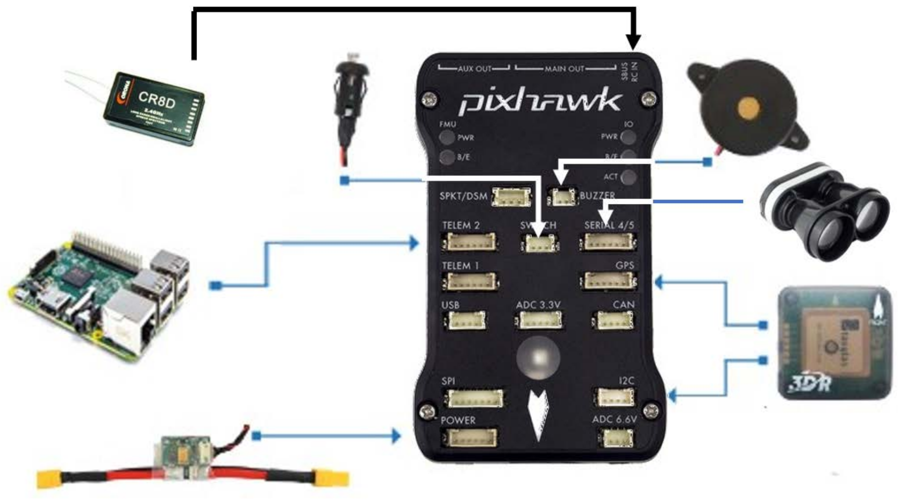


Figure 3.14 – Pixhawk's essential main connections. Adapted from: (Murilhas, 2015).

The figure 3.15 represents the custom-built UAV totally configured and ready to fly, including all the previous described components. To help at UAV landings, a landing gear was installed.

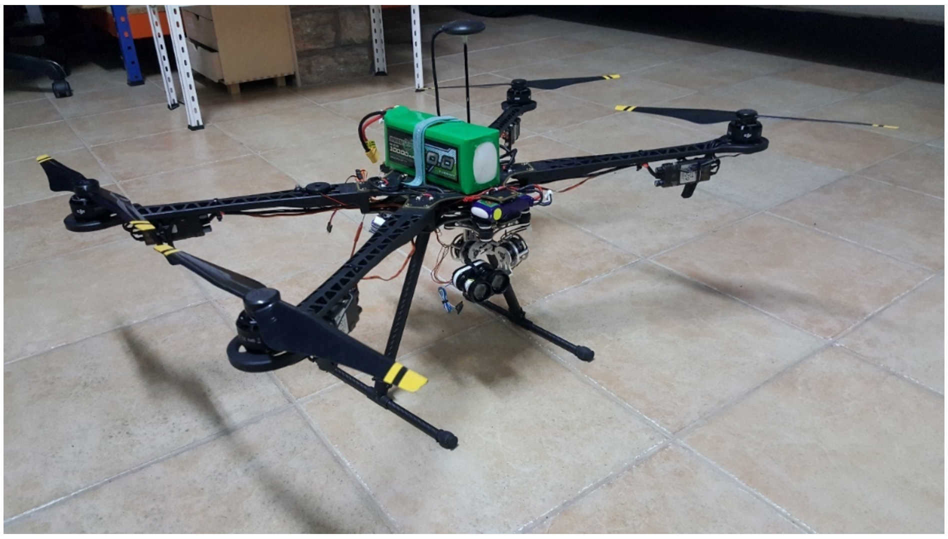


Figure 3.15 – Final UAV build.

# Chapter 4
# CONTROL AND COMMUNICATION

UAV’s possible control and communication modes are described in this chapter, each one presenting its features and limitations. MAVProxy was also mentioned as a gateway between control interfaces.

## 4.1 UAV Control

There are two ways to remotely control an UAV. It can be controlled by a basic Radio Controller, or a Ground Control Station. Some features are only available if the user have a GCS software. Each controlling method has its own way of communication between Pixhawk and the user. For most civilian purposes, RC control is mostly used (Murilhas, 2015). An UAV has two different rotors state: ARMED and DISARMED. ARMED state means UAV is ready to take off. DISARMED state totally disables rotors and is unresponsive until ARMED. In order to “arm” an UAV, specific RC input combinations must be made. In case of GCS control, a MAVLink command can be sent.

### 4.1.1 RC Control

Radio controlling an UAV, as describe before on this dissertation’s Radio Controller chapter subsection 2.3.1, it is the bottom basis to start controlling an UAV. The user has manual control of UAV’s Pitch, Roll, Yaw and Throttle behaviors. Each of this behaviors are associated with a radio frequency channel (Saraiva, 2015). In order to start controlling an UAV, a minimum 4-channel radio receiver and transmitter is needed. Other channels are used to start missions, change flight modes or to control servos. The RC controller used in this dissertation works in 2.4GHz frequency band and uses Pulse-Width Modulation (PWM) to send its signals. The pulse length determines the channel’s output value. PWM signals’ width varies between 1000 µs and 2000 µs, corresponding to maximum (Full Forward) and minimum (Full Reverse) widths respectively. The PWM pulse width neutral value is 1500 µs. The figure 4.1 describes visually how PWM works.

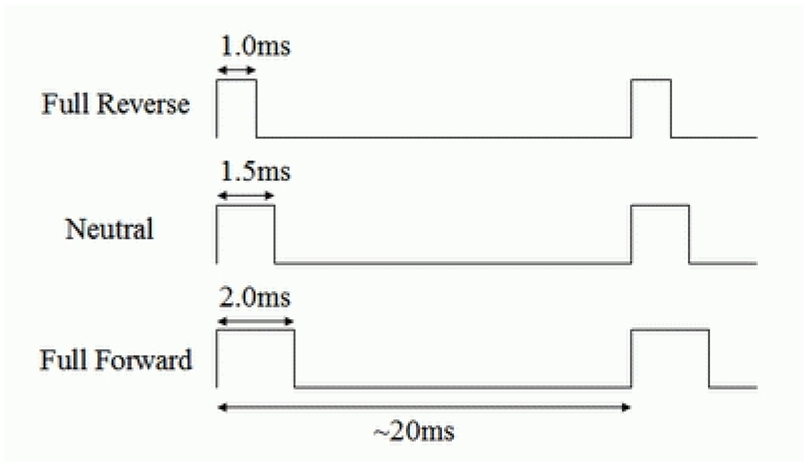


Figure 4.1 – PWM lengths and behaviors (“2.007 Arduino Nano Carrier,” n.d.).

There are two RC configuration modes: Mode 1 and Mode 2. The Mode 1 RC controller has the Roll and Pitch controls on the left stick of the controller, and Throttle and Yaw controls on the right stick of the controller. On a Mode 2 RC controller, the left sticks controls Yaw and Throttle, and the left stick, Pitch and Roll. The use of a RC controller only allows UAV control in user's Line of Sight (LoS). The figure 4.2 shows a Mode 2 RC controller example, and what the direction that user should move sticks in order to increase or decrease described UAV known behaviors.

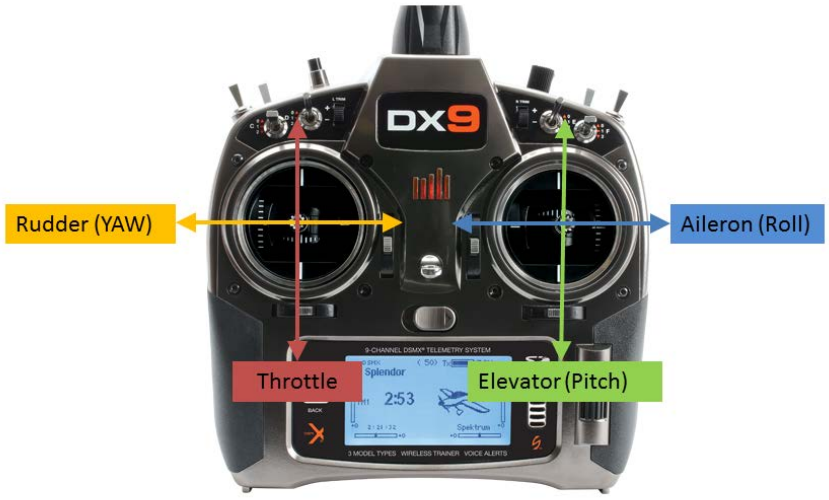


Figure 4.2 – Mode 2 RC controller example (Robert, n.d.).

### 4.1.2 Ground Control Station

The Ground Control Station is the "headquarters" of UAV's operations. It allows user to control and communicate with an UAV. It is possible to update flight controllers' firmware, change predefined parameters, plan waypoint-guided missions, analyze telemetry data, calibrate internal and external sensors, calibrate radio controller channels for minimum and maximum stick limits, check sensors for failures or bad behaviors, check battery status and to send MAVLink commands, which are interpreted by Pixhawk. It also allows for UAV in-flight status, such as real time position monitoring and control (Murilhas, 2015). GCS is the most useful component for controlling, communicating and monitoring an UAV. For this dissertation, Mission Planner GCS software was used. Mission Planner is a full-featured open-source ground control station with a large community support. It was created by Michael Oborne for the ArduPilot open-source autopilot. Available for Windows operating system only, Mission Planner is licensed under the terms of GNU General Public License version 3

(Ardupilot Dev Team, n.d.-d). Mission Planner GCS is not only for UAVs (Plane and Copter firmwares) but also for UGVs (Rover) (ArduPilot Dev Team, 2014). In this dissertation, the use of Mission Planner software provided useful test aids, such as GPS flight path and LiDAR sensor reading and data logs, useful to elaborate results from testing environments.

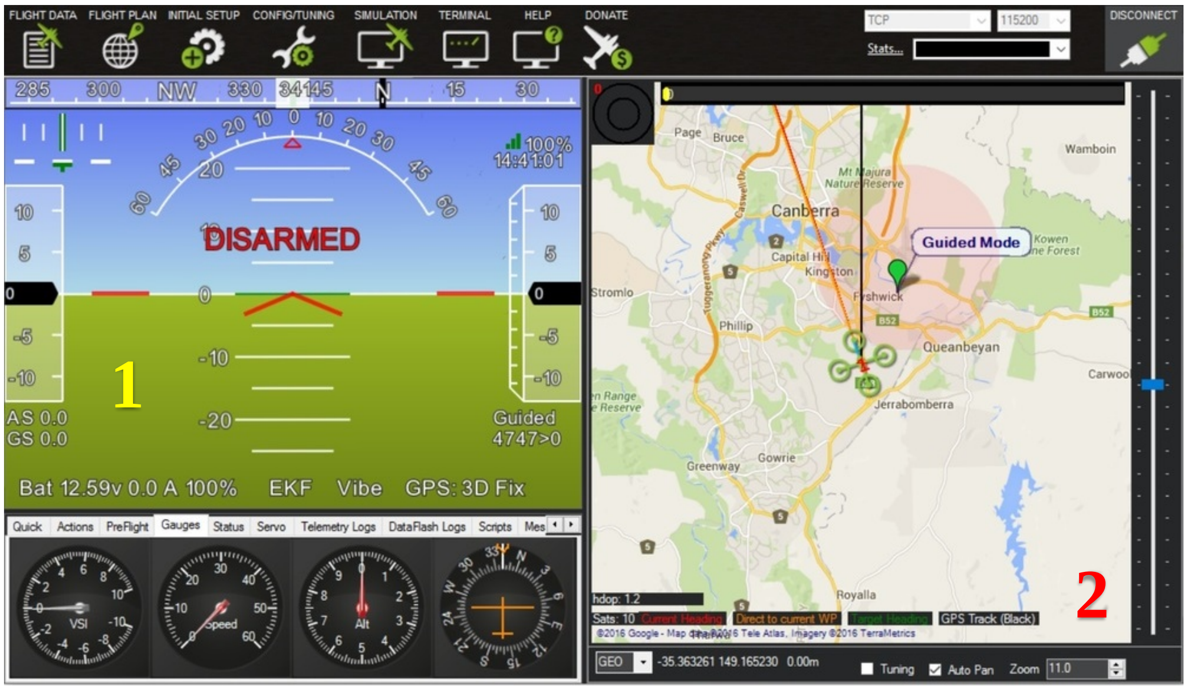


Figure 4.3 – Mission Planner software's main view (ArduPilot Dev Team, 2014).

