ABSTRACT

Currently the use of Unmanned Aerial Vehicles (UAVs) for civilian purposes is growing exponentially, whereas in the past it was a technology used only for military objectives.

There is an array of applications that can be accomplished with the use of this technology, such as making aerial films on a low budget, coast surveillance, parcel deliverance, and many other applications. However, in a civilian environment, these UAVs systems are typically controlled with a line-of-sight (LOS) technology such as Radio Control (RC), which limits the range that the system can be operated. Therefore, the main objective of this thesis is to study the use of mobile wireless networks as a viable replacement of RC control, to have a full control of the vehicle, providing then an extended mobility to the operator and vehicle.

To accomplish this an Unmanned Vehicle System (UVS) was developed in order to monitor and provide the user with the ability to control several drones, such as UAVs and Unmanned Ground Vehicles (UGVs), using mobile networks. The user can operate the vehicle using a Remote Ground Station (RGS) that provides full control of the vehicle, with an android mobile application making it accessible, with no associable costs. On the vehicle a Raspberry PI (RPI) is attached, that receives and handles the information to control the drone and provides the user with a live stream that allowing sight from the Point of View (POV) of the drone, enabling responsibility to the RPI to communicate between the drone and the user.

With this approach the system is completely independent from the RC technology, allowing the user to control the vehicle without the need of physical presence on the side, creating a vast new array of uses for this kind of technology.

***Keywords*:** Multi Drones, Wireless Networks, Mobile Devices, Drones

RESUMO

Hoje em dia o uso de Unmanned Aerial Vehicles (UAVs) para fins civis esta em crescimento exponencial, no passado era uma tecnologia exclusiva para fins militares.

Existe uma variedade de aplicações que podem ser realizadas com o uso dessa tecnologia, tais como: fazer filmagens aéreas, com um orçamento baixo, vigilância de costa, entregar encomendas, e muitas outras aplicações. No entanto, estes sistemas UAVs são geralmente controladas com uma tecnologia line-of-sight (LOS), como controlo de rádio (RC), que limita o intervalo que o sistema pode ser operado. Portanto, o objetivo principal desta tese é estudar o uso de redes móveis para substituição do radio controlo para o controlo total do veiculo, proporcionando mobilidade extra para o operador e veículo.

Para realizá-lo foi desenvolvido uma Unnamed Vehicle System (UVS) que monitoraria e fornece ao utilizador a capacidade de controlar vários drones, tais como: UAVs e Unmanned Ground Vehicles (UGVs), utilizando redes móveis. O utilizador pode operar o veículo da Remote Ground Station (RGS), que fornece o controlo total do veículo, é uma aplicação Android tornando assim acessível sem qualquer outro custo. No veículo está presente um Raspberry PI (RPI) que recebe e trata a informação para controlar o avião e fornece ao utilizador uma transmissão de vídeo ao vivo que permite ver a partir do ponto de vista do Drone (POV), tornando o RPI o responsável por fazer a comunicação entre o veículo e o utilizador.

Com esta abordagem, o sistema é completamente independente da tecnologia de RC, que permite ao utilizador controlar o veículo sem a necessidade da presença física no local, o que cria de novas aplicações para esta tecnologia.

Palavras-chave: Multi Drones, Redes Sem fios, Dispositivos Móveis, Drones

Chapter 1
INTRODUCTION

In this Chapter, it has defined the concepts behind the dissertation, what the motivation to do this work, the objectives to be achieved, and the state of the art of this technology field.

1.1 Overview

In the beginning, the UAVs were vehicles mainly controlled by RC, but nowadays there is a demand for using multiple control systems such as the auto-pilot with pre-planned flights the UAV's, which are capable of flying alone, using a previously planned course.

The UAV's are characterised by their weigh and by the maximum altitude ~~that~~ they can reach. There ~~is~~ are various types of UAV's: Micro, Mini, Close-Range for civilian and other purposes and Tactical, Medium Altitude, Height Altitude Long Endurance for military purposes because of the size and the cost that they demand.**Error! Reference source not found.**

The type of UAV that we are going to focus on is the Micro Aerial Vehicle (MAV). There ~~is~~ are some different types of MAV such as multi motors that allow~~s~~ the vertical landing and take-off, flight stability and have a better load capacity. However, because of the multi motors energy waste, a single motor with the same features as the multi motor type, except for the loading capacity, composes the helicopter. And finally, the fixed-wings ~~has~~ have the lift force horizontally so just allows us to land and take-off on that same direction, but they cannot stabilize on the air, although they use less energy and by doing so, they become more appropriated for long distance missions.**Error! Reference source not found.**

A multi motor is composed by: propellers, motors, Electronic Speed Controllers (ESCs), a flight controller and a RC receptor. The UAV has several sensors such as gyroscope, an accelerometer, a barometer, a magnetometer and a Global Position System (GPS); all of this sensor are known as Inertial Measurement Unit (IMU), and through the Inertial Navigation System (INS) controls the UAV to land and take-off with the data from the IMU without need of external data. **Error! Reference source not found.**

For communication with a UAV beyond line-of-sight, it is typical to use satellite communication but, because of the expensive hardware, it is only used for military purposes. For that reason, the RC communication is adopted for civilian purposes, which delimitates the user to line-of-sight communication; so, to overtake this problem, it is intended to build a mobile network system as alternative.

1.2 Motivation

The development of UAV technology for civilian proposes has been increasing in the past years, which allows us to have ~~a~~ cheaper and easier access to it.

There are numerous ways to use a UAV and currently 89% of the investment is applied in military market. There is a great impact on this area, because it allows to flight in remote and dangerous areas without having a human being putting his life in risk **Error! Reference source not found.**. There are more areas where the UAVs also have a great impact such as surveillance. For example to protect the coastline, to prevent entry of illegal immigrants or drug smuggling; it can be used to prevent forest fires **Error! Reference source not found.**. And it can be used to save someone from drowning, thus instead of having a lifeguard swimming to the person, he/she can stay on the beach controlling the UAV to reach and save the person, avoiding panic attacks from the drowning person.

Big companies such as Facebook, Google and Amazon are showing interest on this technology. For example, Amazon recently bought a company to expand their methods of delivering and their intention are to make UAVs to deliver packages. Google, on the other hand, is more interested in Unmanned Ground Vehicles, for self-driving cars, a way of keeping Google maps Street View updated and other purposes. The Facebook way is to provide wireless Internet using UAV to expand to areas where internet is only reachable by satellite, making this option much more valuable, allows cheaper prices **Error! Reference source not found.**.

To control UAVs it is important to have data and video transmission, which allows remote controllers to control the UAV. The easy and main controller for civilian drones is using the Radio Control, which is a cheap technology but with a big inconvenience: it just allows the controller to controller the UAV just till the line of view. Military drones use~~s~~ satellites to communicate, but it's an expensive technology, impracticable in civilian purposes, so we have to think in a different approach, and the idea is using mobile communications, which allow~~s~~ to control a UAV without watching it, just by viewing the UAV trough a video stream. Although, with this great solution comes some disadvantages, because now we have to worry about latency, loss of coverage and bit rate, etc.

And to take full advantage of mobility, the best way of controlling the UAV is through a mobile device like a smartphone or tablet, which brings us 2 ways of communication: via Wi-

Fi or via 3G/4G; this allows us to have one system indifferent from the location of both of the intervenient.

1.3 State of art

In terms of civilian use, companies have developed hardware/software projects for UASs, not only vehicles construction, but also flight control systems, that are responsible for the stabilization and provides the waypoint navigation, which allows the UAV to flight himself.

1.3.1 UAV

Unmanned aerial vehicles (UAVS) are aircrafts controlled by a ground control station, which can receive commands with no need for a real pilot in the vehicle; or a vehicle that can perform a pre-programmed mission.

Since the creation of the UAV technologies, there has been created many civilian and military applications, such us surveillance, photography, aerial reconnaissance, scientific research, logistics and transportation, etc.

1.3.2 Raspberry PI

Raspberry Pi is a cheap computer, the size of a credit card, developed to help the young kids and older people to learn how to work with computers; because of cheap price, it can reach countries where it is expensive do to have a computer, making the raspberry Pi affordable for everybody.**Error! Reference source not found.**

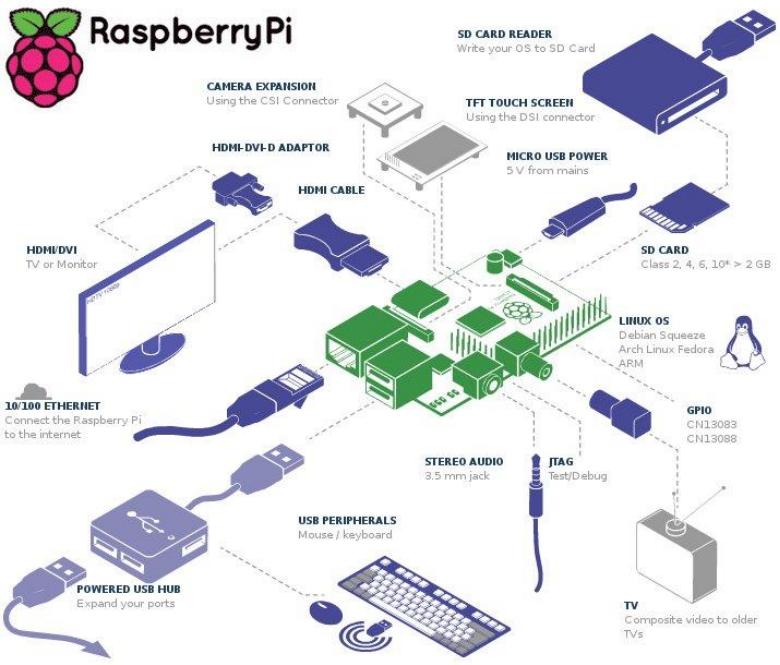

Figure 1-1 *- Structure of RPI and all is IO components. Source:* ***Error! Reference source not found.***

RPI can be used for many things, such as a personal computer, but because of its speed limitations, it may be a bit slow. It can also be used as a media centre: instead of having an expensive smart TV, we can use a RPI as a smart device and connect it to the TV. We can also use RPI to programme, do electronics etc.

It is a powerful tool and it is relatively cheap, so that's why its appropriated to use it as the data receiver on the UAV, and the communication with the Pixhawk, the Open Source Project (OSP), it is relatively easy because it is connected with a BUS cable to transfer the data between the components, which eliminates the latency problem.**Error! Reference source not found.**

One of the IO modules that has been utilised is the Raspberry Camera, with the help of a streaming script. Together these stream the video to the Remote GCS, allowing the user to see the drone's movements. **Error! Reference source not found.**

1.3.3 Flight Controllers

There are many OSPs; however, this project was only focused on two types: Ardupilot and Pixhawk

1.3.3.1 Ardupilot

Ardupilot is an open source project based on Arduino that supports several types of UAV's such as fixed wings, hexacopters etc. To control multi motors it uses the Arducopter firmware. To help on the controlling Ardupilot has server sensors embedded. **Error! Reference source not found.**

1.3.3.2 Pixhawk

Pixhawk is a platform with proposes to implement the control of the MAV from a computer. This platform is built with the autopilot PX4 that is the firmware that supports the Pixhawk and Arducopter. O flight controller PX4 is the combination of the FMU and IO boards, and executes the corresponding firmware to each type of UAV. Pixhawk supports, fixed-wings, multi motors, helicopters, boats, cars, etc. **Error! Reference source not found.**

1.3.4 Communication

To make possible to control this type of vehicles, it is necessary a way of communication, to send and receive telemetry info, controlling commands, and video; there is an array of options that we can choose from such as radio, satellite, cellular networks, etc.

1.3.4.1 Radio

To have a RC communication it is necessary that the user and the controller both have a transmitter and receiver radio system. It is a cheap technology but there is an inconvenience while using this type of communication: the user can only control the vehicle on the line of view, far from that the communication becomes impracticable because the power it is not enough.

1.3.4.2 Satellite

The communication with satellite is complex and expensive; therefore using the raspberry pi allows a cheap and easy way of using this communication, through the RockBLOCK **Error! Reference source not found.**. However, there are some complications with this communication; it only allows the upload of 340 bytes and the download of 270 bytes. In the project, there is no problem with the MAVlink messages, that have the maximum size of 263 bytes **Error! Reference source not found.** but if we talk about the video stream, there is a big problem, because there is no space to send any message. Another problem is the latency problem, with the clear sky, sending a message can take at least 20 seconds, whilst with a restrict view, it can take several minutes, which makes the communication really complicated.