The figure 4.3 shows Mission Planner software on its main view (Flight Data). In this view on the figure's caption number 1 (left side), it is possible to read IMU sensors information, such as UAVs inclination and heading, vertical speed and altitude; GPS fix, battery level, communication signal strength, current flight mode and UAV's rotor state.

In the figure's caption number 2 (right side) there is a two-dimensional map (provided by Google Maps) where it is possible to see UAV's current GPS position and satellite count, pre-programmed waypoints and monitor in real time an UAV's flight.

## 4.2 UAV Communication

UAV communication is a critical requirement in UAV controlling and monitoring (Saraiva, 2015). It establishes the bridge between user controlling commands and UAV responses. There are various types of communication methods in order to exchange data between flight controller and GCS. Each method has its own components dependency and limitations. The subsections

4.2.1 to 4.2.3 describe each communication method, what component requirements need to be met and its features and limitations.

### 4.2.1 Wi-Fi Telemetry

Wireless communication between user and flight controller, using Wi-Fi technology, can only be made with an intermediary component like Raspberry Pi. In order to exchange data between GCS and UAV, Raspberry Pi must have a Wi-Fi dongle connected. Then, in Raspberry Pi, there is the need of creating a Wi-Fi hotspot, also known as Wi-Fi local network, for GCS to connect to it. The Wi-Fi dongles work at radio frequency band of 2.4 GHz.

Although, there are some limitations concerning this type of Wi-Fi connection to get telemetry info, the dongle installed on Raspberry Pi must have a good Wi-Fi antenna power for a higher maximum range. Maximizing the Wi-Fi range leads to maximizing the distance between UAV and GCS without losing communications. In order to connect to Raspberry Pi Wi-Fi hotspot, the user must connect to SSID of the respective hotspot (Saraiva, 2015). Using WPA2 encryption method on Wi-Fi hotspot increases security between communications because it avoids non-wanted users to access our Raspberry Pi Wi-Fi network.

### 4.2.2 Radio Telemetry

Radio telemetry is another method for communication between Pixhawk and GCS. By using radio telemetry, the receiver (Rx) is directly connected to Pixhawk's flight controller, and the transmitter directly connected to GCS as well. These radio transmitters and receivers are already prepared to exchange MAVLink data between them, if the chosen device supports ArduPilot messaging protocol, MAVLink. There are various radio telemetry devices, and each one has different communication frequencies, range, power consumption and cost. Mostly radio telemetry devices operate at 433 MHz (which is legal radio frequency to use in Europe) (Oliveira & Gersão, 2015) (Barbatei, 2014). In this dissertation, a 3DR Radio Telemetry kit was used for its price and compatibility. It has a 1.6 km communication range, and it was just used for test purposes. An XBee-PRO 868 has a large range (40 km), which can be very useful for search and rescue operations, with the disadvantage of being more expensive (Barbatei, 2014).

### 4.2.3 MAVProxy

In order to establish communication between Pixhawk and “Sense and Avoid” algorithm running on Raspberry Pi, both ends need to use same messaging protocol (MAVLink) to make communication possible (Saraiva, 2015). MAVProxy is a gateway intermediary command-line ground control station. Open-source, developed by Canberra UAV OBC team and it was built in python programming language. MAVProxy is available for Windows, Linux and Mac OS. For being light-weight, this GCS can run on small computers with less effort, such as Raspberry Pi (Saraiva, 2015) (Murilhas, 2015). In this dissertation, MAVProxy will serve as a MAVLink message gateway only. Its main goal is to exchange MAVLink messages between “Sense and Avoid” algorithm and Pixhawk’s flight controller. In order to accomplish that, MAVProxy will be always running on Raspberry Pi, waiting for command inputs supplied by “Sense and Avoid” algorithm. MAVProxy is not a unique controlling station. Other GCS can be connected to Pixhawk and/or Raspberry Pi via Wi-Fi, RC or via Radio Telemetry. MAVProxy is only needed for decision in “Sense and Avoid” algorithm. When no decisions are made, other control methods can override MAVProxy GCS and fly an UAV normally. In chapter 5, “Sense and Avoid Algorithm”, there is a mechanism that forces an UAV to disable any other types of UAV control, in case of object detection and in collision course.

# Chapter 5
# SENSE AND AVOID ALGORITHM

This chapter explains the algorithm developed in order to deploy an autonomous object collision avoidance system using distance ranging for UAVs.

## 5.1 Overview

The Sense and Avoid algorithm was developed to achieve this dissertation's main goal: an autonomous avoidance system for UAVs in object collision course to use in search and rescue operations. This algorithm was designed to work fully on outdoor environments only. It was developed in Java programming language, for any operating system that supports Java VM and any flight controller that supports MAVLink messages and ArduPilot ArduCopter (for multirotors) firmware. In order to generate MAVLink messages in a way that MAVProxy could interpret as commands, there was the need of a MAVLink interpreter that uses Java as same programming language as "Sense and Avoid" algorithm. A Java library named MAVLink Java was used and it was developed by GitHub's username Ghelle (Ghelle, n.d.). It is an essential tool in order to read messages generated from Pixhawk and to send MAVLink commands programmatically.

This algorithm was tested on 3D simulated environments and on real outdoor environments. "Sense and Avoid" algorithm relies only in LiDAR external sensor readings to calculate objects in collision course and then, making decisions. These decisions are based on how an UAV is being controlled. For each different way of controlling an UAV, the algorithm behaves according to the situation. This algorithm needs to be always running in order to work. So, in this case and since it's installed on Raspberry Pi's OS, the "Sense and Avoid" algorithm always runs at Raspberry Pi's boot, by running a startup script. In this chapter, every "object" word denomination can not only be referred as an object itself. An "object" can be a person, animal, wall, tree or other "instrument" that has an opaque surface.

The flow chart depicted in figure 5.1 describes how "Sense and Avoid" algorithm works. This chart was developed by Bizagi Process Modeler, a free-to-use software for Windows environments developed by Bizagi company, that allow users to create process models for business purposes (Bizagi, n.d.). In this case, this software was used for modeling a cycling process, which accurately describes the internal functionality of "Sense and Avoid" algorithm.

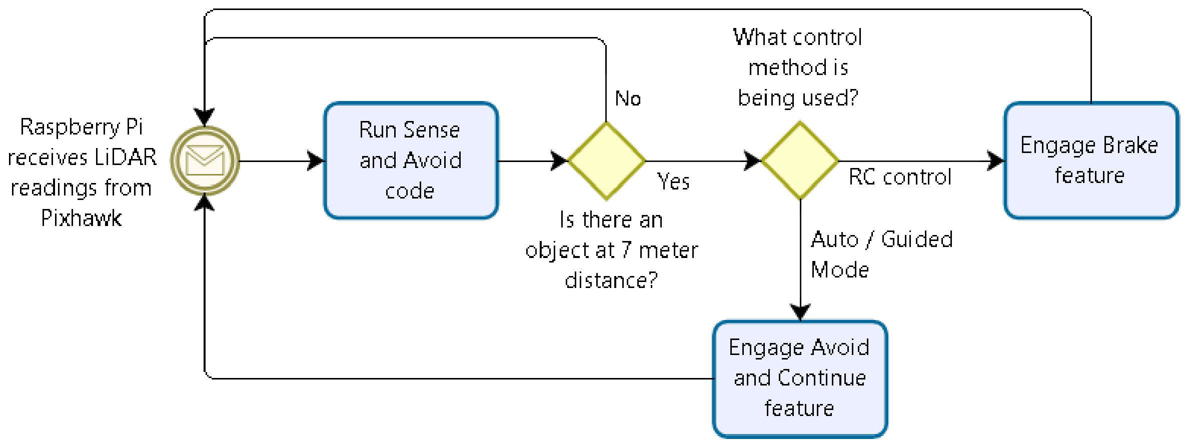

Raspberry Pi receives LiDAR readings from Pixhawk
Run Sense and Avoid code
No
Yes
Is there an object at 7 meter distance?
What control method is being used?
RC control
Auto / Guided Mode
Engage Brake feature
Engage Avoid and Continue feature

Figure 5.1 – “Sense and Avoid” internal functions.

“Sense and Avoid” algorithm starts by receiving LiDAR readings from Pixhawk’s flight controller through MAVProxy. Then proceeds by running a cycling (threaded) code that runs every 0.25s (this value was chosen as an example initial value and worked on experimental purposes). On every cycle, “Sense and Avoid” algorithm checks LiDAR distance read (that automatically outputs object distance in meters). If LiDAR doesn’t detect an object that has a distance inferior or equal to 7 m, it means no action is required and UAV doesn’t change its behavior. This ends cycle process and starts a new one. But, if LiDAR detects an object that has a distance inferior or equal to 7 m, the algorithm rapidly detects which controlling mode active at the moment: RC control method or autonomous methods such as “Auto” or “Guided” flight modes. If an UAV is being RC controlled, then the algorithm activates “Brake” feature. Or, if the UAV is flying on “Auto” or “Guided” flight modes, the algorithm will activate “Avoid and Continue” feature. At the end of “Brake” or “Avoid and Continue” features, the process cycle ends. The previous controlling method is resumed and new cycle begins, as it can be seen on the flow chart.

The 7 m LiDAR distance is the maximum value that the UAV can be distanced of an object on collision course before activating one of the algorithm’s features. Inferior distance values were considered as well in order to predict instant object appearing which could cause UAV collision with that object. The 7 value is not a threshold value, but was given as an example. It was initially chosen when performing experimental tests on a 3D simulator and there were no complications on achieving tests goals. The section 5.2 describes how “Sense and Avoid” algorithm makes decisions based on different modes, and what effects are produced in UAV behaviors.

## 5.2 Algorithm’s Modes

The “Sense and Avoid” algorithm has two modes: “Brake” and “Avoid and Continue”. Each mode is enabled accordingly to UAV’s current controlling method. The “Brake” mode is for RC controlling only and “Avoid and Continue” mode is for autonomous tasks, such as “Guided” flight mode and “Auto” flight mode. The subsections 5.2.1 and 5.2.2 describe each mode.

### 5.2.1 Brake Mode

The “Brake” mode is only engaged if LiDAR “sees” an object at a distance, *e.g.* 7 m or inferior, and if the UAV is currently being controlled by an RC controller. “Brake” mode is an essential feature for users that want an UAV with an object collision avoidance system. This features prevents RC controlled UAV from object collision, activating the “Brake” flight mode available on Pixhawk’s Copter firmware. This mode will only work if UAV has GPS position enabled, because Brake flight mode depends on GPS. The Annex A contains Pixhawk’s ArduCopter flight modes. The figure 5.2 represents the UAV approaching an object on collision course.