1.3.4.3 Cellular Networks

The national 3G/4G existing infrastructure provides the possibility to use this type of communication and implement to the UAVs and Ground Control Station (GCS). With the actual developing of 3G, its reduced latency and the developing of Long-Term Evolution (LTE) and LTE advanced increased the viability to use this type of network to real time application. However, there are some problems with the connection with this type of network: in P2P connection between two devices on the internet. These problems are related with using of NAT, that is the external interface with a public IP and the private IP from the internal network which all the network users transmit data. When a device connects with the network normally it is on the NAT network with a private IP, this is practical because the lack of

public IP address, but creates a problem, that does not allow the user to connect with an external network from his private network because he does not know the external IP address.

1.3.4.4 MAVLink

MAVLink is a protocol with a header-only message that defines a group of messages to transmit data between the UAV and GCS. The OPS Ardupilot and Pixhawk use it. Therefore, is the protocol that we are going to use. **Error! Reference source not found.**

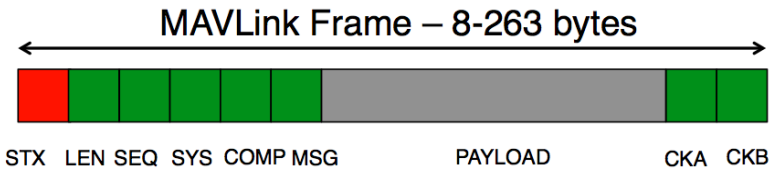

Figure 1-2 *- Mavlink packet structure. Source:* ***Error! Reference source not found.***

1.4 Objectives

The main objective of this dissertation is to build a monitoring and control system to a UAV using mobile networks such as 3G/4G and satellites.

System can be described by 3 topics:

1. Remote Control: Developing of an Android application for a tablet to collect all the information, such video and telemetry.
2. Access Port: Located on the UAV, and it is the connection point between the remote control and the Ardupilot (Board who controls the UAV) and it will use a Raspberry PI as a communicator and it will be the bridge between the Ardupilot and the Remote Control.
3. Connection Control: Manage the connections between the Access Port and the Remote control, it will be necessary to overtake the difficulties of the P2P (Peer-to-Peer) networks about the internet, to the effect it is common to use NAT (Network Address Translation) protocol, which allows us to transform a private IP (Internet Protocol) into a public one.

1.5 Contributions

This dissertation proposes to develop an application for controlling and monitoring multiple drones using wires networks. The main goal of the app is to have a full control of a drone; however, the user can preprogram a waypoints mission. A Raspberry Pi was integrated on the system and two scripts were developed, a telemetry script write in python that sends and receives all the telemetry, and the stream script, through the Raspberry Camera.

To communicate and overtake network problems, a server allowing all the apps to communicate with each other was developed.

To show to the scientific community an article about the system was written. [14]

1.6 Dissertation structure

The objective of this section is to present the dissertation structure. The dissertation is divided in seven chapters:

Chapter 1 describes the introduction, within the motivation to approach the problem and the dissertation objective. It also describes the state of the art.

Chapter 2 describes the main structure of a UAV, with the components description, and it will be explain the applications where the UAV can perform there function.

Chapter 3 describes the drone brain, how it is composed, and all the main components that make all the system works.

Chapter 4 describes three different applications that were developed: Remote GCS, Server and Raspberry Pi.

Chapter 5 explains the communication system and how it must be improved as much as possible to provide real-time and reliable operation of the UAV.

Chapter 6 shows results that were collected and how are they related with each other.

Chapter 7 presents what conclusions can be taken after all the developed work, and what improvements can be done in the future.

Chapter 2
UNMANNED AIR SYSTEMS

Description of a UAS structure and applications

2.1 UAS Overview

An Unmanned Aircraft System usually has the same composition of a normal manned aircraft; however, the design is made in function of a control without the presence of an aircrew. Unmanned System comprises a number of sub-systems such as Unmanned Aircraft; Ground Control Station; Payload; Lunch and Recovery; Navigation Systems; and Communication.

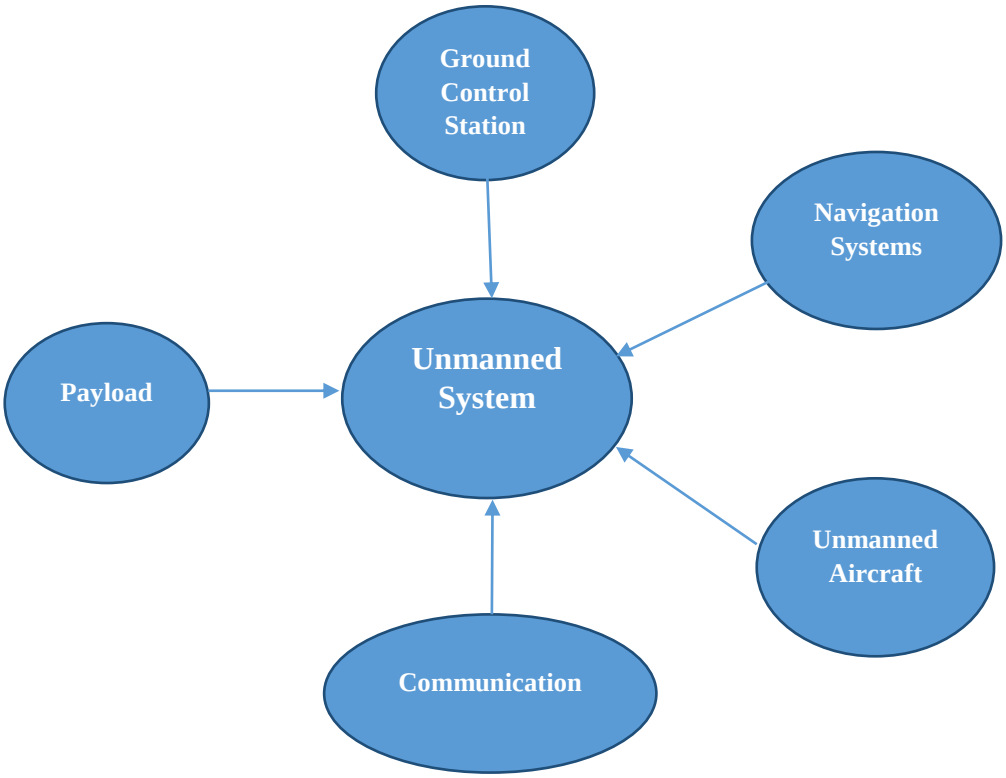

***Figure 2-1**- Unmanned System Sub-systems*

All the elements or sub-systems, presented in the ***Figure 2-1***, are built with the purpose of working together in order to get the desired objective. However, some of the sub-system can work individually in other uses, but when together with the other sub-systems, they must be able to operate together.

For example, the Radio communications sub-system is one individual interface between the GCS and the UAV, the element must be installed on both vehicle and the GCS, and both must operate with the same configurations, for example, they have to use the same protocols in the same wavelength in order to have a communication.

2.1.1 Unmanned Aircraft

Three types of categories sort unmanned Aircrafts: High Altitude, Medium Altitude and Low Altitude. High and Medium Altitude aircrafts have a fixed-wing structure; the Low Altitude Aircrafts have a rotary-wing structure, such as a helicopter or a multi-rotor.

Category	Mass [kg]	Altitude [m]	Autonomy	Range [km]	Structure	Operator
HALE	450-13500	> 20000	Days	Trans global	Fixed-Wing	Military
MALE	450-13500	9000	< day	500	Fixed-Wing	Military
TUAV	15-450	5000	< day	100-300	Fixed-Wing Rotary-Wing	Military
Mini UAV	<20	3000	< hour	<30	Fixed-Wing Rotary-Wing	Military/ Civilian
Micro UAV	<2	< 1500	< hour	<20	Fixed-Wing Rotary-Wing	Military/ Civilian
Nano UAV	<0.025	< 100	Minutes	Short range	Fixed-Wing Rotary-Wing	Military

Table 2-1**- Unmanned Aircrafts Vehicles Categories and characteristics* ***Error! Reference source not found.Error! Reference source not found.

Table 2-1 represents the characteristics of each UAV category, sorted by Altitude and endurance. The first three categories, High-altitude Long-Endurance (HALE), Medium-Altitude Long-Endurance (MALE) and Tactical Unmanned Aerial Vehicle (TUAV) are used only for military purposes, because of the scale and the cost.

For civilian purposes, it is more common the use of MAVs because they are typically cheaper and have a simple structure, which allows an easy transport and launch which makes MAVs versatile.

***Figure 2-2** - Overview of the Hexacopter structure*

A MAV (Mini UAV) with a Rotary-Wing structure was developed in this dissertation. This kind of structure has the advantages of allowing a Vertical Take-off and Land (VTOL). ***Figure 2-3*** illustrates the developed vehicle presented on this dissertation.

As we can see in ***Figure 2-3*** the UAV, structure is composed by six components: frame, battery, ESC, gimbal, Motor and propellers and finally the Electronic Processing System where were included the Pixhawk and the Raspberry PI - it will be explained with more detail on ***Chapter 3.***

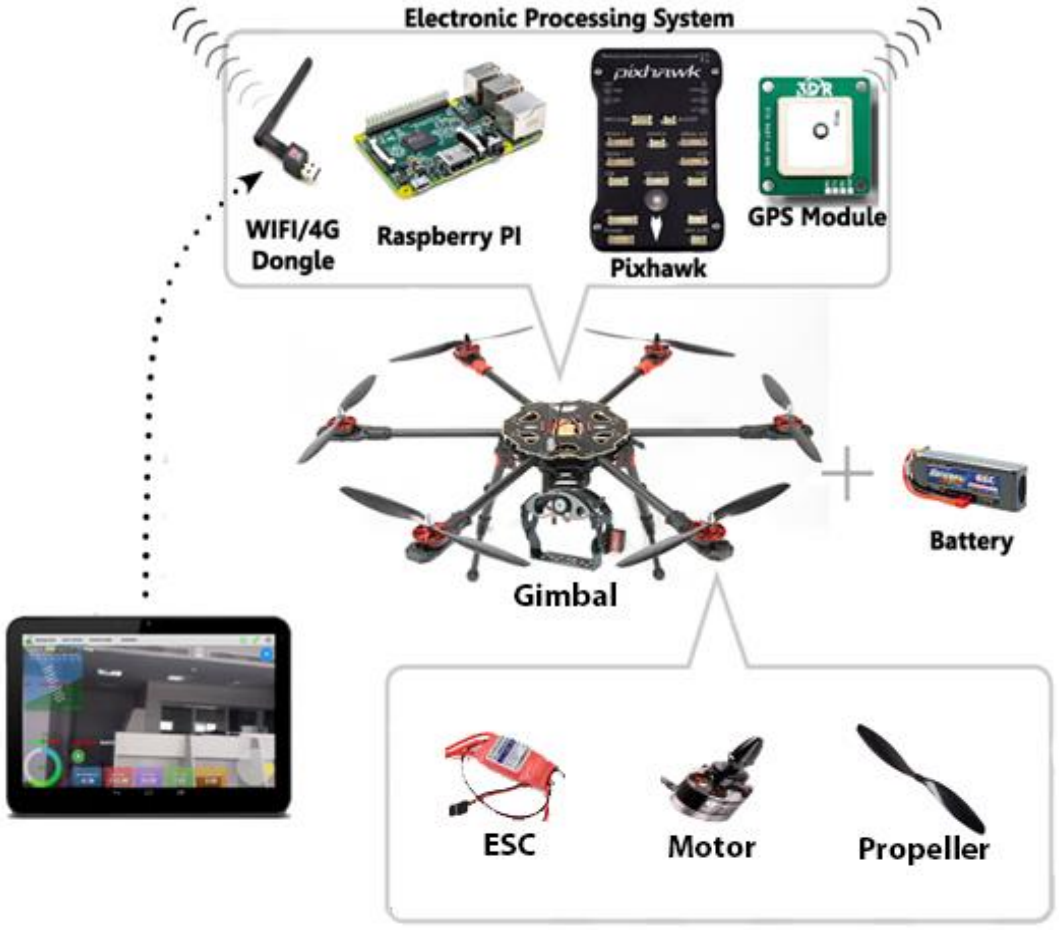

Figure 2-3 - *UAV block structure*

2.1.1.1 Frame

Vertical Take-Off and Landing (VTOL) vehicles have a wide range of possible configurations, from one rotor to multi-rotor. Helicopters use single rotor configurations. Within multi-rotor configurations, it is possible to have from three to eight rotors, *i.e.* from tri-copters to octo-copters (with the possibility to have co-axial rotors). The number of rotors is an important factor when choosing the frame configuration: more rotors leads to higher energy consumption but also leads to a higher stability during the flight. A configuration with co-axial rotors allows carrying more payload weight without a larger frame. The Frame is the vehicle's chassis and is composed by the arms, landing gears, motors mounts, centre plates and camera gimbal (which is optional). It is the frame's responsibility to support the payload as well as all on-board components (motors, ESCs, flight controller and sensors).

For this project, a hexa-copter configuration was used, as the one represented in ***Figure 2-4.***

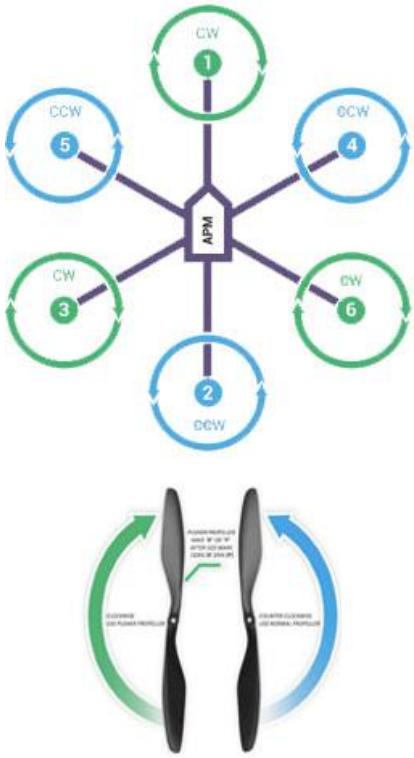

Figure 2-4 *- UAV Frame Hexa+*
*Source:****Error! Reference source not found.***

2.1.1.2 Battery

The first thing that we have to be careful about is the voltage level that UAV system runs at. To increase the overall voltage, cells are connected in serial. Connecting the cells in serial, cumulatively adds to the overall voltage. For example, two 3.7 V cells will output 7.4 volts and a three cells will output 11.1 volts and so on.

2.1.1.3 Electronic Speed Control

The electronic interface between the motor and the flight controller is responsible for the speed control of the motor. ESCs have a current limit and they cannot cross that limit. The more power an ESC can handle the larger and heavier and more expensive the ESC has to be.

2.1.1.4 Gimbal

The gimbal is an optional part of the frame where a camera can be sited. This mount allows the camera to move in diverse angles, giving a great variety of shooting and filming angles. Normally, the gimbals make use of servo motors to do those movements. Thus, the referred angles will depend on the servo used.

2.1.1.5 Motor and Propellers

The motors can be divided in two types: outrunner and inrruner ***Table 2-2***. Its classification is made considering where the rotation shaft is relatively to the magnets: if it is on the outside, so it is an outrunner; else if it is inside it is an inrunner.

Motor type	RPM	Torque
Outrunner	Low	High
Inrunner	High	Low

Table 2-2 *- Motors Types: Outrunner vs Inrunner*

The motor and the propellers can be seen as a set, since the motors size depends on propellers size. Choosing the right propeller is one of the most important tasks when building an UAV, since they are closely related to UAV stability, speed and efficiency. If the propeller is too

large for the motor, then it will struggle to spin the propeller; else, if it is too small for the motor, then the motor will contribute to the overall weight.

There are many kinds of propellers. The ones with multiple blades are less noisy and capable to meet higher power requirements but are less efficient.

2.1.1.6 Handling

The UAV can move in 3 different axes: the Z-axis, represented in ***Figure 2-5*** with the up/down movement, the Y-axis, represented by the roll movement which is the UAV left and right movement. The X-axis, represented by the pitch, which is the movement along the axis from the front to the back of the UAV, and finally the yaw movement, which is the rotation of the UAV with respect to the centre axis, can rotate clockwise or counter clockwise.

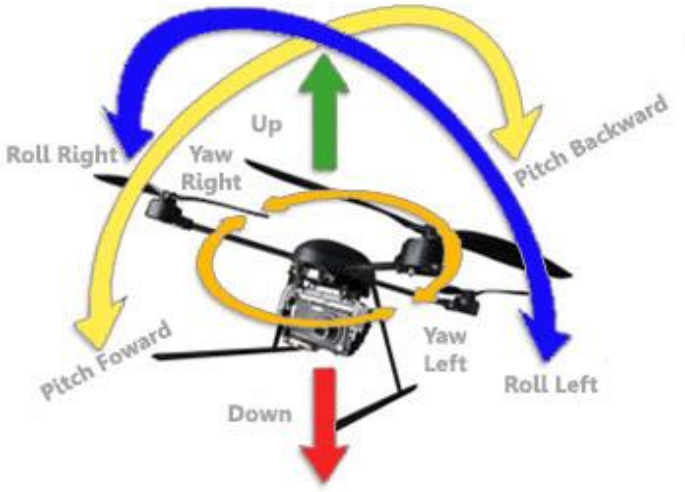

Figure 2-5 - *UAV orientations.*

2.1.2 Navigation Systems

The operators need to know, on demand and at any given moment in time, the position of the aircraft. In addition, the aircraft may need to "know" if autonomous flight is required on any trail of its flight path. This logic is either part of a pre-programmed mission or an emergency or loss of connectivity "return to home" operation function. For full autonomous operation without the need of an operator, the aircraft needs to carry sufficient navigation equipment. GPS systems nowadays are extremely lightweight and cheap, and are able to give continuous position updates so that only a very simple form of INS is normally needed. The accuracy is further improved by the use of differential GPS.

2.1.3 Ground Control Station

The control station is the control center of the operation and the man-machine interface. It is usually situated at the ground where the UV mission is usually pre-planned, in which case the control station is known as mission planning and control station. The mission can also be planned from a central command centre and sent to the control station for execution.

At the control station, the operator commands the aircraft via the system up-link in order to instruct its flight profile and to operate different types of mission payload that it carries. The aircraft also uses the up-link to return information to the operator. This information may include data from the payloads, status information, images, etc... Launching and recovery of the aircraft may be controlled from the control station.

2.1.4 Communications

Most demanding and principal requirement is to provide communication data links between the control station and aircraft. Data link is the term used to describe how the UAS commands and controls information. The link is usually radio frequency but alternatives include light in the form of laser beam or via optical fiber.

The uplink from the control station to the aircraft transmits the flight path to the aircraft and stored in its automatic flight control system. The uplink can transmit real-time control commands to the aircraft when man-in-the-loop is needed and transmit control commands to the aircraft mounted payloads and ancillaries. In addition, the link transmits updates on position information to the aircraft when relevant. Additionally downlink from the aircraft to the control station transmits payload imagery, housekeeping data, battery level, engine temperature, etc.

UAS operations can be divided into radio frequency line-of-sight and beyond line-of-sight. Line-of-sight operations refer to operating the UA via direct radio frequency. Beyond line-of-sight, operations refer to operating the UA via a wireless communication link such as 3G, 4G, Wi-Fi or satellite.

Figure 2-6 *- Communications Links*

2.1.5 Payload

The operation task defines the type and performance of the payloads. These can range from:

Relatively simple sub-system, which consists of a lightweight non-stabilized video camera with a fixed lens, weighing as little as 2000g. Other systems can be use and employ a longer focal length lens with zoom facility, gyro-stabilized and with pan and tilt function with a mass of 3 to 4 kg to a high-power radar

More sophisticate UAV can carry a combination of different sensors within a payload module or series of modules. The data from the sensors can be processed to provide enhanced information or information, which needs data from different sensors to be obtained. For example, images from thermal, color camera and radar scanner can be merged together and may add hidden information of the color image. In addition, other sensors can complement the loss of performance on sensor due to weather conditions, pollution or others. All data is processed in such a way so it can be transmitted via downlink to the control station or other destination.

2.2 UAS applications

There is an extensive array of UAS applications, and it can be divided in two major groups: civilian and military. The most common being the following:

Civilian purposes:

- Aerial photography: Film, video, etc.
- Agriculture: Crop monitoring etc.
- Border Surveillance;
- Coastguard: Surveillance of the coastline;
- Crowd control;
- Exploration: Access to remote areas;
- Fire prevention;
- Meteorological services.

Military purposes:

- Security and Control;
- Aerial Reconnaissance;
- Terrain Search and Rescue;

Figure 2-7 *- UAS Applications*

Chapter 3
ELECTRONIC PROCESSING SYSTEM

The drone brain is composed by two components Pixhawk and the Raspberry PI, and for the system is known as Electronic Processing System (EPS).

3.1 Overview

In order to control the UAV, it was necessary to build an electronic processing system, with a combination of a Pixhawk (flight controller), responsible for controlling the UAV and the Raspberry PI which is in turn responsible communicating with the Remote GCS. Both the Raspberry PI and the Pixhawk are together on the top of the UAV, and they have an important role on the overall system.

3.2 Pixhawk

It is the flight controller used on this project. It can control any UV, depending ~~of~~ on the firmware installed. Pixhawk is a board built on base of the Ardupilot Mega (APM), so some features of the APM are present on the Pixhawk as well, but the Pixhawk power and processing is much more efficient that the APM board.

3.2.1 Pixhawk vs Ardupilot

When choosing between the Pixhawk and the APM, the Pixhawk is the main choice, built on Advanced Rich Machine (ARM) Central Processing Unit (CPU), with a 32 Bits architecture; it is more advanced, with the fastest CPU, most Random Access Memory (RAM), backup accelerometers and gyros.

The last APM, the 2.6, it is built on AVR CPU, with a 8 Bits architecture, with the actual firmware of the Arducopter, consumes all the 8 Bits making this technology obsolete.

	Pixhawk	APM 2.6
CPU	ARM (32 Bits)	AVR (8 Bits)
RAM	256 Kbits	8 Kbits
Flash Size	2 Mbits	256 Kbits
Backup Sensors	Yes	No
Input Ports	14	6

Table 3-1 *- Pixhawk vs APM 2.6* ***Error! Reference source not found.Error! Reference source not found.***

As we can conclude based on ***Table 3-1***, Pixhawk is way better than the APM 2.6; the only advantage of the APM 2.6 is the price, which is around 60$, comparing with the Pixhawk, which is 200$.

3.2.2 Sensors

Pixhawk has embedded backup sensors, which means it has not only one set of sensors but also four different sensors. This makes the system more reliable because several values can be fetched from several sensors; it makes the system safer because if one sensors fails, there should be another to replace it. Those sensors are: MPU6000 as main sensors for accelerometer and magnetometer, ST Micro 16-bit gyroscope, ST Micro 14-bit accelerometer and magnetometer and the MEAS barometer [18].

3.2.3 Composition

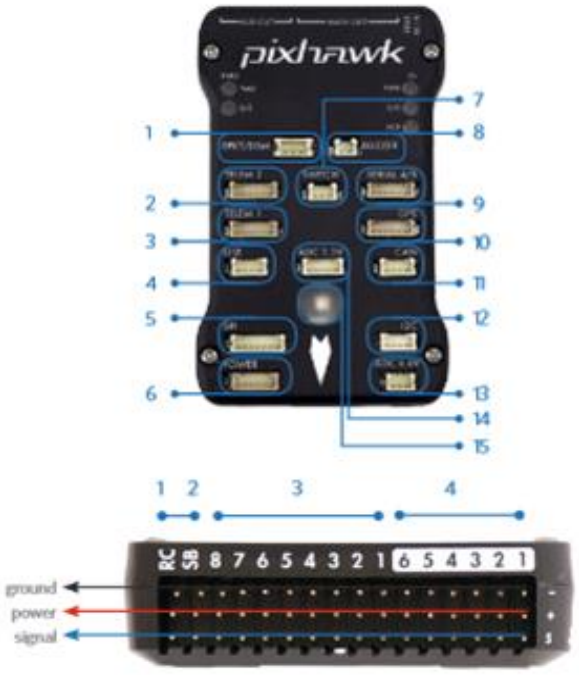

Figure 3-1 *- Pixhawk inputs and outputs*

Figure 3-1 illustrates the inputs and outputs of the Pixhawk. At the top there are the inputs ports. Port 1 is the Spektrum DSM receiver (***Figure 3-2***), which is the radio input signal.