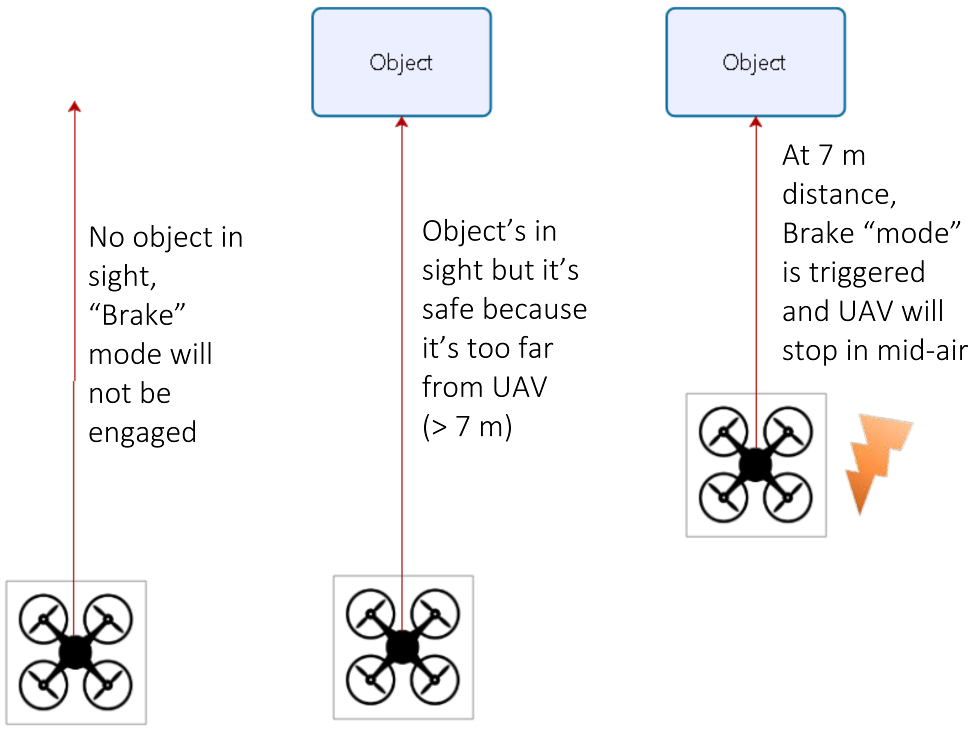

Object
Object
No object in sight, “Brake” mode will not be engaged
Object’s in sight but it’s safe because it’s too far from UAV (> 7 m)
At 7 m distance, Brake “mode” is triggered and UAV will stop in mid-air

Figure 5.2 – “Brake” mode example.

An object-free environment is represented on figure 5.2 (left side), where no action is required because there is no object in “sight” or the object distance from UAV is higher that LiDAR maximum distance measurement. On the same figure’s middle, an object is detected but no action is required because “Break” mode is only triggered if that object is at a distance of 7 m or lower. On the figure 5.2 (right side), the same object is now at a 7 m distance. This is when “Brake” feature is triggered. This mode switches UAV’s current flight mode to “Brake” flight mode. This flight mode will stop the UAV mid-air and holding the same altitude the UAV had upon “Brake” flight mode switching. This will cause UAV to drift until hold position. Higher speeds mean higher brake times. After activation, any other form of controlling the UAV is disabled, and this means RC attitude control. Then, UAV proceeds to fly backwards until LiDAR measures object distance greater than 7 m. When that reading occurs, previous flight mode is activated and RC control is resumed. GPS is required to save current position. This position refers to the UAV’s actual position when Brake flight mode was activated. In case of drifting, this mode will readjust UAV’s position to the position when Brake flight mode was activated. The flow chart represented by figure 5.3 demonstrates how “Brake” mode changes UAV behaviors.

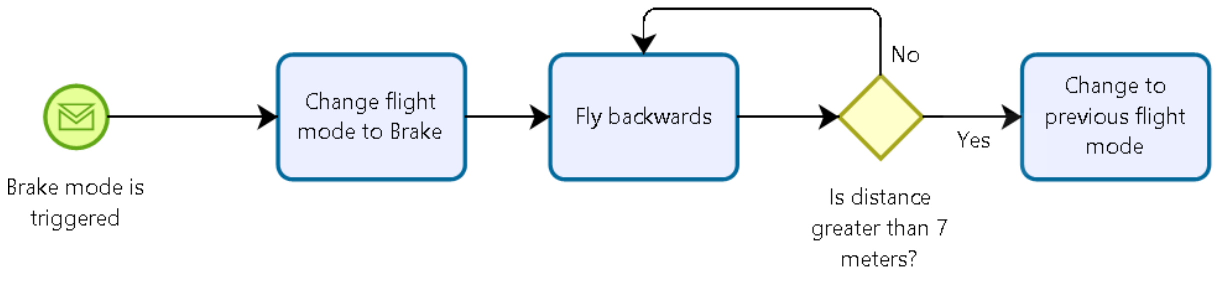

Brake mode is triggered
Change flight mode to Brake
Fly backwards
No
Yes
Is distance greater than 7 meters?
Change to previous flight mode

Figure 5.3 – “Brake” mode flow chart.

### 5.2.2 Avoid and Continue Mode

This mode is useful for fully autonomous tasks, such as missions. Implementing this feature not only guarantee object collision avoidance, but also resumes previous established goals. This mode was developed for search and rescue missions that require object collision maneuvers. The flow chart represented by figure 5.4 describes how “Avoid and Continue” mode behaves.

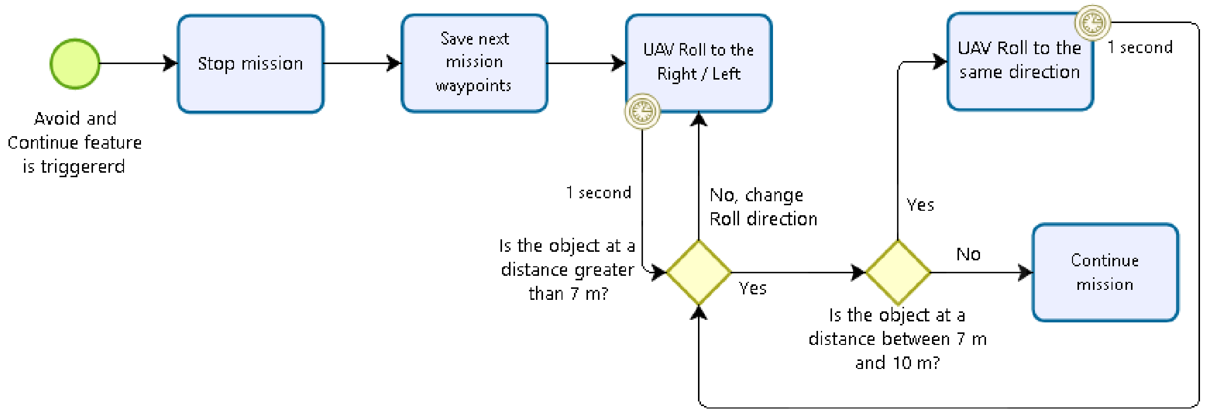

Avoid and Continue feature is triggererd
Stop mission
Save next mission waypoints
UAV Roll to the Right / Left
1 second
Is the object at a distance greater than 7 m?
No, change Roll direction
Yes
Yes
UAV Roll to the same direction
1 second
No
Is the object at a distance between 7 m and 10 m?
Continue mission

Figure 5.4 – “Avoid and Continue” mode flow chart.

When “Avoid and Continue” mode is triggered, it comes from an autonomous flight mode: “Guided” or “Auto” flight mode. Those modes use pre-defined GPS positions. These positions are also called waypoints, and it sets a fly course for the UAV. This fly course is called mission. Missions are performed autonomously, flying orderly from the first waypoint created until the last one. A mission can have multiple waypoints. But, if in between these waypoints there’s an obstacle in collision course, the algorithm will be triggered. Firstly, the algorithm will stop the mission, in order to prevent collision. Then, will orderly save the list of non-traveled waypoints, *i.e.*, the next mission waypoints. This step is very important, because the UAV needs to keep record of its mission main goals. After saving waypoints, the UAV proceeds to Roll to the right or left. This UAV direction change (left or right) is chosen randomly, because the LiDAR sensor has a narrow beam (SF11/C beam divergence is 0.2$^{o}$) and it can only sense the object at one small point in UAV’s front. For that reason, it is impossible to know the width of the object in order to choose direction that the UAV should take for a fast avoidance (shortest path to an open area). But, a low divergence beam can be useful, because it makes object distance measurement more accurate. If the object on sight is at a distance greater than 10 m, then the algorithm will resume mission by loading previous saved waypoints. If the distance is between 7 m and 10 m, the UAV will repeat the same Roll attitude in the same direction. This prevents the cases when the Roll attitude duration of 1 s is not enough to continue mission safely. These 3 m distance value is also called “threshold”. But, if the distance is not greater than 7 m, there is the need of changing actual Roll direction. For example, if the UAV’s LiDAR is reading an object at 7 m, then it proceeds to Roll right, and that distance increases even more, the UAV could be facing the object with a certain angle that will rapidly increase the distance between

UAV and that object. To avoid that, the UAV will invert the previous Roll rotation until distance is higher than 7 m so it can resume mission as described. The Roll attitude duration of 1 second was chosen as an example value when performing experimental tests. This value should be adjusted according to possible-to-collide environment objects. Roll attitude duration can be very aggressive on smaller UAV, because they're more agile. By default, autonomous flight modes fly at an approximate horizontal speed of 5 m/s (Ardupilot Dev Team, 2016). The case represented on figure 5.5 describes what could be a real flight scenario on a city by avoiding collision with a building.

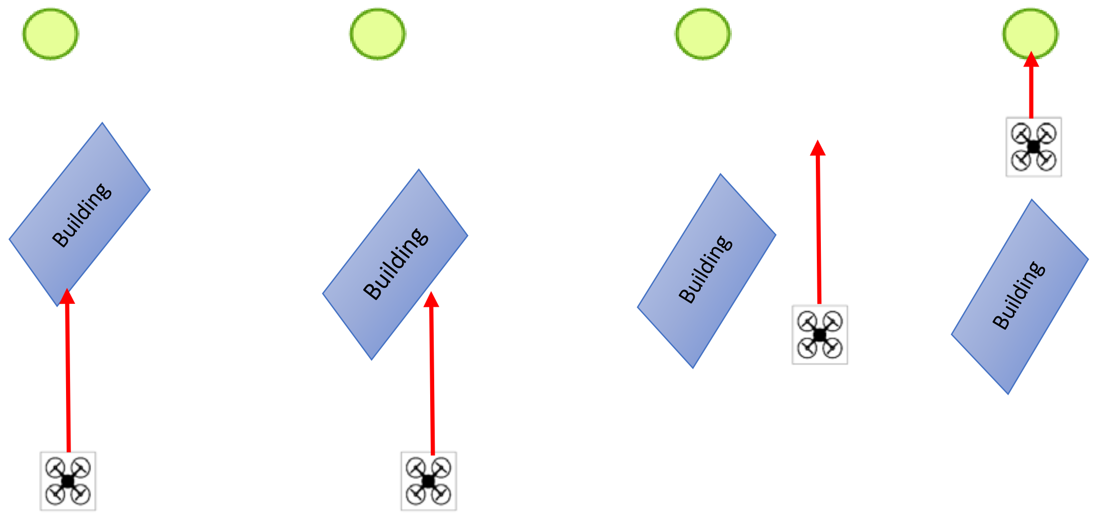


Figure 5.5 – "Avoid and Continue" mode scenario example.

The figure 5.5 demonstrates an example situation on how "Avoid and Continue" mode can be useful if used on a city or village. This represents an autonomous mission, which the green circle represents a waypoint. On the left side of figure 5.5, the UAV sense the building within 7 m of distance. Then proceeds to choose randomly a Roll side, represented on the right side of the previous one, as a right roll maneuver. After flying to the right 1 s, the UAV no longer have the building on its collision course, represented on figure 5.5 before the right side. After the object collision maneuver, on the figure's right side, the UAV proceeds to fly to the designated waypoint that was previously saved. If UAV is again in collision course with the same object (for cases when the building is longer, and/or the roll time value was too low), the procedure is the same as the original, considering the same object as if was another object.

# Chapter 6
# SIMULATION AND EXPERIMENTAL RESULTS

This chapter includes “Sense and Avoid” simulation and experimental results, which contains different test environments. The algorithm’s performance was made to be compared with similar solutions.

## 6.1 Environments

“Sense and Avoid” algorithm experimental tests were made on two different environments. Firstly, and for security reasons, there was a need to find a solution that was secure, easy to deploy and close to real UAV behaviors. A 3D simulated environment was the perfect solution for test “Sense and Avoid” algorithm. After a series of trial-and-error tests, resulting on many successful results on “Sense and Algorithm” main features made on a 3D environment, some tests were made on the custom UAV built (fully described on chapter 3) on an outdoor environment. The subsections 6.1.1 and 6.1.2 describe the two test environments and what tests were made on each one. A performance evaluation on the algorithm was made to justify its efficiency.

### 6.1.1 3D Simulation

In order to test the “Sense and Avoid” algorithm on a 3D simulated environment, it was necessary to find a solution that represented real outdoor environments and UAV “close-to-real” behaviors. Gazebo is an Apache 2.0 licensed open-source 3D environment simulator that is capable of simulating multi-robot behaviors on indoor or outdoor environments. It also features sensor readings (for object awareness) and “physically plausible interactions” between rigid objects (“Gazebo,” n.d.; Gomes Carreira, 2013). This simulator operates on Robot Operating System (ROS), which is BSD licensed open source library toolbox to develop robot-based applications. The ROS toolbox allows installing third-party plugins in order to add extra features useful to simulate all UAV’s flying requirements, such as control and communication methods. (Gomes Carreira, 2013; Open Source Robotics Foundation, 2014). The Gazebo simulator makes it possible to test “Sense and Avoid” algorithm because it provides realistic scenarios, with 3D model inclusions and real physics simulation. The ROS/Gazebo implementation will work as an external simulator for Pixhawk’s ArduCopter firmware (Erle Robotics, n.d.). Its main features are:

- Simulate 3D UAV’s physical stability and user command responses;
- Read sensor data from the virtual-created UAV (IMU, Compass);
- Communicate between user inputs and UAV model;
- GPS information and SoNAR / LiDAR distance readings (Erle Robotics, n.d.).

The figure 6.1 demonstrates how ROS/Gazebo works internally in order to send and receive data from user inputs and its execution.

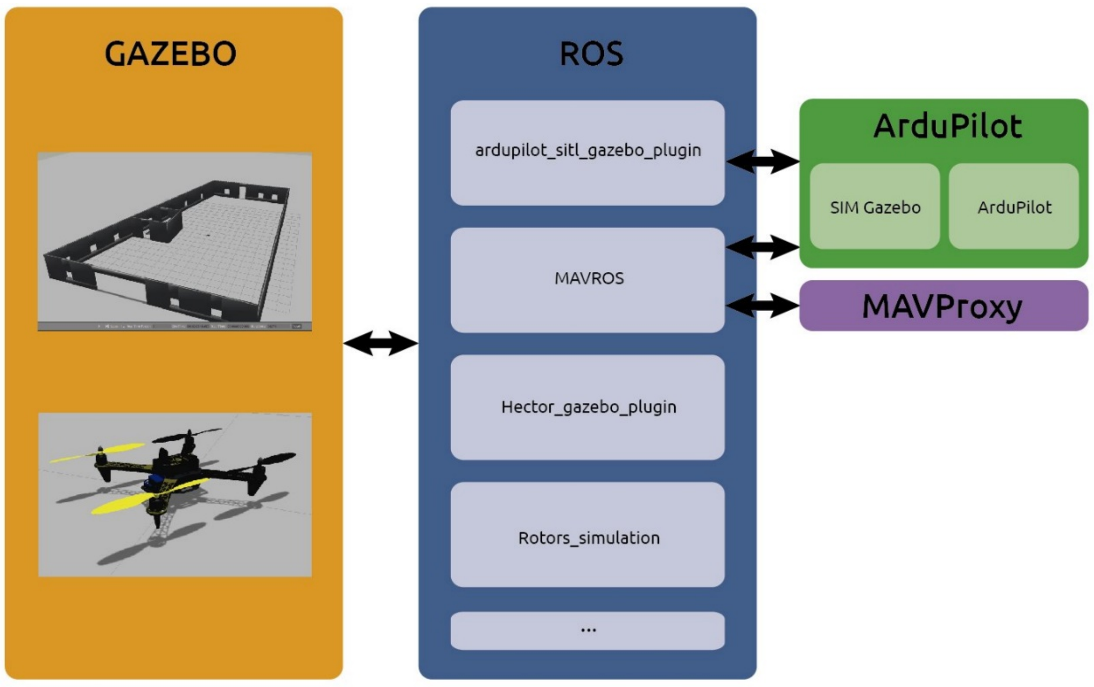


Figure 6.1 – ROS/Gazebo interfacing with virtual-created UAV. (Erle Robotics, n.d.)

On the figure, “Gazebo” features the 3D environment, including the UAV model and an example indoor scenario. ROS contains a list of plugins that can be used in order to develop the UAV 3D solution. The ArduPilot SITL Gazebo plugin is responsible for interfacing the virtual-created UAV (ArduPilot SITL simulation) with Gazebo simulation. The MAVROS plugin package, or Micro Air Vehicle ROS, provides communication features between ArduPilot firmwares (ArduCopter in this case). It uses the MAVLink communication protocol to exchange data. Additionally, MAVROS provides a bridge for MAVProxy GCS, which will be useful to “Sense and Avoid” algorithm, since it relies mainly on MAVProxy GCS to stablish a communication and control gateway. The Rotors Simulation plugin provides Gazebo simulator the UAV kinematics and sensor readings (IMU and LiDAR / SoNAR) (Erle Robotics, n.d.; Furrer, Burri, Achtelik, & Siegwart, 2016). With these plugins, it is possible to test “Sense and Avoid” algorithm by experiencing real UAV responses to algorithm decisions. The LiDAR sensor simulation on Gazebo can read real distances from rigid 3D objects on the presented scenario, which gives an approximate real-feel experience from outdoor UAV flights and LiDAR readings. This simulator is only available for Linux operating systems. For this case, the Linux Ubuntu 14.04 LTS distribution version was used on a Virtual Machine environment, installed on Windows 10 operating system.

Since this dissertation’s main objective is to use “Sense and Avoid” algorithm on a Search and Rescue approach, there was the need to create a scenario that includes the UAV facing an object in collision course. In early test stages, a 3D cube was used as an object-to-avoid scenario model. All Gazebo and ROS installation procedures and some 3D models were made by Erle-Robotics company, which presents a well-made guide that was essential in order to start our 3D UAV model in an outdoor environment.

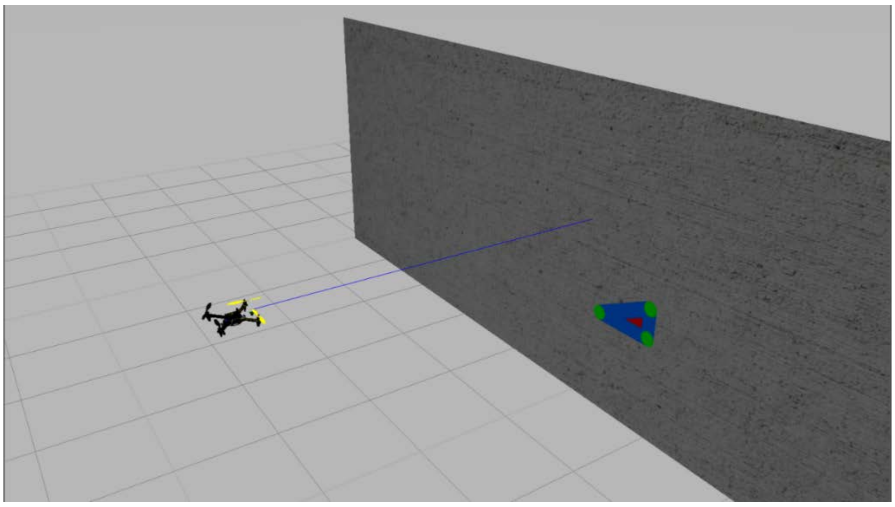


Figure 6.2 – Quadcopter UAV model in a simple environment.

In early algorithm development stages, a simpler environment was made to test algorithm’s efficiency.  The figure 6.2 presents the UAV 3D model, with a LiDAR sensor installed and is facing a concrete wall model. All models are available to use upon Gazebo installation. The UAV model named “Erle-Copter” was created by Erle-Robotics company. In order to add the laser “effect” on the UAV 3D model, some additional configurations were made. This simple environment served as the main field test of the “Brake” mode of “Sense and Avoid” algorithm. By receiving LiDAR readings of wall distance, it was possible to perform RC-based inputs, and therefore, acting like a real user sending RC controller commands. The RC-based inputs were sent via MAVProxy console. The simulation on this test scenario is a user-deliberated UAV control in order to collide with the concrete wall. The “Brake” mode will prevent it, and when compared with real cases, it relates to non-intentional collisions, classified as human errors. Annex B shows a top-camera view of this environment. The figure 6.3 shows a more realistic environment. This environment was the base environment to test “Avoid and Continue” mode.

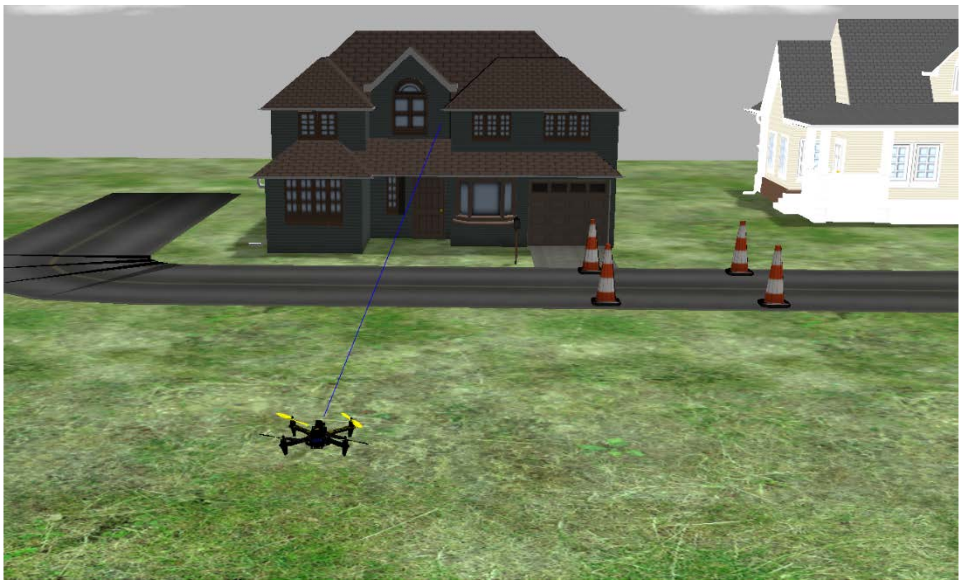


Figure 6.3 – 3D Outdoor environment simulation.

The figure 6.3 represents a possible outdoor environment, which contains various scenario components. This environment was created to test “Avoid and Continue” mode of the “Sense and Avoid” algorithm. In order to simulate a Search and Rescue scenario, this simulation represents an UAV flight using Pixhawk’s Auto flight mode (waypoint navigation), flying from point A to point B in order to perform a certain SAR mission. Since it doesn’t have a clear path, the UAV must avoid the object in collision course and proceed to reach B point successfully. The point A is at UAV’s current position and point B is behind the brown house model. The figure B.2 present on Annex B contains a top-camera view that explains waypoint locations.

### 6.1.2 Real Outdoor

In order to test “Sense and Avoid” algorithm, using the custom built UAV, some experimental tests were made on an outdoor environment. Firstly, some Guided and Auto missions were made to ensure UAV’s stability and to avoid erranous behaviors when testing the algorithm. This tests were performed in an open-air area and obstacle-free environment. After ensuring UAV’s stability on autonomous missions, a obstacle was needed to test “Brake” mode. Unfortunatly, only “Brake” mode was tested on a real environment, because tests location was on a urban environment, and UAV failures could result on damaging the UAV or the object itself in case of crashing.

The figure 6.4 represents the 2D satellite map of the outdoor location and its description. This snapshot was taken from Google Maps utility. The tests were made in Encarnação, Mafra, Portugal.