Figure 3-2 *- Spektrum DSM X Module*

Port 2 and 3 are the telemetry inputs and they are the connection with the Raspberry PI. Port 4 is the external Universal Serial Bus (USB) port. Port 5 is the Serial Peripheral Interface (SPI) typically used to connect other sensors. Port six it is the power module; the port can only accept voltage of 5.7 volts and it will be destroyed if the voltage gets higher than 20 volts. In order to control the voltage, a power module is needed, as we can see in ***Figure 3-3.***

Figure 3-3 *- Pixhawk power module*

On port 7, there is the safety button, a really important feature of the Pixhawk, ~~it~~ which is responsible to allow the UAV to fly, otherwise the UAV wouldn't work. It is a button with a Light-Emitting Diode (LED) and there are four different LED combinations for different messages, as we can see in Figure ***3-4***

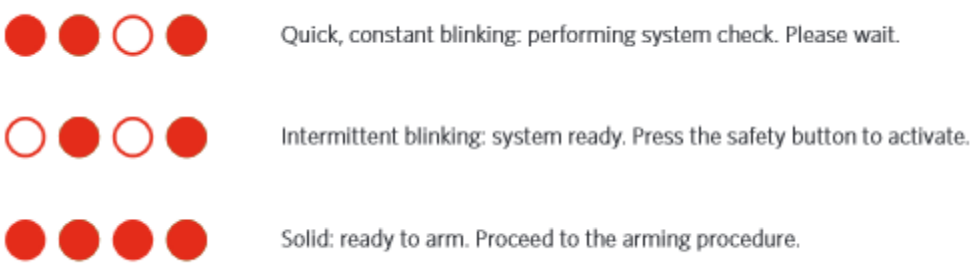

Figure 3-4 *- Safety button different messages Source:* ***Error! Reference source not found.***

Port 8 is the buzzer port. It emits sound, allowing the user to know the different errors or if everything is right. There are four different sounds, one for errors, for calibration, for arm/disarm and finally for GPS lock. Port 9 is a serial port, the same as port 5. It is used for other sensors that are not present on the Pixhawk. Port 10 is designed for the GPS module.

Port 11 is the Controller Area Network (CAN) bus. Port 12 is for compass, port 13 and 14 are analog digital convertors and finally port number 15, the LED indicator, together with the buzzer are the Pixhawk information displayer, with different colors combinations for different messages types as we can see in ***Figure 3-5.***

Figure 3-5 *- LED messages combination* ***Error! Reference source not found.***

On the bottom of the Pixhawk, as we can see on ***Figure 3-1***, there are located the output ports. In first place the RC port is for output telemetry to the radio control, the Serial Bus (SB) is for sensors output. The following 8 ports are for the ESC that give speed to the motors, which mean that the Pixhawk can control 8 motors, the final 6 ports are for extra servos, to control for example a 3 axis gimbal is required 3 ports.

And finally there is the auxiliary output, which is an extra 6 port for control servos, for example to control the 3 axis gimbal, there is a need for 3 auxiliary outputs.

For this thesis it is not required to use all the input ports on the Pixhawk, so as it is shown in ***Figure 3-6,*** there is only one need: Raspberry PI, Spektrum DSM X, Safety Button, Buzzer, GPS Module and the Power Module.

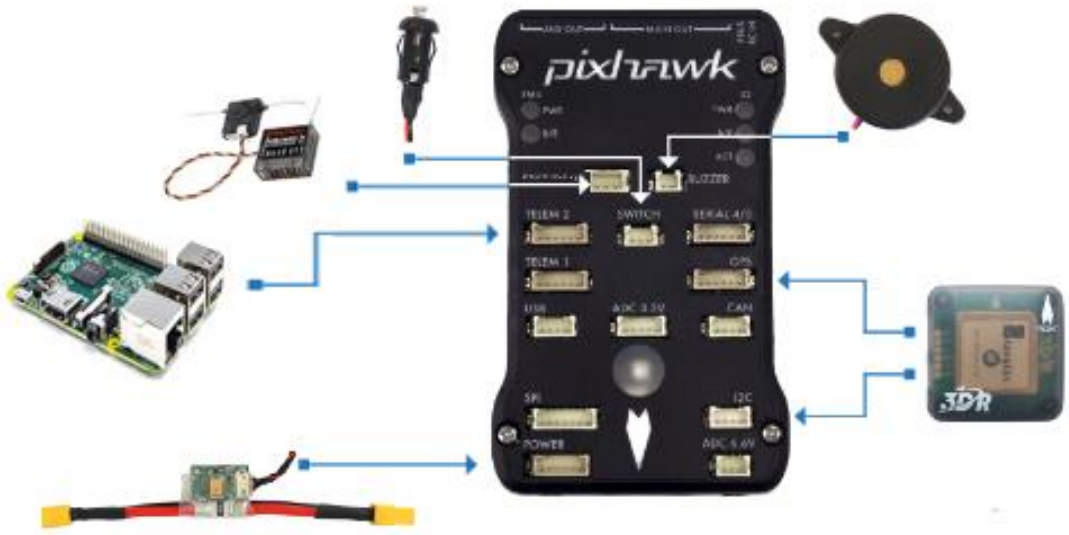

Figure 3-6 *- Required components to control the UAV*

3.2.4 Arducopter

The firmware that the Pixhawk runs is the Arducopter, in this thesis the latest version 3.2.1 was used, it is an open source firmware that supports the fully autonomous waypoint based flight and the real time control, using MAVLink protocol to communicate with the Remote GCS.

Arducopter is a really strong firmware to control UAV, no programming is required and it is easy to install. The main technical features are:

The flight autonomous stabilization and the altitude control, allows the user to have a simple control of the UAV, making the UAV to self-control, making the balance using the Pixhawk sensors data, giving the user ~~a~~ the simple task of only controlling the direction of the UAV.

It allows the Waypoints navigation, it only needs the Geo coordinates for the waypoints and the UAV flies autonomous.

It provides fail safes to the UAV, which means if everything goes wrong for example low battery, the UAV automatically lands and turns off.

It provides the user an array of different flight modes, allowing a diversity to control the UAV, not only the standard control. The possible flight modes to select are:

Flight Modes	Description
Stabilize	Provides a self-stabilization to the UAV, allowing the simple flight to the user.
Altitude Hold	Makes the UAV still in the current position, with the pre-defined Altitude, this altitude is sent by the user using the application
RTL (Return-to-Lunch)	When the user starts the fly it is saved the Lunch coordinates, so anytime of the fly the user can performs this mode and the UAV returns to the lunch side.
Auto	Auto mode it is the Waypoint navigation mode.
Circle	Makes the UAV to perform circles on the current position.
Position Hold	Makes the UAV still in the current position.
Land	Set the UAV to land on the current position.

***Table 3-2** - Flight Modes available to the user.*

3.3 Raspberry PI

In order to have the Pixhawk connecting with the Remote GCS, there was a need to have a bridge connecting both systems, and the solution found was to use a Raspberry PI.

Raspberry PI is a small packed computer with a Linux based operating system, and it is relatively cheap. The model that is used on this thesis is the Raspberry PI 2, which is an upgrade of the previous version, with 1 GB RAM and a quad core ARM processor, making even possible to run Windows 10. For our purpose, it makes the MAVLink processing really fast and efficient.

3.3.1 Connect to Pixhawk

Using the Raspberry Pi as the bridge between the Pixhawk and the Remote GCS was possible because both boards have compatible ports. As we can see in ***Figure 3-7*** the connection is

made by using Universal Asynchronous Receiver/Transmitter (UART) cables, which connects the Pixhawk to the Raspberry I/O pins.

With this solution, the delay in communication between the Pixhawk and the Raspberry was eliminated. Previous works were based on Bluetooth communication, which compared to UART communication has a certain delay.

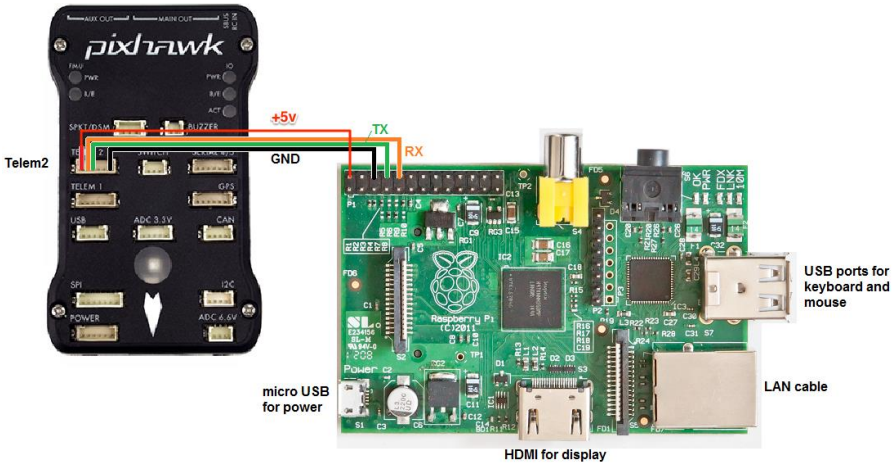

Figure 3-7 *- Connection between RPI and Pixhawk. Source:* ***Error! Reference source not found.***

To make the Raspberry compatible to receive MAVLink packets, several Python liberties were installed. Raspbien (Raspberry OS) was built with Python, and there was also a need to have a Python scrip to read and write to the Pixhawk that only communicates via MAVLink. The solution found was to install the MAVProxy project built in Python, which was the perfect solution for the problem.

3.3.2 Composition

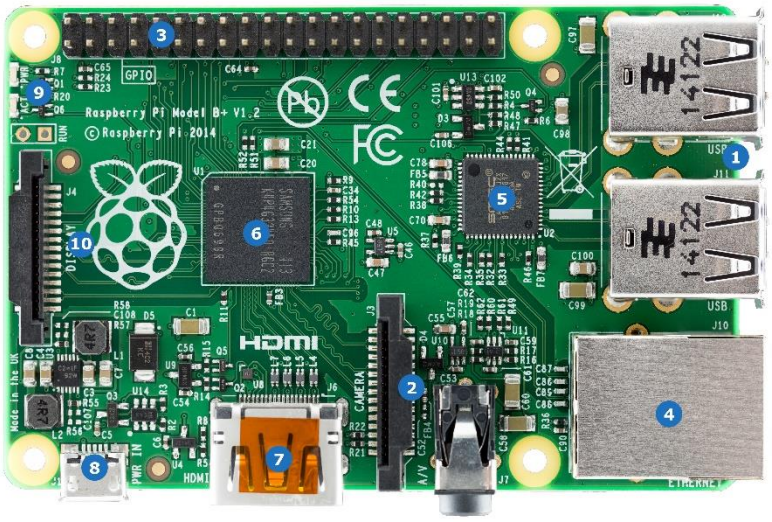

Figure 3-8 *- Raspberry PI composition*

Figure 3-8 illustrates the Raspberry Pi 2 Model B, this model was used on this project and it is the most updated model on the market.

For a better comprehension of the Raspberry composition it was divided in 10 points. The number 1 is the 4 USB ports, number 2 is the Camera Interface using the Camera Specifications Interface (CSI). Number 3 is the output pins, it is used for connecting the raspberry pi with the Pixhawk. Number 4 is the Ethernet port, useful to connect the raspberry pi to the internet via Ethernet cable. Number 5 and 6, it is the RAM module and the CPU module respectively, with 1 GB of RAM available it makes a really powerful tool on the system. The CPU module is the Broadcom BCM2836, which includes a CPU, Graphical Process Unit (GPU), Digital Signal Processing (DSP) and Synchronous Dynamic Random Access Memory (SDRAM). With the 900 MHz quad-core ARM Cortex A7 CPU the Raspberry PI can perform six times faster the same task than the old Raspberry. Important as well is the GPU module the GPU Broadcom VideoCore IV @ 250 MHz that can encode and decode h.264/MPEG-4 AVC really fast, essential to record and stream the video. Number 7 is the High-Definition Multimedia Interface (HDMI) input, allowing the Raspberry Pi to be displayed on a screen. Number 8 is a micro USB interface, which is used for the power supply to the Raspberry PI, it is need a minimum of 3.3V and maximum of 5V for the power supply. Number 9 is the LED indicators, there are 2 LED, a green and a red, and they are for the Raspberry status. Blinking green everything is okay, blinking orange something is wrong but it is not critical, blinking red critical error, the Raspberry will not work. Finally number 10 is the Display Serial Interface (DSI), and it is for have a small display connected to the RPI.