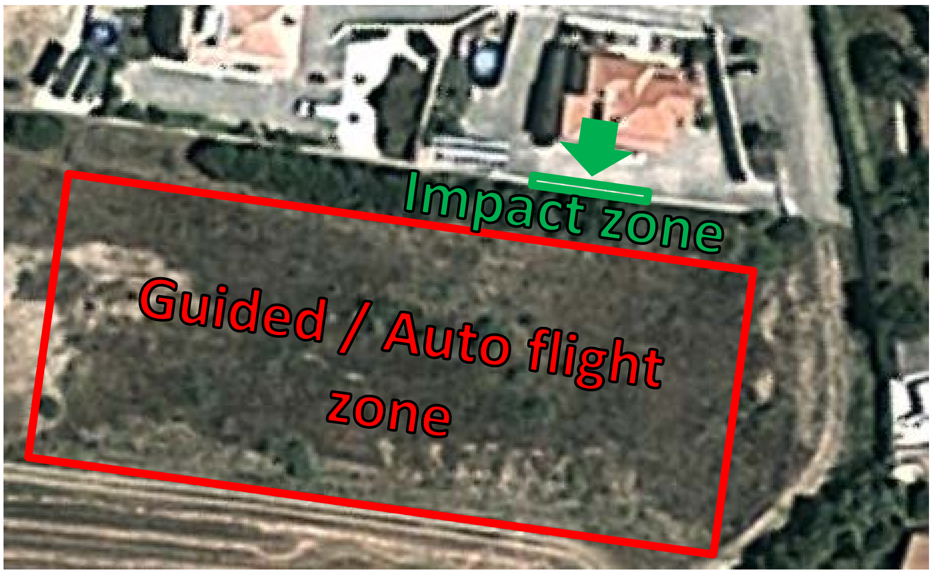


Figure 6.4 – Real outdoor environment and UAV flight zones.

The “Guided / Auto flight zone” delimits the location were pre-test flights were made in order to assure that everything was ready to test “Brake” mode. The “Impact zone” represents the object to test collision avoidance feature of “Brake” mode. The direction and heading of the UAV is represented by the green arrow

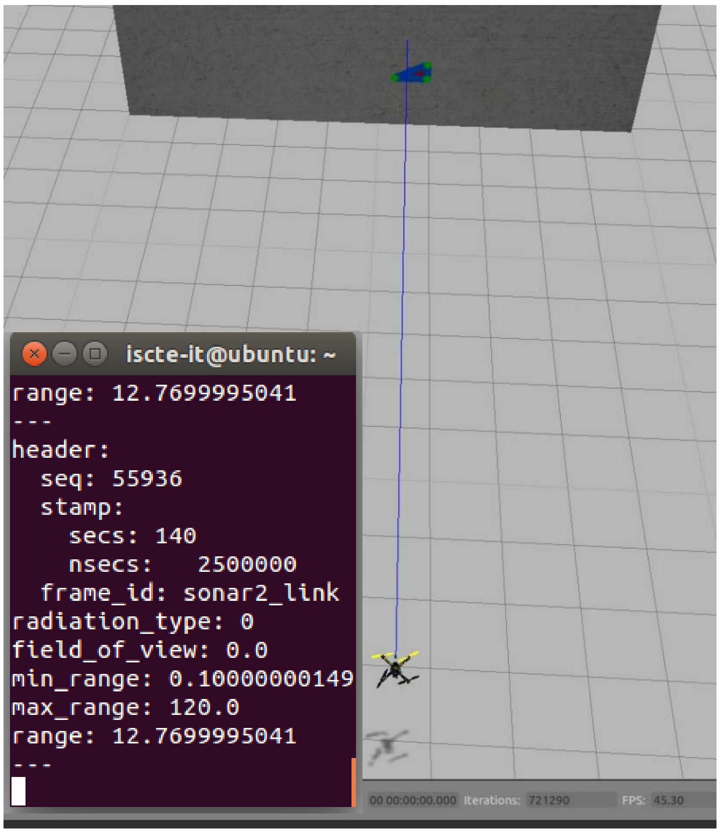
The “Brake” mode was tested on both environments, but “Avoid and Continue” mode was only tested on a 3D simulated environment. Of all tests made, three of the best test case scenarios were chosen in order to evaluate “Sense and Algorithm” performance on both “Brake” and “Avoid and Continue” modes: Two tests made on 3D environments, each one for each mode; and one test on outdoor environment, only for “Break” mode. The previous tests followed a known way of resolving and optimizing solutions: trial and error. These tests were very important in order to achieve better results and to prove “Sense and Avoid” algorithm’s main objectives.

## 6.2 Results and Performance Evaluation

The subsections 6.2.1 and 6.2.2 show the best-case scenarios when testing one of each “Sense and Avoid” modes in both 3D and real outdoor environments.

### 6.2.1 Brake Mode

The first tests were made with algorithm’s “Brake” mode. The figures 6.5 to 6.8 demonstrate the sequential steps taken.

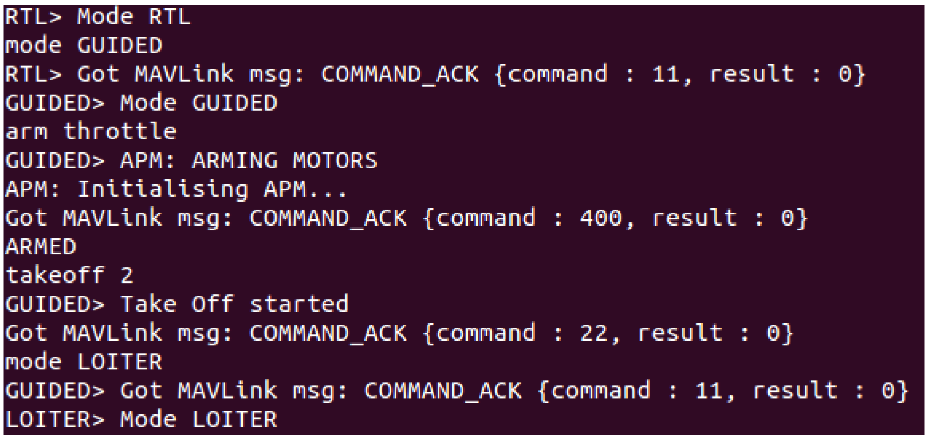


```
range: 12.7699995041
---
header:
  seq: 55936
  stamp:
    secs: 140
    nsecs:   2500000
  frame_id: sonar2_link
radiation_type: 0
field_of_view: 0.0
min_range: 0.10000000149
max_range: 120.0
range: 12.7699995041
---
```

Figure 6.5 – “Brake” mode test start position.

The figure 6.5 demonstrates the UAV initial position on 3D environment in order to start “Brake” mode tests. The grey object at UAV’s front is a concrete wall model. This model will be used as object in collision course with the UAV. The blue line represents the “transmitted visible” virtual LiDAR beam ray. The command line window on the inferior left side contains some data read from the “virtual” LiDAR sensor. Some parameters are totally configurable, such as LiDAR minimum and maximum distance measure, field of view, update rate and actual sensor distance measure (“range”, figure 6.5). Replicating as much as possible a real case scenario of LiDAR readings, the parameters were adjusted according to the SF11/C LiDAR altimeter specifications, used in the custom-built UAV. The “max_range” and “min_range” parameter sets the maximum and minimum distance of the LiDAR in use, which is 120 m and 0.1 m, respectively. Since SF11/C LiDAR doesn’t have a horizontal nor vertical field of view, the parameter “field_of_view” was set to 0. In order to launch the data window, the user must open a new command-line window and write “rostopic echo /sonar_front”. This command will print and constantly update the LiDAR sensor readings in real-time.

The start point distance from the wall is approximately 12.77 m. The value was chosen as an example. The UAV is on LOITER flight mode and is hovering at 2 m altitude from ground plane. The figure 6.6 shows how to introduce MAVProxy commands orderly to reproduce a similar behavior.

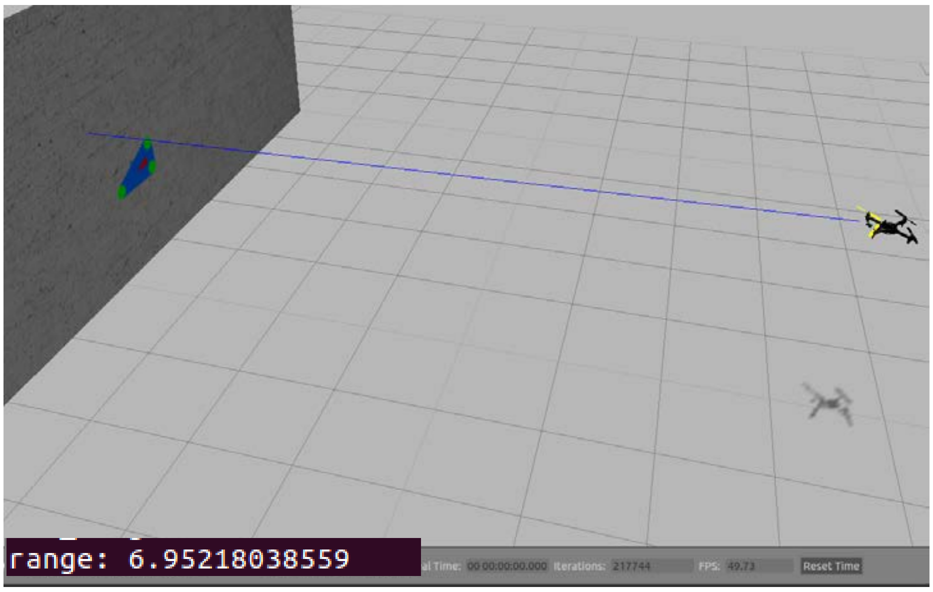


```
RTL> Mode RTL
mode GUIDED
RTL> Got MAVLink msg: COMMAND_ACK {command : 11, result : 0}
GUIDED> Mode GUIDED
arm throttle
GUIDED> APM: ARMING MOTORS
APM: Initialising APM...
Got MAVLink msg: COMMAND_ACK {command : 400, result : 0}
ARMED
takeoff 2
GUIDED> Take Off started
Got MAVLink msg: COMMAND_ACK {command : 22, result : 0}
mode LOITER
GUIDED> Got MAVLink msg: COMMAND_ACK {command : 11, result : 0}
LOITER> Mode LOITER
```

Figure 6.6 – MAVProxy needed command inputs to hover UAV.

To hover the UAV at a fixed position, the user must start by changing to GUIDED flight mode. Then, in order to arm the rotors, the user must write “ARM THROTTLE” command. Using “TAKE_OFF 2” command on MAVProxy will take-off UAV automatically and it will hover at 2 m above ground. Then, to allow RC controller inputs, the user must change to LOITER flight mode in mid-flight. The LOITER mode will maintain its previous defined altitude of 2 m. The “COMMAND_ACK” message type means that UAV received the MAVLink message with a certain command ID and a result. The value “0” on the result means that the command was successfully executed (ETHZ, 2014).

In order to reproduce an UAV movement towards the wall, a RC input simulation was needed. This means simulating a Pitch attitude to the front. For this test, only RC channel 3 was used to change UAV's Pitch attitude. By running the command "RC 3 1480", will perform a smooth UAV Pitch to the front. It's smooth because RC channel's PWM neutral value is 1500 µs, and the difference was only 20 µs. Maximum and minimum PWM values are 1000 µs and 2000 µs, respectively. These values can vary according to RC transmitter used (Ardupilot Dev Team, 2014d). That's why an RC calibration is always needed to establish limits on RC maximum and minimum values. The figure 6.7 shows when "Brake" mode was engaged.



```
range: 6.95218038559
```

Figure 6.7 – "Brake" mode engaged.