3.3.3 Camera

The Raspberry Pi camera module (***Figure 3-9***) can be used to record high-definition video as well as taking pictures. It is very beginner friendly but also providing advanced features for experienced users. Visual effects can be created if you use the available libraries bundle with the camera. For High Definition (HD) video, it is mandatory to know that the camera features a five-megapixel fixed-focus camera that supports 1080p30, 720p60 and VGA90 video modes. It attaches via a 15cm ribbon cable to the CSI port on the Raspberry Pi. All models of Raspberry Pi 1 and 2 support the camera. It can be accessed through the MMAL and V4L APIs and there are numerous third party libraries built for it, including the Picamera Python library.

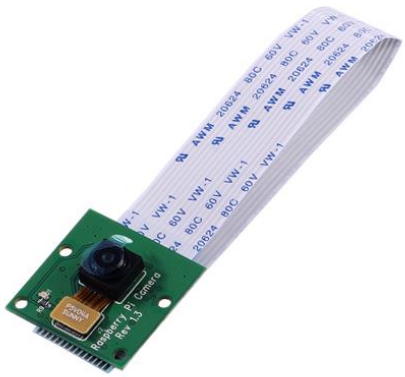

Figure 3-9 *- Raspberry Camera Source:* ***Error! Reference source not found.***

3.3.4 Network Communication

In order to connect the Raspberry PI to Server, it has to have the ability to connect to the internet and for that there are two different approaches, wired or wireless. Because the Raspberry PI is located on the UV, there is not the possibility to have a wired internet connection, so the use of the Ethernet port for internet access is impossible, so the only solution is the use of wireless networks.

With the USB ports available is it possible to attach to the Raspberry PI wireless dongles, which can connect to the internet. There is two different type of dongles 3G/4G dongle and Wi-Fi dongle.

Using a Wi-Fi dongle the Raspberry PI is ready to use it without using any driver, but for the 3G/4G dongle there is the need to install the Sakis 3G software as we can see on ***Figure 3-10***, because there is no direct way to connect a 3G/4G dongle to Raspberry PI. The use of Sakis 3G does not bring any advantages, because only some dongles work with this program, and none of the new 4G dongles work, which limits the system to use 3G. Not only that but when using Sakis 3G the Raspberry PI doesn't have the ability to get the signal strength which is an important feature to collect data. There is no other programs that allows this type of connection, so the use of direct 3G/4G dongle is doable, and makes the system usable but compared with the Wi-Fi, this solution is worse as it can be proven on ***Chapter 6.***

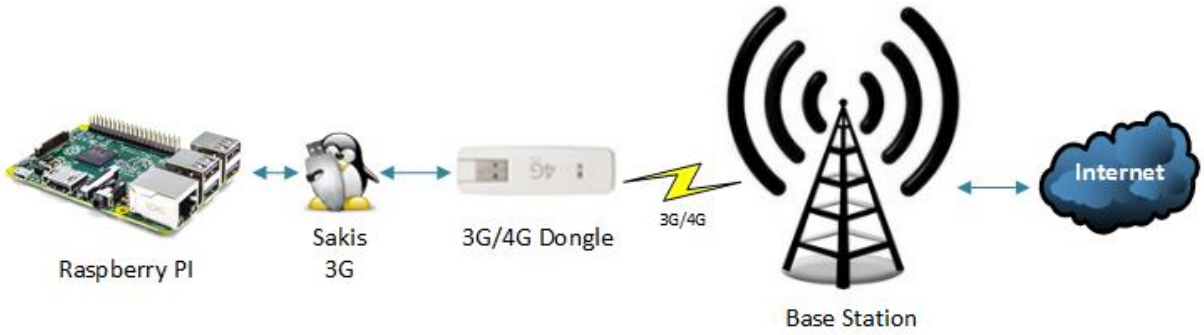

Figure 3-10 *- 3G/4G Raspberry Connection with Sakis 3G*

For the Wi-Fi dongle there is no need to do anything the Raspberry PI is already able to connect to the internet, it is only need the internet credentials such as the SSID and WPA2 password, and it connects automatically. However, while on air Wi-Fi cannot cover some complications. There is no Wi-Fi hotspots to cover a big area, so the perfect solution is to build a cellular system. Nevertheless, that connects to the Raspberry PI via Wi-Fi, and the solution is possible because of the use of a portable hotspot located on the UV, that connects to the 3G/4G cellular network and then creates a Wi-Fi hotspot, that the Raspberry PI can connect as we can see on ***Figure 3-11***. This solution joins the two approaches the 3G/4G and the Wi-Fi, and makes a reliable system.

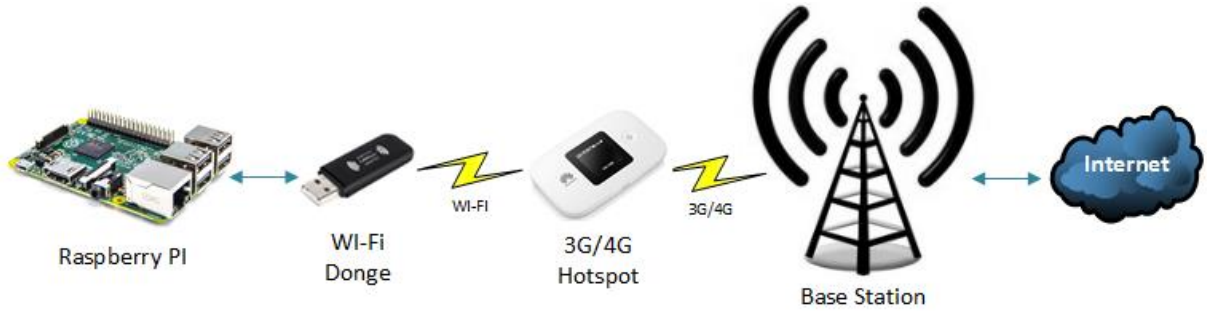

Figure 3-11*- Wi-Fi Raspberry PI Connection*

This page was intentionally left in blank

Chapter 4
APPLICATIONS

To have a robust system, three different applications were developed:

Remote GCS, Server and Raspberry Pi.

4.1 Overview

The Remote GCS application was built using an Android application, and it is the user interface to control the UV. Server application, which is invisible to the user, are critical to the control of the system. Finally, there are the Raspberry applications that are responsible to control the UV. All the three applications are connected together.

4.2 Remote GCS

The Remote GCS application is made only for android platforms. Two different approaches were developed, for tablet and phone devices. It is highly recommended the use of the tablet, because it has a higher area, which allows the user to have a better view of the UV.

When the user opens the application, the login screen becomes visible, as it is shown in ***Figure 4-1***. It is important to have a login system to centralize all the data and sort by user, so after a connection the UV is connected to user, where all the data is saved on a database.

Figure 4-1 *- Remote GCS Application Main Activity*

If it is the first time that the user opens the application, a registration must be done. This will add a new user to the database where all the data from the flights will be stored. After this first registration, the user can do the login with the registered credentials.

Figure 4-2 *- Remote GCS Application User Registration*

After the login, the user enters on the Flight Control tab, which allows the full control of the UV. The application was built to have a simple look. In this point, the user has only two options: connect to Server; to search for online UVs; or change some settings, both of the buttons are located on the top right corner.

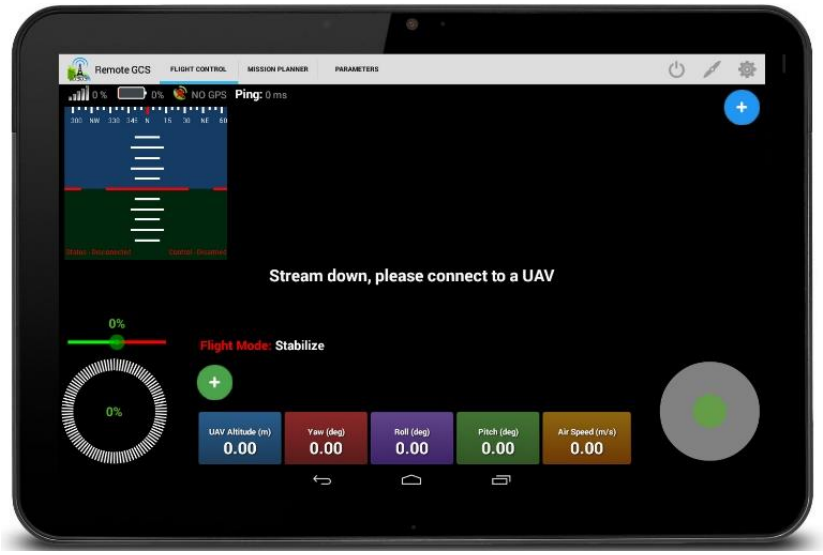

Figure 4-3 *- Flight Control View*

When the user connects to the Server a list of the available UVs to control will appear on the left side of the screen as can be seen in ***Figure 4-4***. After choosing the UV, the connection is completed and the user has all the commands to control the UV.

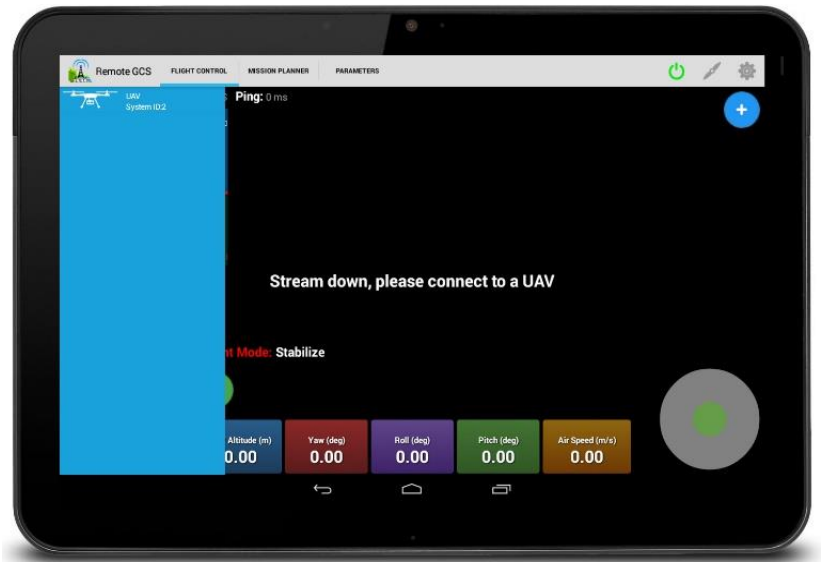

Figure 4-4 *- List of UVs for control*

After selecting the pretended UV, it starts to receive the video stream from the UV, it is real time video stream, which allows the behind line-of-sight (BLOS) control. The Flight Control Tab (***Figure 4-3***) can be divided in 4 Areas: The top left corner is where the connection status is located, such as ping which is the time that takes a message to do the travel between the Remote GCS and the UV. UV battery, GPS signal and the signal quality, which is the rate of successful packets received. Bellow that status is the Head up Display (HUB) and it is where the UV is displayed, Yaw, Roll and Pitch position, on top of the HUB is the Heading the orientation of the UV.

The system commands, connect, arm and settings buttons are located at the top right corner. The possible settings that the user can change are shown in ***Figure 4-5***. The network settings allow the changing of the server IP and Server and UV Port; those settings are for more experienced users though. The Gimbal Settings allow the user to set speed of the Gimbal motion, selecting between Slow, Medium, and Fast. Statistics Settings and General Settings are only check boxes allowing the user to display the flight statistics, enable the ping, and signal measures.

Figure 4-5 *- Application Settings*

The UV menu is located below, with more options to control the UV such as, set control to RC, change the gimbal position, voice commands, etc.

The bottom part of the view is divided into 3 parts: on the left the throttle, level is located, which is a wheel that can set the UV throttle level by dragging. Above there is the yaw bar, that allows the UV to spin through the Z-axis. In the middle, you find the UV information, where it has displayed the UV current information, such as the altitude, yaw, roll, pitch and speed. Above there is the flight mode changer, which allows the user to select the flight mode to apply to the UV. Finally, on the right, you have the direction joystick; together with the throttle is the responsible to control the UV.