The "Brake" mode is configured to change to "Brake" flight mode when UAV is being controlled by RC inputs and LiDAR reads a distance inferior than 7 m. Since it was a very slow and smooth Pitch movement, and wind is not a problem on a 3D environment, the UAV stayed at distance of approximately 6.95 m from the concrete wall. "Brake" mode is prepared to "null" the drift effect movement by flying on a symmetric way. In this case, and to raise the distance again to a value higher than 7 m, a symmetric Pitch attitude is done. Since UAV is on "Brake" flight mode, simulated RC inputs are disabled. To overcome this, the algorithm used a MAVLink command called "RC_CHANNEL_OVERRIDE" to programmatically gain control of the UAV (ETHZ, 2014). In order to obtain the Pitch value to nullify the drift behavior, the equation 6.1 was used:

$$S = RC_n + [RC_n - RC_i] \tag{6.1}$$

where $S$ means the symmetric PWM value, $RC_n$ corresponds to the nominal value of the RC channel (which is a constant value of 1500), and $RC_i$ represents the RC PWM value when “Brake” mode was activated. In this case, $S = 1520$.

This ensures that the same Pitch attitude speed is granted. The symmetric, or backwards Pitch movement is performed at the same speed of frontwards Pitch movement: 20 PWM value of difference, or 1520 RC channel 3 (Pitch) PWM value. After achieving a distance higher than 7 m, the “Brake” mode resets all RC input values to nominal, and restores last flight mode, which was LOITER mode. The figure 6.8 shows the final UAV position after all “Brake” mode procedures.

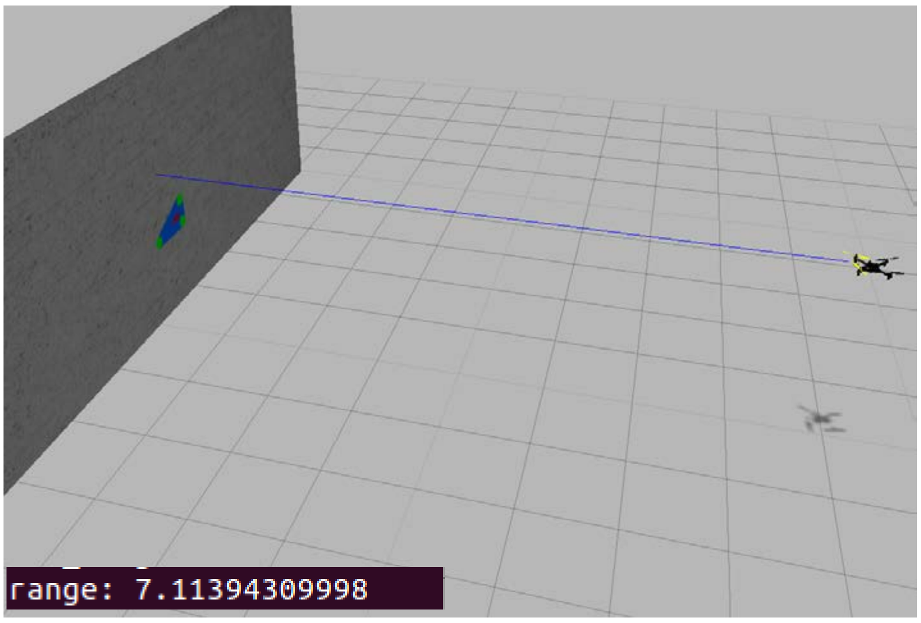

```
range: 7.11394309998
```

Figure 6.8 – “Brake” mode anti-drift feature and UAV final position.

As presented in the figure 6.8, on the UAV final position, LiDAR sensor measures approximately a 7.11 m distance from the concrete wall. This value is distant from the base value (7 m) also because of the drifting effect. The UAV Pitch movement puts the UAV at a certain linear speed, and when switched to LOITER mode again, the UAV takes a little time to stop and hold position, causing the drifting effect.

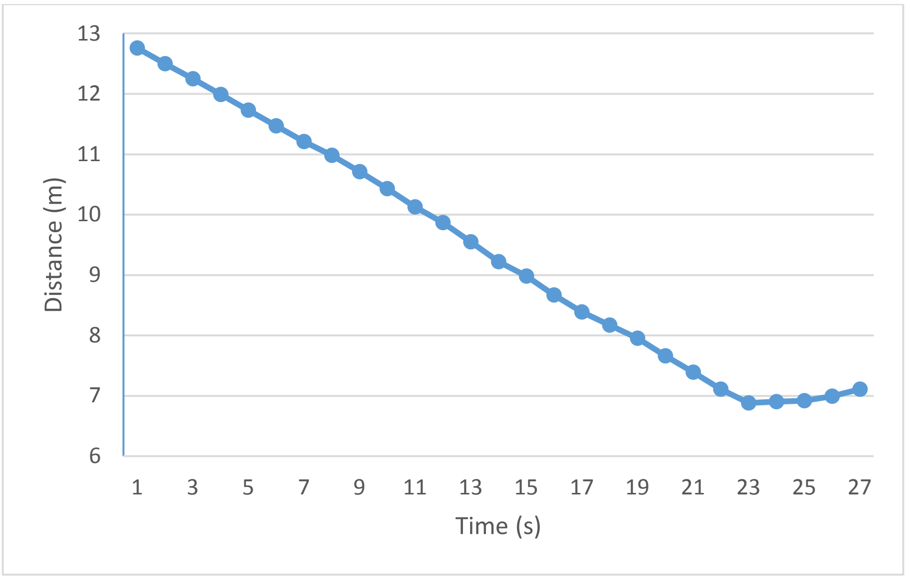


Figure 6.9 – LiDAR readings during test.

The figure 6.9 shows the distance values obtained from LiDAR sensor. Since it was a very slow and smooth Pitch movement, the distance span between seconds is also low (between 22 cm and 28 cm). The distance was obtained considering a sampling rate of 1 s. In this dissertation, the centimeter precision value was considered, which corresponds with SF11/C LiDAR precision measurement values.

With all the data gathered from LiDAR sensor, the “Brake” mode performance on a 3D simulated environment was as expected. It avoided collision with the concrete wall, and proved its concept. The algorithm behaved as expected, changing flight modes and nullifying drift effect. There was no delay on communications, because all procedures were made locally on the same computer. The algorithm ran through MAVProxy, simulating its installation on Raspberry Pi. This means that it’s expected to work on real case scenarios. The above test scenario was performed on the custom-built UAV on the location described in section 6.1.2.

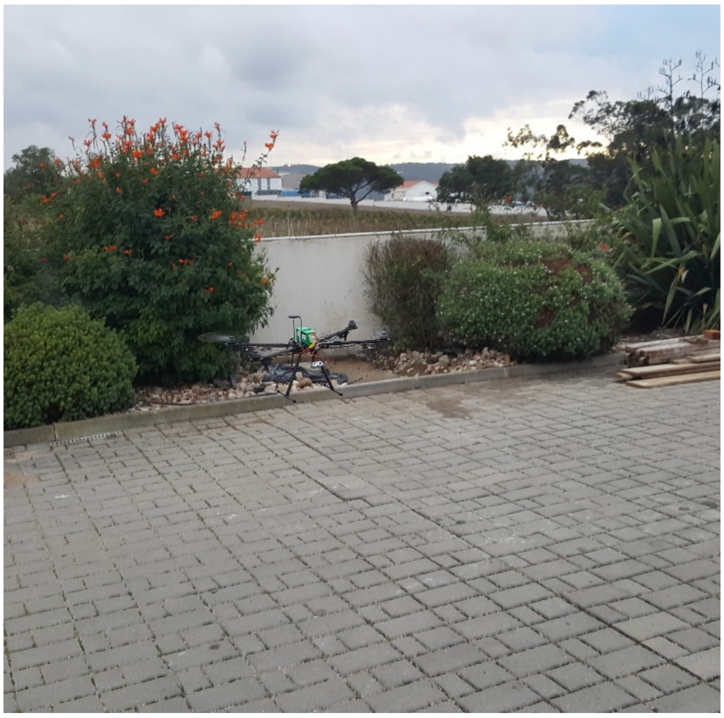


Figure 6.10 – UAV facing the “Impact zone”.

This test scenario was made by a human UAV pilot, using a RC controller. The figure 6.10 shows the “Impact zone” scenario facing the UAV. The chart represented by figure 6.11 shows the relevant LiDAR distance readings over time, on the environment presented on figure 6.10.

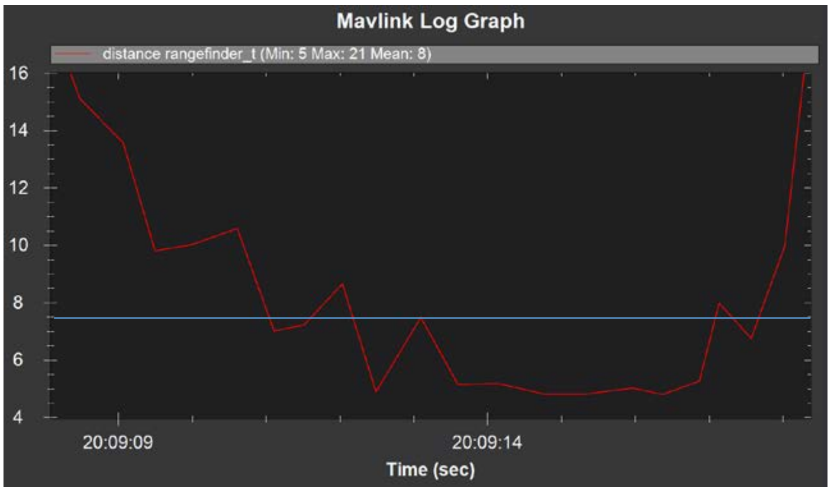


Figure 6.11 – LiDAR distance readings. Retrieved from Mission Planner software logs.

The test was made at a distance of approximately 16 m from the “Impact zone”, on a day with a wind speed of 19 km/h (Weather Underground, 2015). The “X” axis on the chart represents the local time when tests took place. The blue line represents the 7 m distance value. When approaching the wall, *i.e.*, UAV Pitch movement forward, some LiDAR reading “spikes” were found on this test being caused by object shapes facing the UAV. By analyzing the figure 6.11, the UAV is facing some bushes. The bushes’ shape and the moderate windy environment caused bushes’ leaves to move. Therefore, LiDAR light beams sometimes “hit” the bushes and sometimes “hit” the wall. The UAV behaviors were just as the ones performed on 3D environment. The only differences from 3D environment simulation is that RC PWM input value was not programmatically controlled and UAV took off in LOITER mode already. Analyzing the chart represented by figure 6.11, the UAV approached the wall at approximately 5 m distance being caused by drifting effect. The almost linear line on the 5 m distance value demonstrates the “Brake” mode trying to nullify the drifting effect. When compared with same experiment performed on 3D environment, on an outdoor scenario it took some time to “brake” UAV’s movement, since its linear speed was higher than the 3D environment tests. The increasing distance value corresponding from “20:09:17” time to the chart end represents the final “Brake” mode step: the symmetric Pitch movement (backwards), and the distance traveled caused by the backwards drifting effect. The chart below compares LiDAR readings in both 3D simulation and real outdoor scenarios. Since real outdoor experiment has a shorter time duration than 3D simulation, values were adjusted in order to compare graph curve.

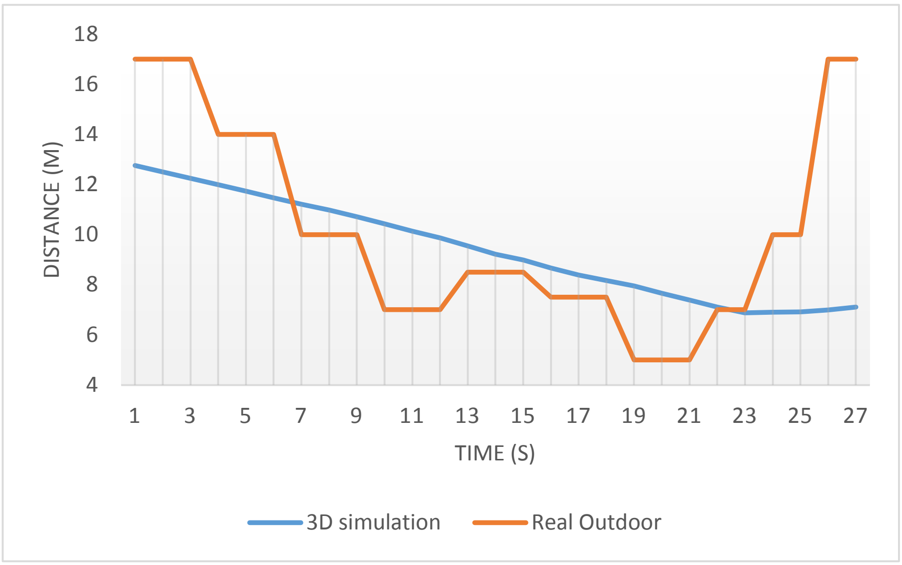


Figure 6.12 – Comparison between LiDAR readings on both test environments.