Figure 4-6 *- Flight Control Tab View*

The second tab is the Mission Planner that provides the user the option to create a mission for the UV to perform. The way that he can accomplish that is by doing long clicks on the map, it creates a waypoint, the default waypoint color is blue, and after the pretended route is created, the user can write it to the UV. On this point after the UV, loading the route the UV is available to follow the route. The user has the option to read a mission from the UV, and it will be displayed on the map the route that the UV has. An example of a route course is shown in ***Figure 4-7***

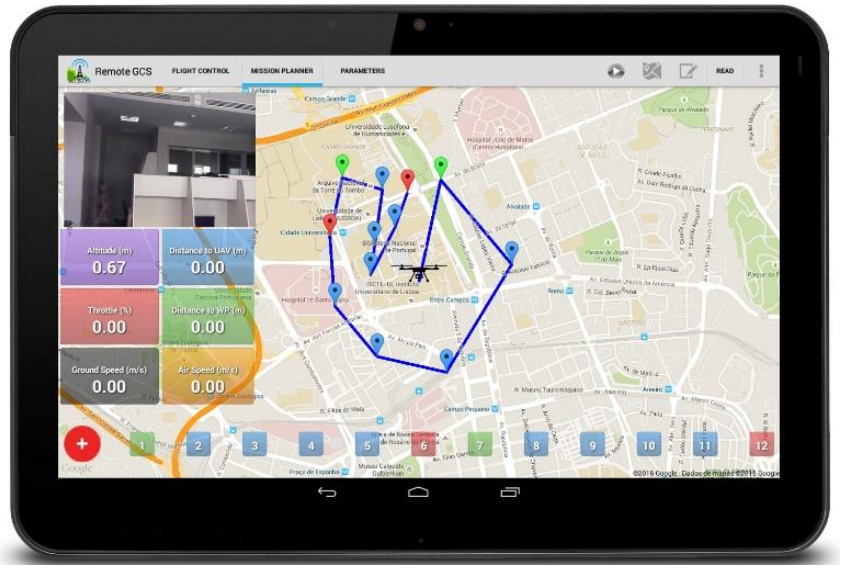

Figure 4-7 *- Application Mission Planner Preview*

There are three different colors: red, green and blue. The red waypoints are for take-off, which mean that the UV is going to take-off on that location; the green waypoints are the land waypoints, indicating where the UV will land, and finally the blue are the normal waypoints where the UV will pass through while performing his route. If the user wants to switch the settings of the waypoints, it is possible to do it by clicking on the bottom menu where all the waypoints are displayed on the map. By clicking on the waypoint, it displays a menu according to the type. The first field is the type, where we can select between the three types, and then change the settings according to the type. The ***Figure 4-8*** represents the settings of a normal waypoint where the user can change the radius that is an imaginary circle around the waypoint when the UV reaches the center it is considered to have reached the waypoint. Then, there is the delay, which means the time that the UV stays on the waypoint until passing to the next one, the orbit is for the UV performs an orbit on the waypoint, and finally the altitude is the altitude waypoint. The geographic coordinates cannot be changed in order to prevent some position errors.

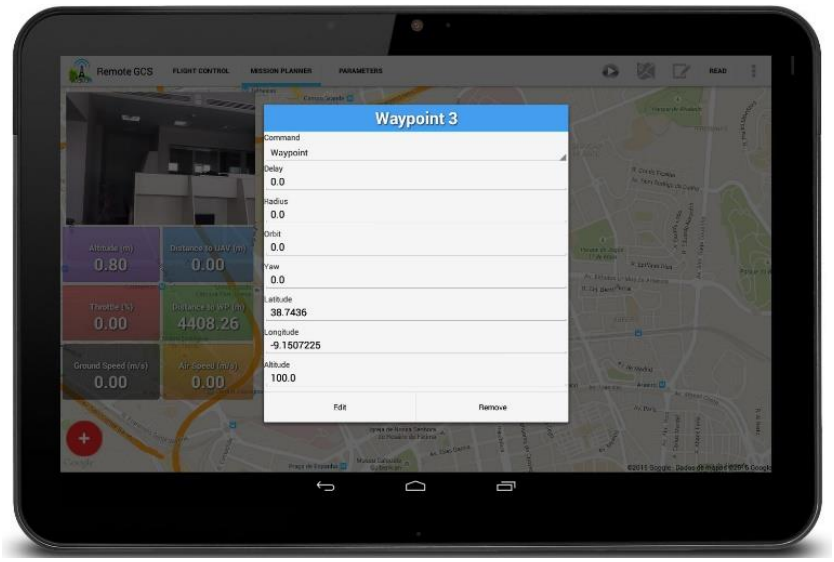

Figure 4-8 *- Mission Planner Waypoint Editing*

The third and last tab is the Parameters Tab, where the user can change the settings related to the UV. There are several parameters, and not all of these parameters can be changed while the UV is performing a mission or flying.

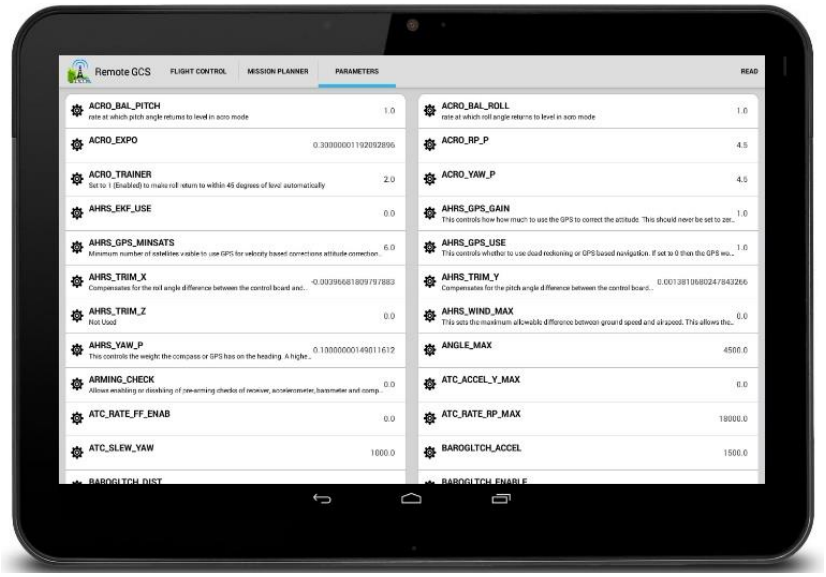

Figure 4-9 *- Application Parameters*

When the user wants to change a parameter, a brief information related to that specific parameter appears as we can see in ***Figure 4-10***, and some extra information. When changing a parameter, the user must have to be careful, because these parameters are not confirmed, which means that a bad parameter will be read by the UV as it is, that will provoke bad reading.

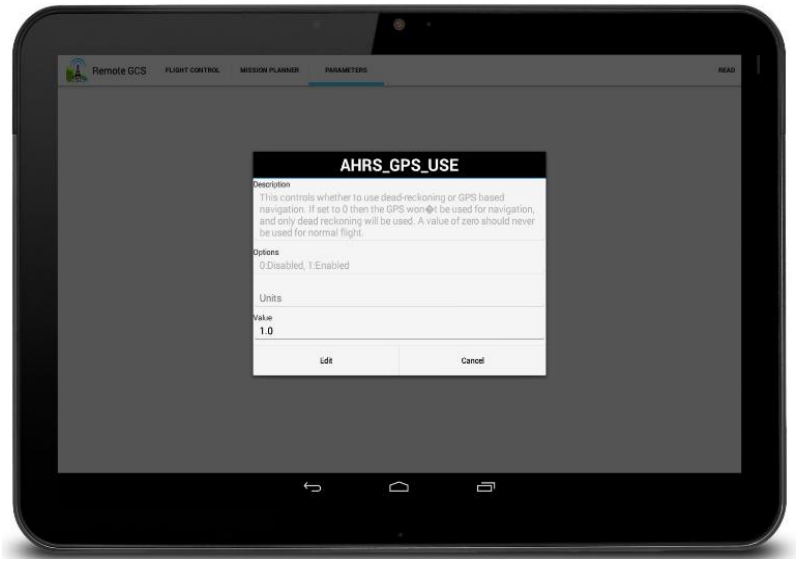

Figure 4-10 *- Parameters Editing*

4.3 Server

The server was created to overtake the NAT problem, describe on ***Chapter 5.*** To make the bridge between the UV and the Remote GCS there was the need to have a server to control. The objective of the server is to manage the messages between the UV and GCS. The server was built to be able to handle multiple clients, which means it can manage various flights at same time. One GCS can only control one UV, and only one UV can be controlled by one GCS. The server was built in Java as we can see the ***Figure 4-11,*** the server classes Diagram.

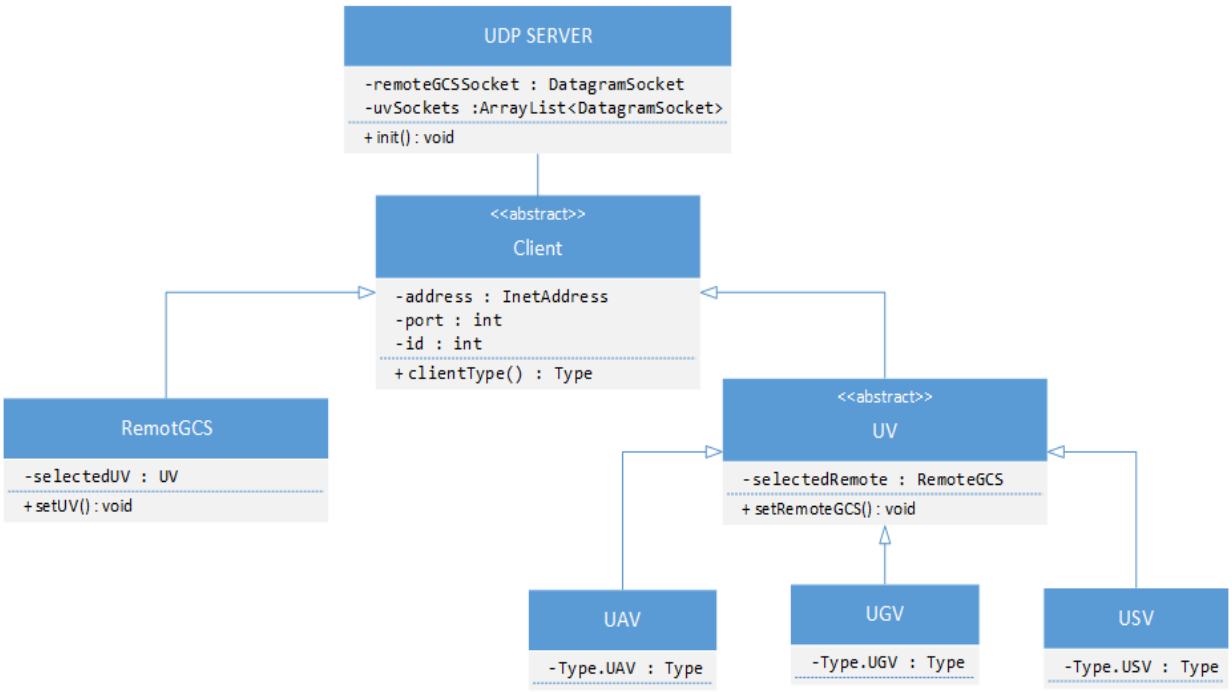

Figure 4-11 *- Server Classes Diagram*

To communicate via the internet, there were two viable options, the UDP or the Transport Control Protocol (TCP). The option chosen was the UDP, because we are dealing with ~~a~~ real time communications and UDP is faster than the TCP **Error! Reference source not found.**. However, the UDP is not confirmed, which means that the messages could not reach the destination. Therefore, to overtake that situation, the server must have a confirmation system.

When a UV turns on, it try to connect automatically to the server, it does not get the response that it is connected to the server, it does not need to know that. When the server gets the UV messages, it creates thread to handle that UV, and identifies the UV type. At this point, the UV is connected to the Server. The UV never stops setting messages to the server, unless the server is offline.

When one Remote GCS connects to the server ***(1)*** the server replies with and Acknowledgment (ACK) ***(2)***, but this ACK is not a simple ACK, it carries the UVs available to select. Then the user selects ***(3)*** one of the UVs and the server replies with an ACK.

After this initial protocol, the Server, the UV and the Remote GCS are all connected, now the thread that deals with the UV is now dealing with the connection between the UV and the Remote GCS.

When the user disconnects, the UV is free and another Remote GCS can select it again.