### 6.2.2 Avoid and Continue Mode

The “Avoid and Continue” mode was only tested in a 3D simulated environment. Tests on real outdoor environment were not performed for security reasons only. Since it was an urban environment, a slight non-expected behavior could cause damage on the objects around test area, including human beings, or on the UAV itself. This test was one of the successful tests made with “Avoid and Continue” mode.

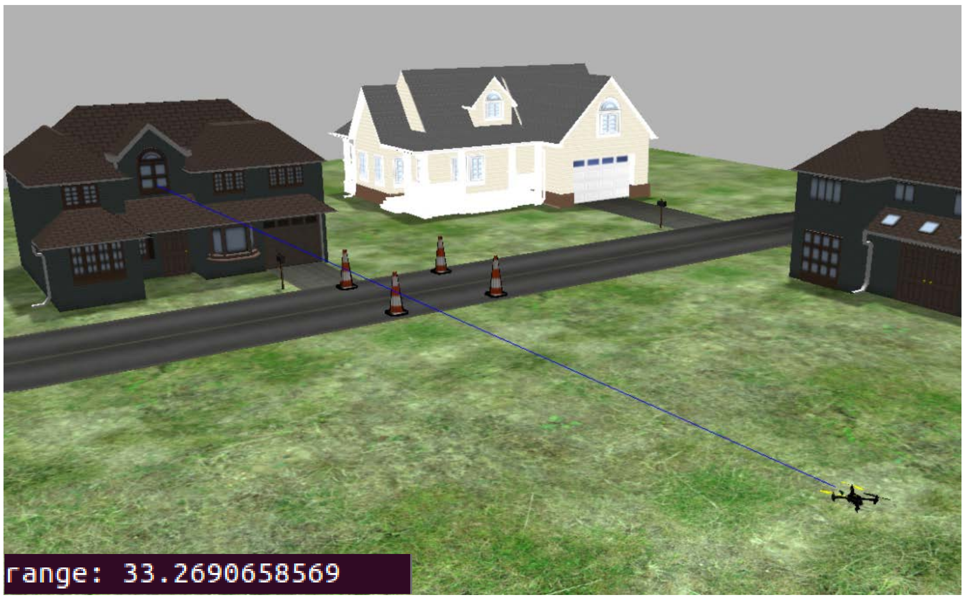


Figure 6.13 – UAV initial position in “Avoid and Continue” mode.

The figure 6.13 demonstrates the scenario used and the UAV initial position. The scenario contains multiple models, which aimed to simulate as much as possible a real outdoor environment. This “village” scenario is available as a “world” model of Gazebo model repository. The UAV started at GUIDED flight mode, hovering at 3 m altitude from the ground plane, and at a distance of approximately 33.27 m from brown house model.

In order to create waypoints, MAVProxy GCS was used and MAVProxy map module was launched including an image overlapping the map area. This feature allows the “link” between UAV and other object model GPS positions relatively to the GPS 3D simulated environment positions. With this system, it’s possible to design autonomous missions in MAVProxy GCS, using real GPS latitude and longitude values. The image was previously designed by the “village” model developer (at Gazebo’s model repository) and it represents a near-real objects scale when compared with a real outdoor environment. This means that, if the UAV is facing a certain model on the overlapping image, the UAV will be facing the same model on 3D simulated environment. The figure 6.14 demonstrates UAV position in MAVProxy 2D map.

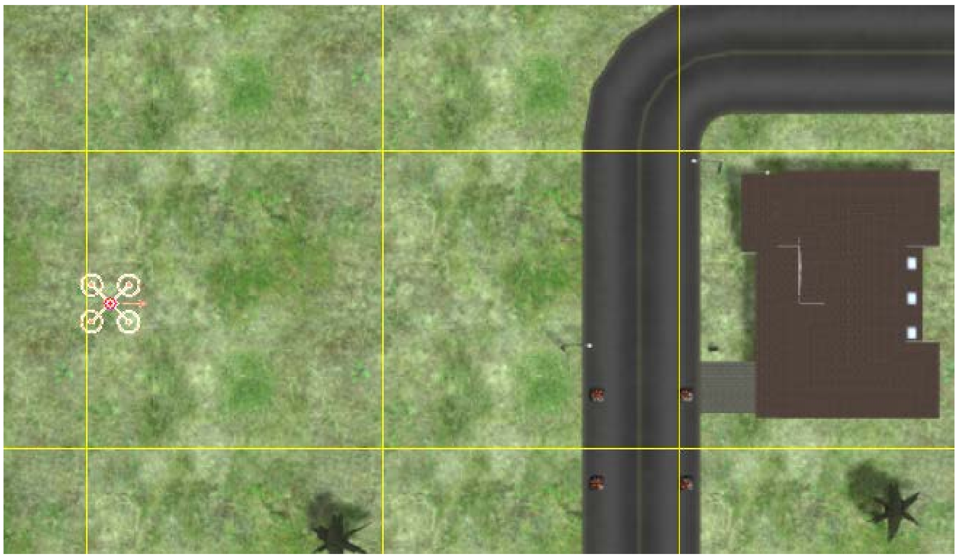


Figure 6.14 – MAVProxy map with top-view image overlap and UAV representation.

This image corresponds to the georeferenced top-view camera scenario of the Gazebo 3D simulated environment that corresponds to a part of the Gazebo 3D scenario plant.

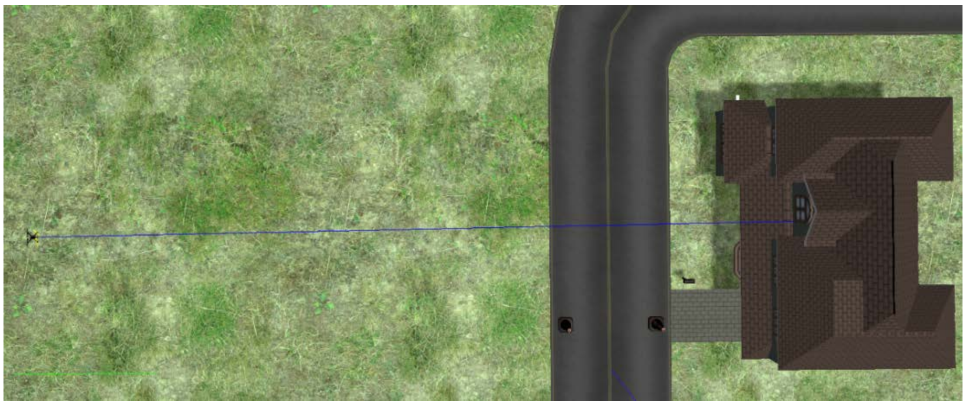


Figure 6.15 – Gazebo 3D UAV model facing the brown house model.

When comparing the two figures, the UAV’s heading and current location on figure 6.14 is almost the same as the one represented by figure 6.15. This allows an almost accurate waypoint mission navigation, *i.e.* if a mission waypoint is designed behind the brown house model in MAVProxy GCS overlapped image (as seen on figure 6.14), on 3D simulated environment, the

UAV should be able to fly to that mission waypoint location, which is in fact, behind the brown model in Gazebo.

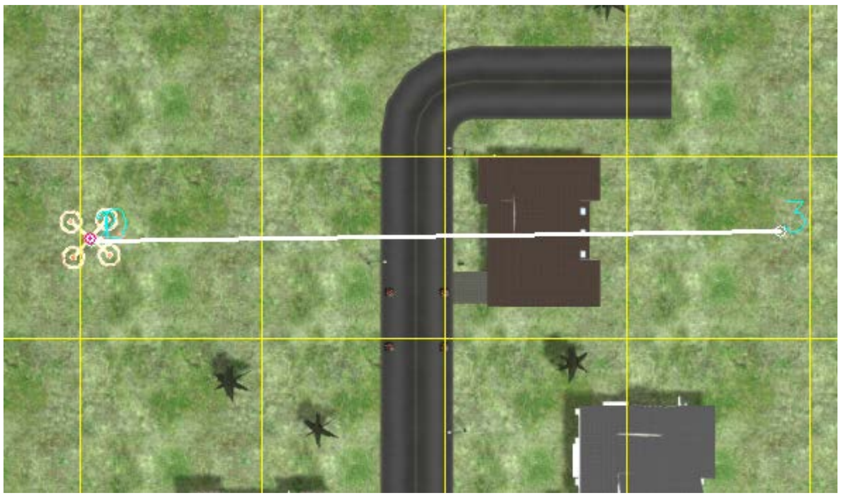
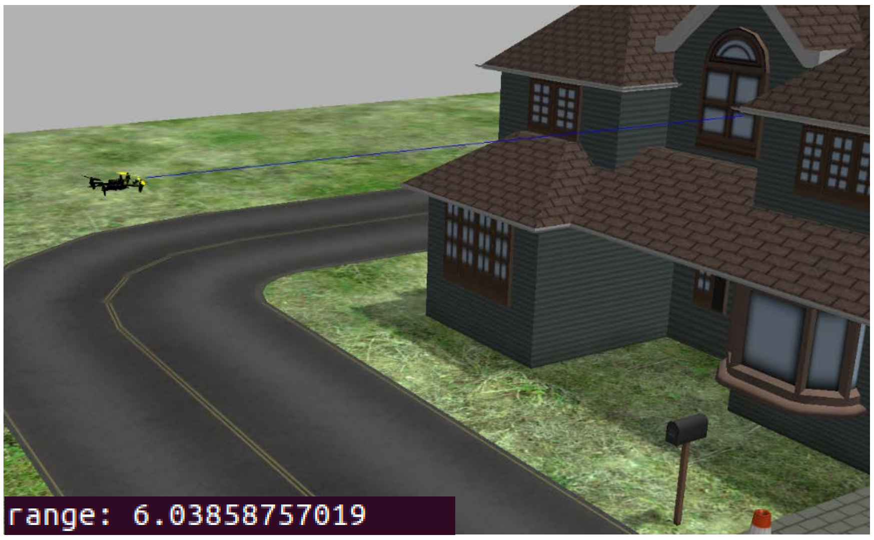


Figure 6.16 - MAVProxy mission path.

The figure 6.16 demonstrates the waypoint mission (caption numbers 0 to 3 presented on figure 6.16) designed on MAVProxy GCS using the map module ("--map" argument when launching MAVProxy). Every waypoint designed has a function called "Target altitude" which defines what altitude the UAV should have upon reaching that waypoint. For example, if a waypoint has a difference target altitude of 2 m from the previous one, the UAV will raise its altitude while flying to that waypoint in order to guarantee the target altitude set for that waypoint. In this case, all waypoints designed were set at a target altitude of 3 m.

Analyzing the figure 6.16, the UAV mission path has the brown house model in collision course, because the model has an altitude of approximately 4 m and the UAV is flying at a fixed altitude of 3 m.

In order to perform the established mission, the UAV must avoid collision with brown house model. "Avoid and Continue" mode is prepared to avoid collision on autonomous flight modes (Auto and Guided) by choosing a Roll direction to proceed the avoidance maneuver. The figure B.2 of Annex B represents the mission waypoint locations (Waypoint A is at UAV's starting position, and Waypoint B is behind the brown house model).



range: 6.03858757019

Figure 6.17 – UAV position at “Avoid and Continue” activation.