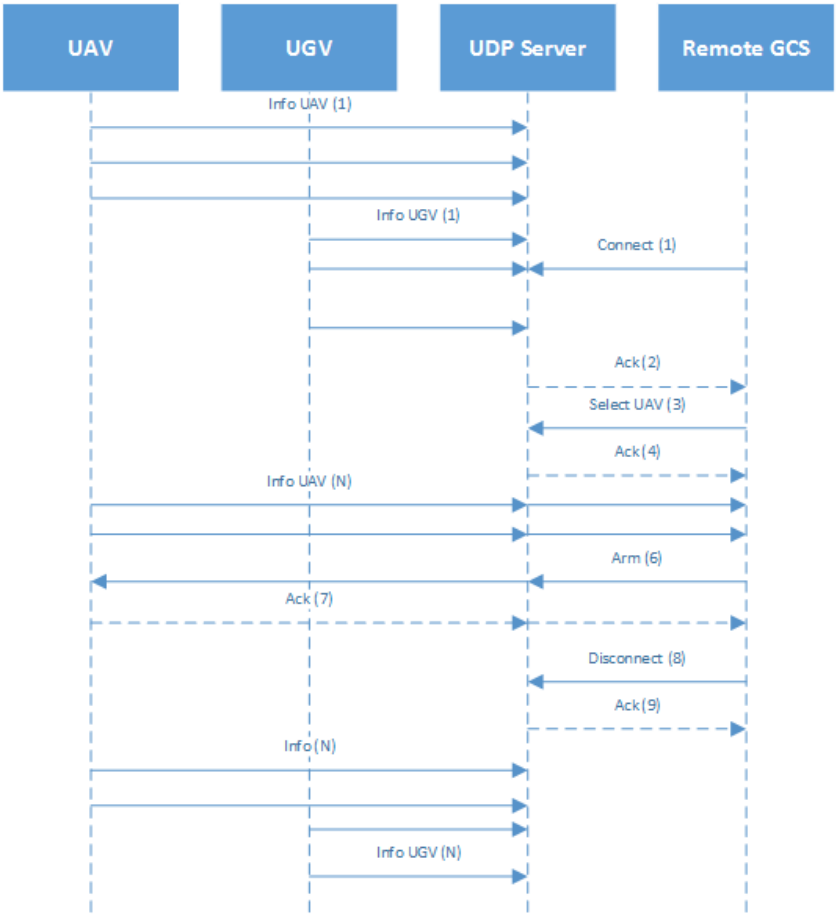

Figure 4-12 *- Server Messages routing Protocol*

To overtake the non-message confirmation from the UDP, the Server includes a confirmation system, at it is shown in ***Figure 4-13***.

When the GCS connects, it sends a CONNECT message to the server, and if the server does not send an ACK confirming the receiving, the CONNECT message, the GCS sends the same message all over again. If GCS waiting time for the ACK answer is equals or higher than the timeout time, the GCS displays a message saying that the server is offline. When the server replies, the CONNECT ACK carries a message saying there is a UVs on the system, if not the GCS cannot connect to the server. In the case of UVs on the system, the CONNECT ACK have the list of UVs present. It appears in a list to the user select. When the user selects a UV it sends a SELECT UV message with the id of the pretended UV, and the server replies with a simple ACK. If not by some reason, the user is disconnected. After the UV selection and the Server confirmation, the user is online, and at this moment the communication is stablished.

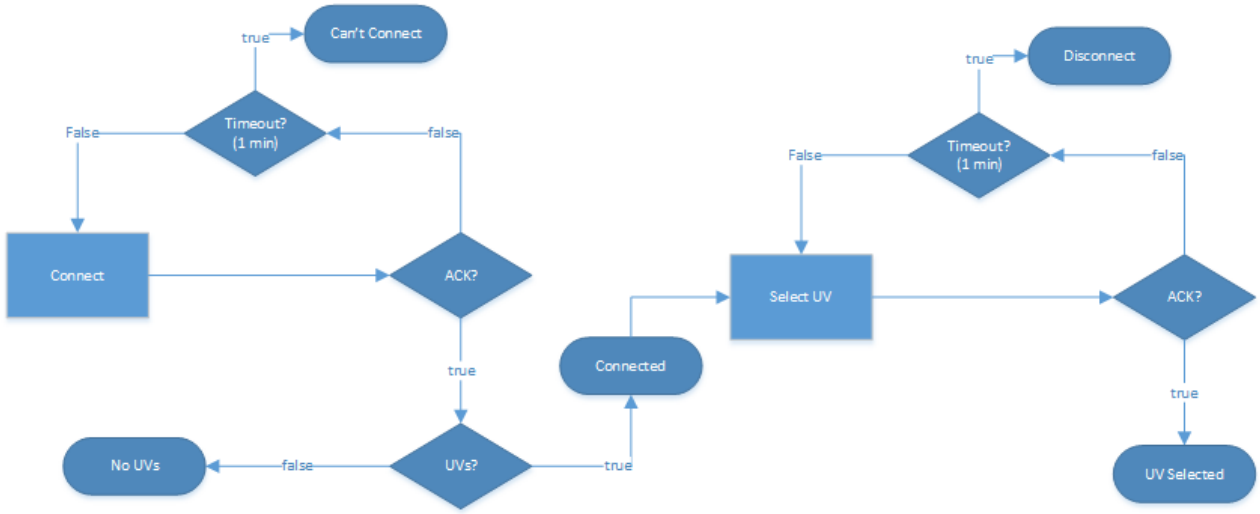

Figure 4-13 - *Server Connecting Fluxogram*

4.4 Raspberry PI Scripts

The Raspberry PI is located on the UV, and it is the gate between the GCS and the UV control. It connects to the flight controller via UART cable **Error! Reference source not found.**, so there is no delay in the communication. The flight controller uses MAVLink protocol to communicate so the Raspberry Pi has to communicate via MAVLink, and that is where the MAVProxy enters the picture; MAVProxy is an array of MAVLink classes. A Raspberry Cam is used to capture the video, and then is streamed to GCS.

4.3.1 MAVProxy

To build the communication between the Flight Controller, in our case the Pixhawk, there was a need to build an interface that allowed the connection to the internet in order to communicate with the Remote GCS; the solution was using a raspberry as a gate. Raspberry PI allows the use of WIFI or 3G/4G dongles to have internet access so it can connects to the server and send data to the Remote GCS.

To communicate with the Pixhawk (that uses MAVLink to send commands to the UV), the Raspberry had to use the same protocol. In order to accomplish that MAVProxy suited perfectly. Its project developed by Canberra UAV OBC team **Error! Reference source not found.**, built in python that can not only run on Raspberry PI and communicate with the Pixhawk, but also allows the out gateway. So the MAVProxy communicates with the server and the communication between the UAV and the GCS is now completed.

4.3.2 Stream

In order to have a live video stream on the Remote GCS different approaches were tested. One was to use a GoPro **Error! Reference source not found.**, but the live stream was only provided via WIFI, and if the UV was behind the line of sight, it would be impossible to have that kind of stream.

The next solution was to use a Raspberry Camera, which is a small component relatively cheap that can be attached to the Raspberry easily. However, this option presented a problem: the camera not only encodes the video in H264, a format that is not playable on Android devices, but there was also a delay of 45 seconds, which wasn't viable for live control. So the solution found was to encapsulate the H264 stream into a Fast Forward Moving Picture Experts Group (FFMPEG) stream, which is compatible with Android devices.

The Real Time Messaging Protocol (RTMP), that sends the stream to the Server, was used in order to transport the stream. The Wowza Media Systems **Error! Reference source not found.**, was installed on the server. It manages all the incoming streams and provides Real Time Streaming Protocol (RTSP) transportation to each incoming stream, as we can see of ***Figure 4-14***. Finally, the stream reaches the Remote GCS with less than 1 second of delay, which allows a live control.

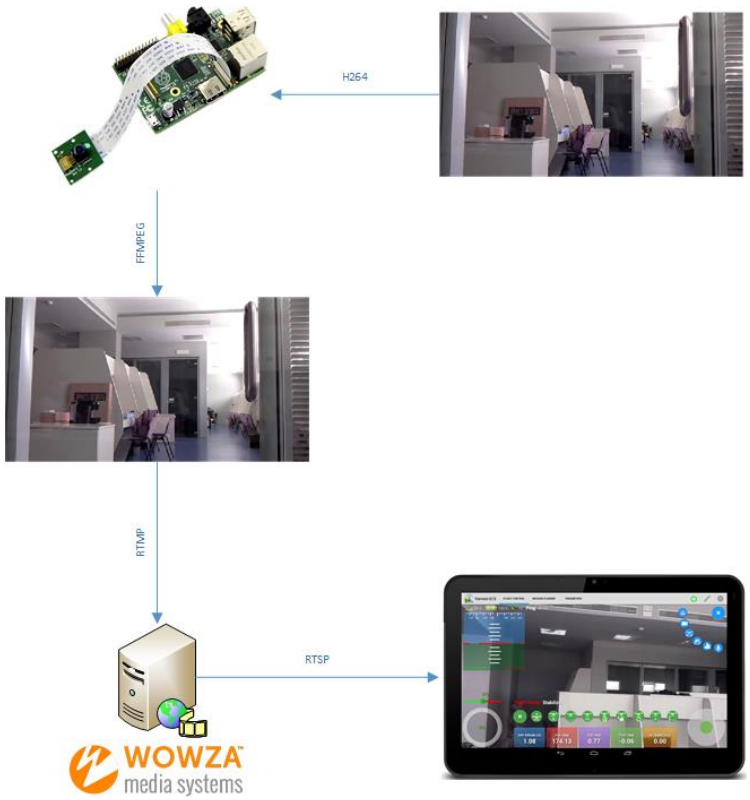

Figure 4-14 *- Video Stream Architecture*

To get the least delay possible and get the lowest bitrate and in order to get more processor the telemetry possible. Several video options were tested, and four are available to user selection, depending on the internet environment - if there is a good coverage making possible to reach good bit rates, it can use the normal stream as we can see in ***Figure 4-15***

Figure 4-15 *- Normal video stream 20 FPS*

Making the video black and white as we can see in **Figure 4-16** decreases the delay and the bit rate, which is still a viable option to control the UV.

Figure 4-16 *- Normal video with black and white saturation*

To get even better bit rates, we can low grade the video quality as we can see in ***Figure 4-17,*** decreasing the bitrate to half of the normal value, which is a viable option in low coverage environments.

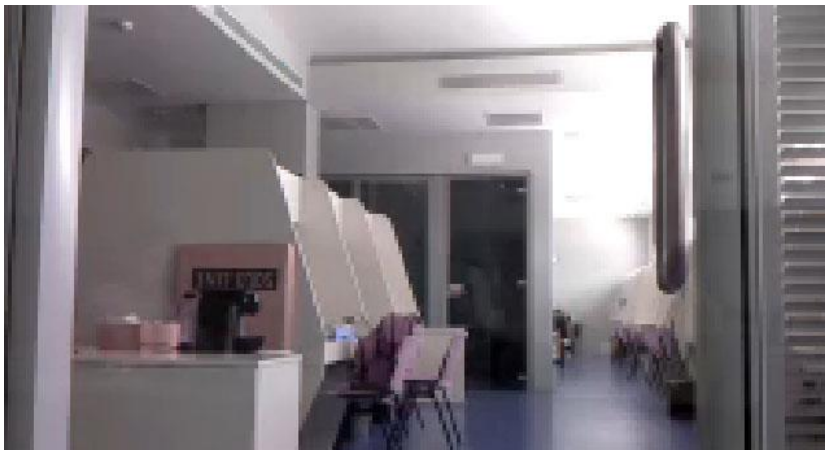

Figure 4-17 *- Low quality video 15 FPS*

Finally, the economic option with the lowest bitrate and lowest delay is the one with the low quality and the black and white saturation as we can see in the ***Figure 4-18***, still a viable option to control the UV.

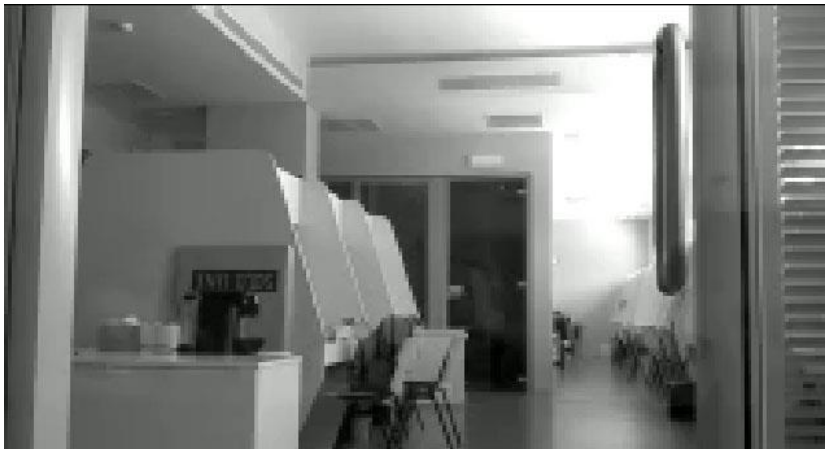

Figure 4-18 *- Low quality video with black and white saturation*

In order to get the values of ***Table 4-1,*** the video stream was tested for 10 minutes. Moreover, it can be concluded that the results from the low quality videos are better than the normal ones, and that the black and white makes a significant difference on the stream, so the video option is given to the user to choose whatever option he wants depending of the case.