The figure 6.17 represents the UAV position and distance from brown house model before the Roll attitude. The “Avoid and Continue” mode was triggered at a distance of 7 m, but the drifting effect caused the UAV to be at a distance from brown house of approximately 6.04 m. By now, “Avoid and Continue” mode already saved the waypoints previously designed in order to proceed with the mission after object avoidance.

The UAV’s flight mode is changed to LOITER, to allow RC simulated inputs. Then, the UAV proceeds to Roll to a randomly chosen side direction for 1 s. This Roll attitude command is done by sending the MAVLink command “RC_CHANNELS_OVERRIDE”. In this command, the field name values (“chan1_raw” to “chan8_raw”) correspond to the RC channel PWM input values. In this case, the field “chan1_raw” was used to perform a RC simulated Roll movement (which corresponds to RC channel number 1). Its threshold value is 200 µs. This means that for left Roll attitude maneuvers, the RC input on channel 1 will be 1400 µs, and for right Roll attitude the value is 1600 µs. This threshold value was chosen as a test example and can be modified to adjust the Roll attitude speeds. In this test, the algorithm chose the left direction (RC PWM value of. 1400 µs).

If at Roll attitude end, the UAV is not at a higher distance than 10 m from the brown house, the “Avoid and Continue” mode will repeat the Roll attitude on the same direction until distance from object is higher than 10 m. If the Roll attitude lowers even more the distance from the object, then the “Avoid and Continue” mode will change the Roll direction, by performing the symmetric Roll attitude of the previous one. For example, if the right Roll attitude was chosen by the algorithm and the distance from object lowered, if the RC Roll attitude had a PWM value of 1400, the symmetric Roll attitude will be 1600 μs. This follows the principle explained in the "Brake" mode (subsection 6.2.1), but in that case, it was for reducing UAV drifting effects. In this test, there were two Roll attitude maneuvers. The figures 6.18 and 6.19 show that behaviors.

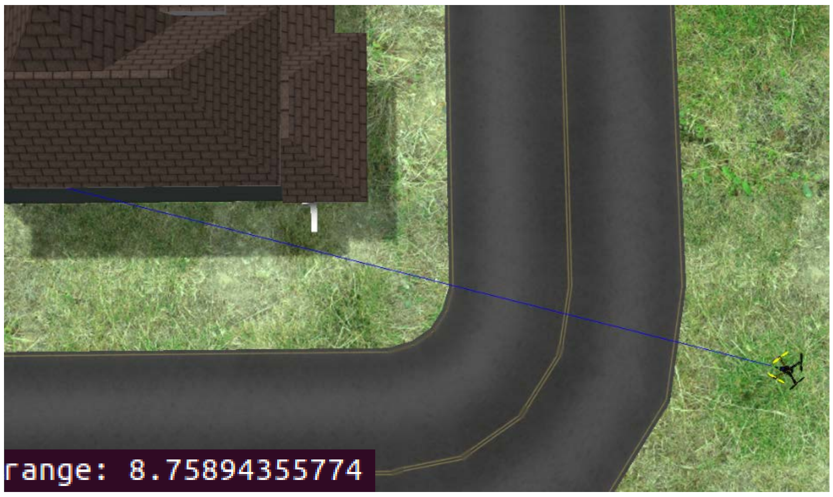


Figure 6.18 – UAV position and distance measured when 1st Roll attitude ended.

The figure 6.18 represents the UAV position at the first Roll attitude end. Since it's not at a 10 m distance yet (8.76 m approximately), a new Roll attitude maneuver is needed in order to continue mission safely and with a object-free path.

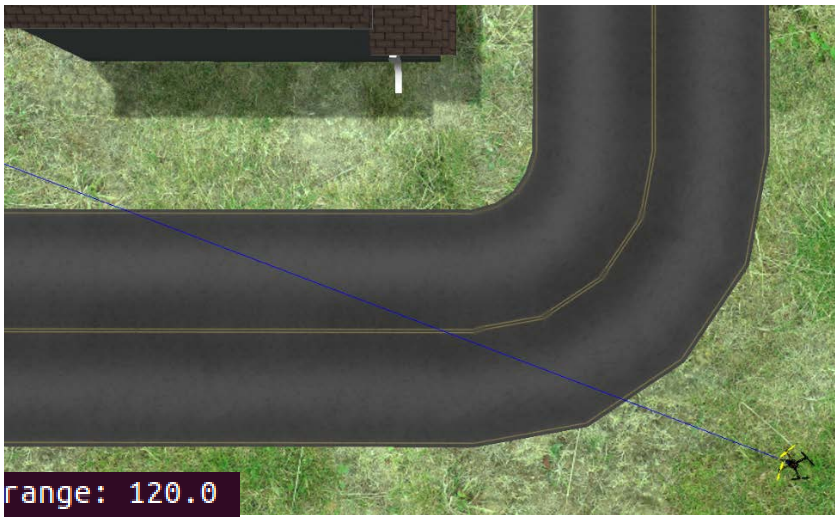


Figure 6.19 – UAV position and distance measures when 2nd Roll attitude ended.

The figure 6.19 represents the new UAV location and distance at the second Roll attitude end. The virtual LiDAR is measuring a distance of 120 m. This means that no object is in sight. For that reason, the Gazebo 3D simulator presents a distance of 120 m which corresponds to the maximum distance measurement previously configured, which also corresponds to the maximum distance measurement of SF11/C LiDAR. After the second Roll attitude, the UAV is not facing the brown house anymore and it can proceed with the mission safely. The “Avoid and Continue” mode, which previously saved the waypoint behind the house model, will change to Guided flight mode again and proceeds to fly to the mission waypoint.

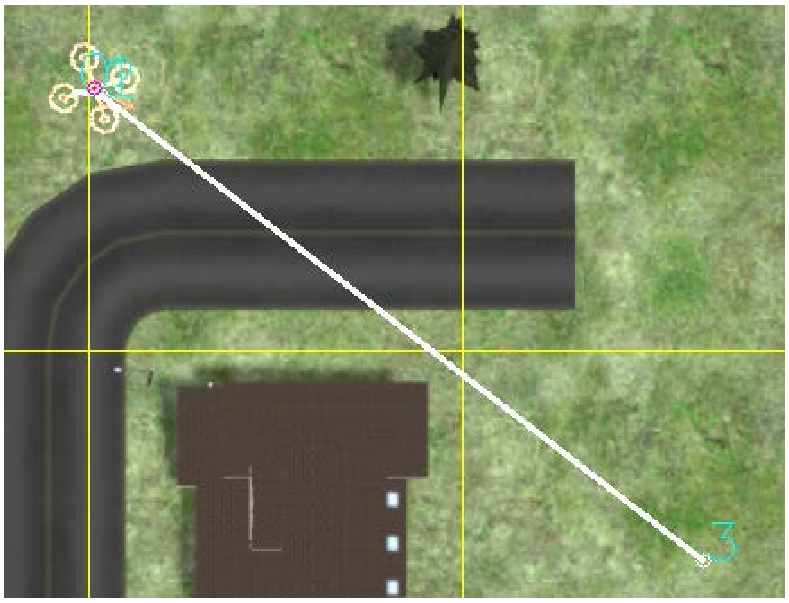


Figure 6.20 – MAVProxy new mission path to waypoint B.

The figure 6.20 represents the new mission path generated by “Avoid and Continue” mode. For the UAV flight’s controller, the new mission generated is not related to the previous one. This means, for Pixhawk’s flight controller, the UAV performed two missions independently, but technically it only performed one with avoidance collision maneuvers. The figure 6.21 represents the UAV position at mission end, concluding this test and proving this mode’s main goal: avoiding objects in collision course and continue previous waypoints until mission end.

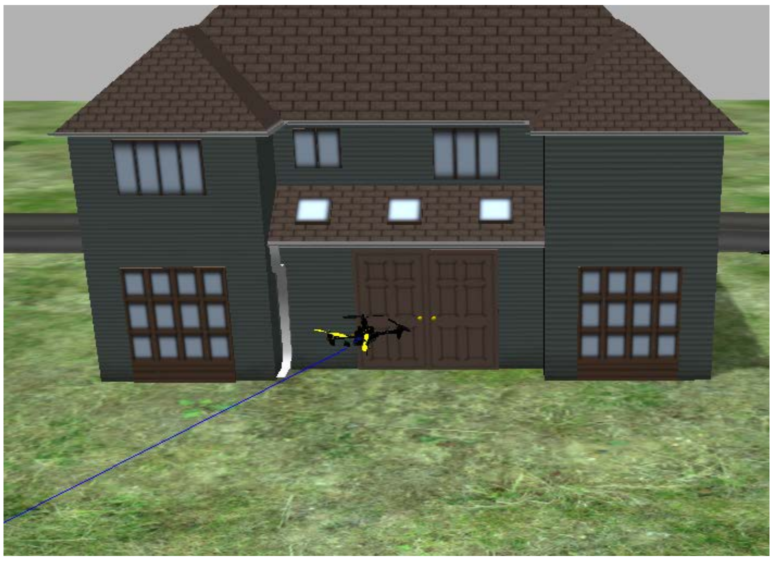


Figure 6.21 – UAV autonomous mission end.

The perfect solution was to fly above the brown house, but since LiDAR cannot sweep the environment (by forming a vertical arc with a certain azimuth angle), it is not possible to detect the objects height in order to determine the best avoidance maneuver. A simple Throttle attitude, by raising the UAV's current altitude would had solved the problem, in this case. The "Avoid and Continue" was developed for city environment purposes, where buildings are tall enough to discard the "flying above the object" solution. This was the reason why the Roll attitude was considered in "Avoid and Continue" mode.

This mode performance was like expected but it requires some improvements. If the UAV encounters a concave wall, it will be trapped forever and occasionally it will cause the UAV to crash. The Roll attitude behaviors can cause the collision with side objects, since LiDAR sensor is pointed to the front and has no visibility to the other sides. Also on Roll attitude behaviors, the path chosen can be the farthest one to an object-free area being caused by the limitations of the LiDAR sensor used. These problems can be solved adding a system that analyzes the environment in every 3D direction. With that system, the UAV could climb the concave wall (if the height justified that behavior) or chose the shortest path to a clearer, object-free area.

# Chapter 7
# CONCLUSIONS AND FUTURE WORK

In this chapter, the main conclusions and possible work extensions are presented.

## 7.1 Conclusions

The main goal of this dissertation was to develop a system that was capable of taking control of an UAV autonomously if an object in collision course was detected. Therefore, “Sense and Avoid” algorithm, considering “Brake” and “Avoid and Continue” modes, was developed to avoid UAV collisions and to ensure autonomous object collision avoidance maneuvers. The algorithm was tested in two different test environments: a 3D simulation with realistic physics and collision simulations, and a real outdoor environment. A custom-built UAV was made in order to test algorithm on a real aircraft. All the work and research done, and after test results and performance analysis, some conclusions were made:

- The LiDAR sensor proved to be a reliable distance measurement sensor. The accuracy of SF11/C LiDAR was outstanding and it was the best solution to detect with precision objects in collision course;
- The “Brake” mode of “Sense and Avoid” algorithm worked flawlessly after a long number of trial-and-error tests. The successful results on a 3D simulated environment proved that same behavior should be expected on a real outdoor experiment. The UAV maneuvers were as expected on real outdoor environment.
- The “Avoid and Continue” mode proved to be useful feature for every UAV on autonomous missions. Unfortunately, its results showed some flaws caused by SF11/C LiDAR’s capabilities and limitations. The main reason of this is because the algorithm was developed and adjusted according to the LiDAR capabilities. “Avoid and Continue” mode potential should be expanded by improving current system (for example, a 360º LiDAR environment sweep) or implementing another distance sensing system.

## 7.2 Future Work

By following this dissertation research, some improvements can be made:

- Implement a system that can sweep an entire environment. With this, an UAV can have total control on objects around it, improving the decisions and response behaviors of “Avoid and Continue” mode.
- Distance sensor diversity (hybrid solution using LiDAR, SoNAR and RaDAR) could be added to an UAV. This highly improves the object sensing by using a chosen technology according to the work environment or purpose.