Video	FPS	Delay	Bitrate	Buffering*
Normal	20	±0.35 s	±65 kbps	3
Normal Black & White	20	±0.30 s	±62 kbps	2
Low Quality	15	±0.20 s	±11 kbps	1
Low Quality Black & White	15	±0.15 s	±10 kbps	1

***Buffering –** Number of times that the video made buffering.

Table 4-1 *- Different videos performances*

Chapter 5
COMMUNICATIONS

One of the focuses of this project is the communication system: must be improved as much as possible to provide real-time and reliable operation of the UAV.

5.1 Overview

In order to communicate with the UV there are three different approaches via RC Control. Which consists in a RC controller sending telemetry to the UV, but limits the control to the line of sight control, though Wi-Fi communications which allows a behind line of sight control however there is a need to build a Wi Fi Access points in order to the UV to have Wi-Fi signal. Finally, via Cellular Networks, using the 3G/4G network to access the Internet to control the UV, this solution allow a behind line of sight control and there is no need to build any infrastructure because the cellular network is present in some places. In order to communicate from the Electronic processing system and the UV it is used the MAVLink protocol. In ***Figure 5-1*** there is represented the overview of the communication architecture, with the three solutions presented.

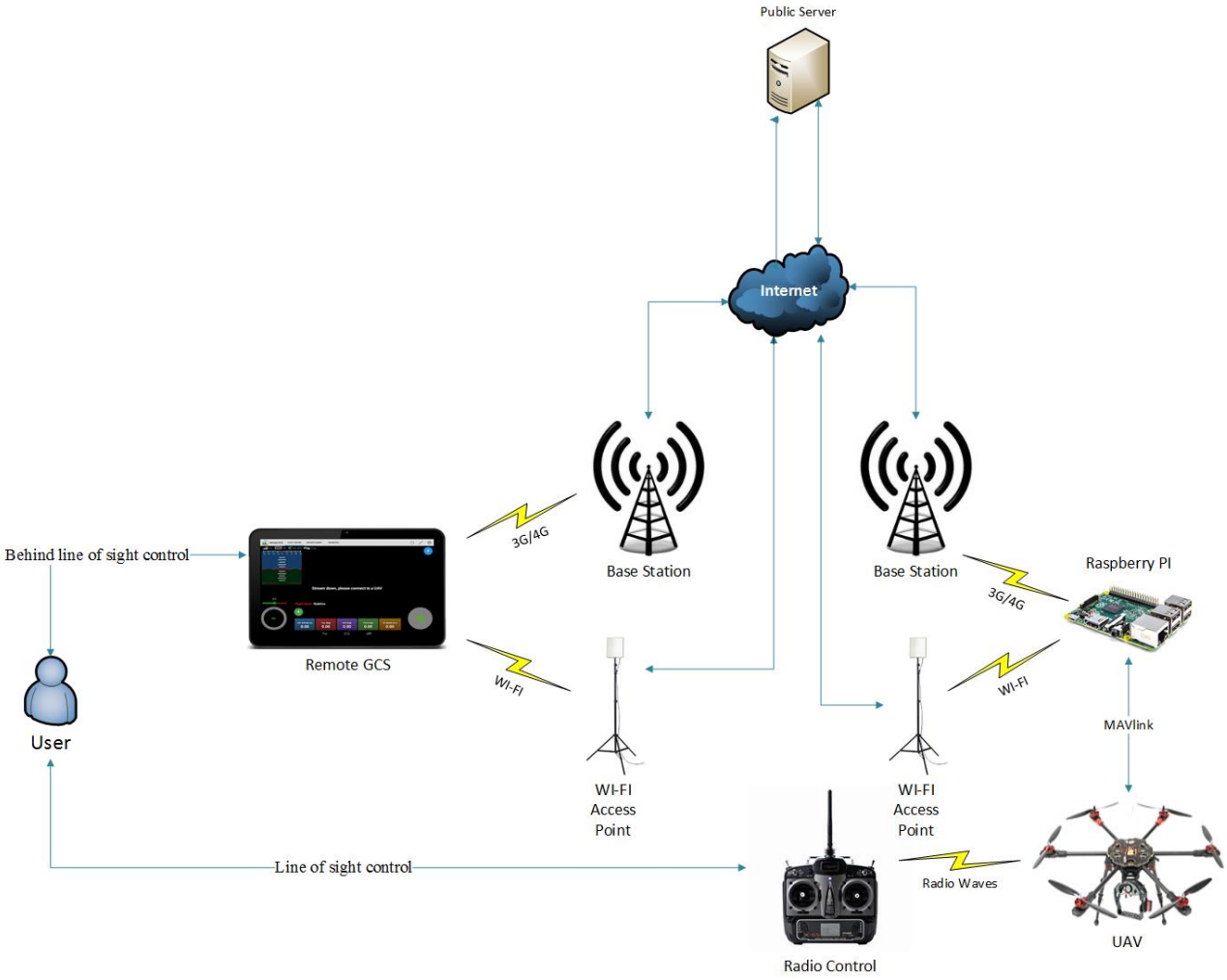

Figure 5-1 *- Communication System Overview*

5.2 Radio Control

The first solution to control the UV is to use a Radio Control (***Figure 5-2***), which allows a full control of the drone.

The RC Control that is used on this project was the Spektrum DX61. Normally the typical RC Control works on High Frequency (HF) and Very High Frequencies (VHF) bands such as 27MHz, 35 MHz, 49 MHz and 72 MHz .Spektrum RC Control works on 2.4 GHz, Industrial Scientific and Medical (ISM) band, because it uses a Direct-Sequence Spread Spectrum (DSSS) technology. Direct-sequence spread spectrum is a spread spectrum modulation technique, which is the information, is splinted into pieces and allocated across the frequency spectrum, the information then is combined with a Chipping Code that divides the information according with a spreading ratio, and this is useful because it helps the signal to be resistant to interferences [26].

The Spektrum DX6i uses 6 different channels, and each channel can be assign to preform whatever task the user wants, however some channels are better to perform a specific tasks, for example the output from the sticks, can only control the Throttle or the Roll/Pitch control, in order to have a safe control.

Figure 5-2 *- Spektrum DX6 RC control, Source:* ***Error! Reference source not found.***

5.3 Wireless networks

In order to solve the Line-of-sight problem there was the need to build an architecture able to overtake this problem, and the solution found was using the already existing cellular networks, in order to access to the internet.

The use of the Cellular Networks, 3G or 4G has the possibility to access the internet, which is critical for the system. The 2.5G appeared in order to upgrade the 2G with the Global System for Mobile (GSM) technology, introducing the General Packet Radio Service (GPRS) and with the 2.75G the Enhanced Data for Global Evolution (EDGE) technologies, which allows the user to access the internet for the first time on mobile devices.

The 3rd generation (3G) comes along to be an upgrade to the 2G, it has a better spectrum efficient. Beside the voice, already present on 2G, the 3G introduce the data packets, thus provides a better internet connection. Universal Mobile Telecommunications System (UMTS) was the first 3G technologies which achieved speeds of bitrate of 340 Kbps and with a latency around 100 MS **Error! Reference source not found.**, with the new upgrades of the UMTS it can achieve bitrates of 2 Mbps **Error! Reference source not found.Error! Reference source not found.**. To accomplish higher speeds and better latencies a new technology was created, the High Speed Packet Access (HSPA), which is divided in two, the High Speed Uplink Packet Access (HSUPA) for uplink communications and High Speed Downlink Packet Access (HSDPA) for downlink communications, which can get bitrates speed of 14.4 Mbps for uplink and 5.7 Mbps of downlink. HSPA+ was an upgrade to the HSPA and it can achieve bitrate speeds of 28 Mbps and 42 Mbps, and can get latencies of 50 MS, which is half of what could be get using the UMTS **Error! Reference source not found.Error! Reference source not found.**.

In order to get better bitrates the Long Term Evolution (LTE) was created to be the next 4G, however it didn't fulfill the requirements to be the next 4G technology intend is more known as the 3.99G, it has bitrates of downlink 100 Mbps and uplink 50 Mbps. The 4G was then created with the use of the LTE Advanced, which is an upgrade of the LTE technology, and have bitrates of 3 Gbps of downlink and 1.5 Gbps of uplink.**Error! Reference source not found.Error! Reference source not found.**

	UMTS/HSPA/HSPA+			LTE	
Downlink Transmission	384 Kbps	14 Mbps	28 Mbps	150 Mbps	1 Gbps
Uplink Transmission	128 Kbps	5.7 Mbps	11 Mbps	76 Mbps	500 Mbps

Round Trip time	150 ms	100 ms	50 ms	10 ms	10 ms

Table 5-1 *– Different technologies Bitrates and Round Trip Times* ***Error! Reference source not found.Error! Reference source not found.***

5.4 Network Address Translation

A computers or a mobile device that is connected to the internet has an IP associated, and each device has to have a single IP address, thus an IPV4 that is the protocol that most of the devices used, is composed by 32 bits. Which means that the IPV4 only allows 4.294 Million users to have a single IP address. It seems a large number but if we take in consideration the number of computer, mobile devices such as 3G/4G phones and tables, it makes this number small and not possible to satisfy all the devices, so the number of devices already overtakes this number. To overtake the lack of IP addresses the solution found was to use a simple solution, is the use of NAT protocol, which allows several devices on a network to access to the internet via only one public IP. Which means that for each network with several devices there is only one public IP, so inside of that network there are created the number of the devices the number of private IPs, making the use of IPV4 addresses possible.

The NAT is located normally on the routers that access to the Internet, as we can see on ***Figure 5-3.*** The Host A is located on the Network A, and wants to communicate to the Host B located on the Network B, Host A has the private IP: 192.168.1.3 and sends the packet through the router that has the NAT. The NAT translates the Host A private IP into a public IP of the Network A which is 10.1.1.2 and it sends over the internet to Network B.

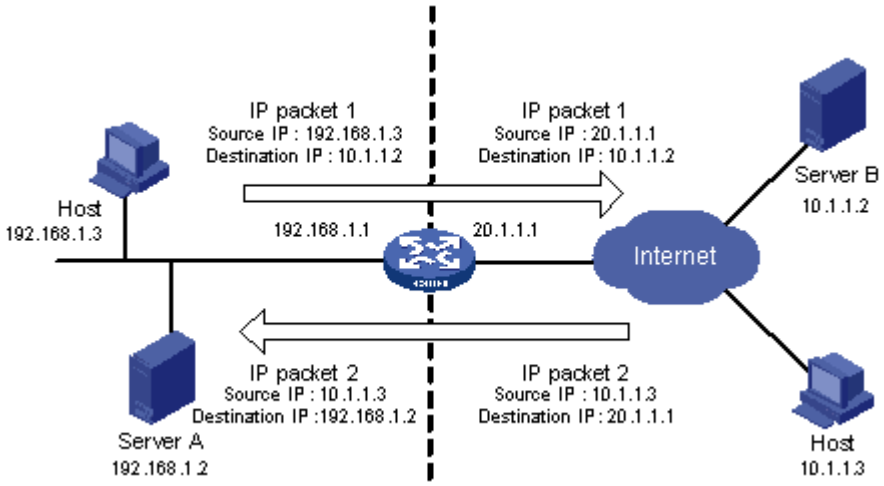

Figure 5-3 - *NAT Protocol Example Source:****Error! Reference source not found.***

However, this solution creates another problem that is critical to our system; when two devices want from different networks want to communicate with each other. They cannot do it, because they have a private IP associated to their machine, there is solutions to overtake the NAT problem, such as the use of TURN, STURN and ICE **Error! Reference source not found.** however, none of those solutions is used, because of their complexity. To overtake this problem the creation of the server on a public IP, resolves this problem, as we can see on ***Figure 5-4*** both devices don't need to know each other public IP or private IP they only need the server public IP, and with the combination of Ports the server knows who is communication with who.

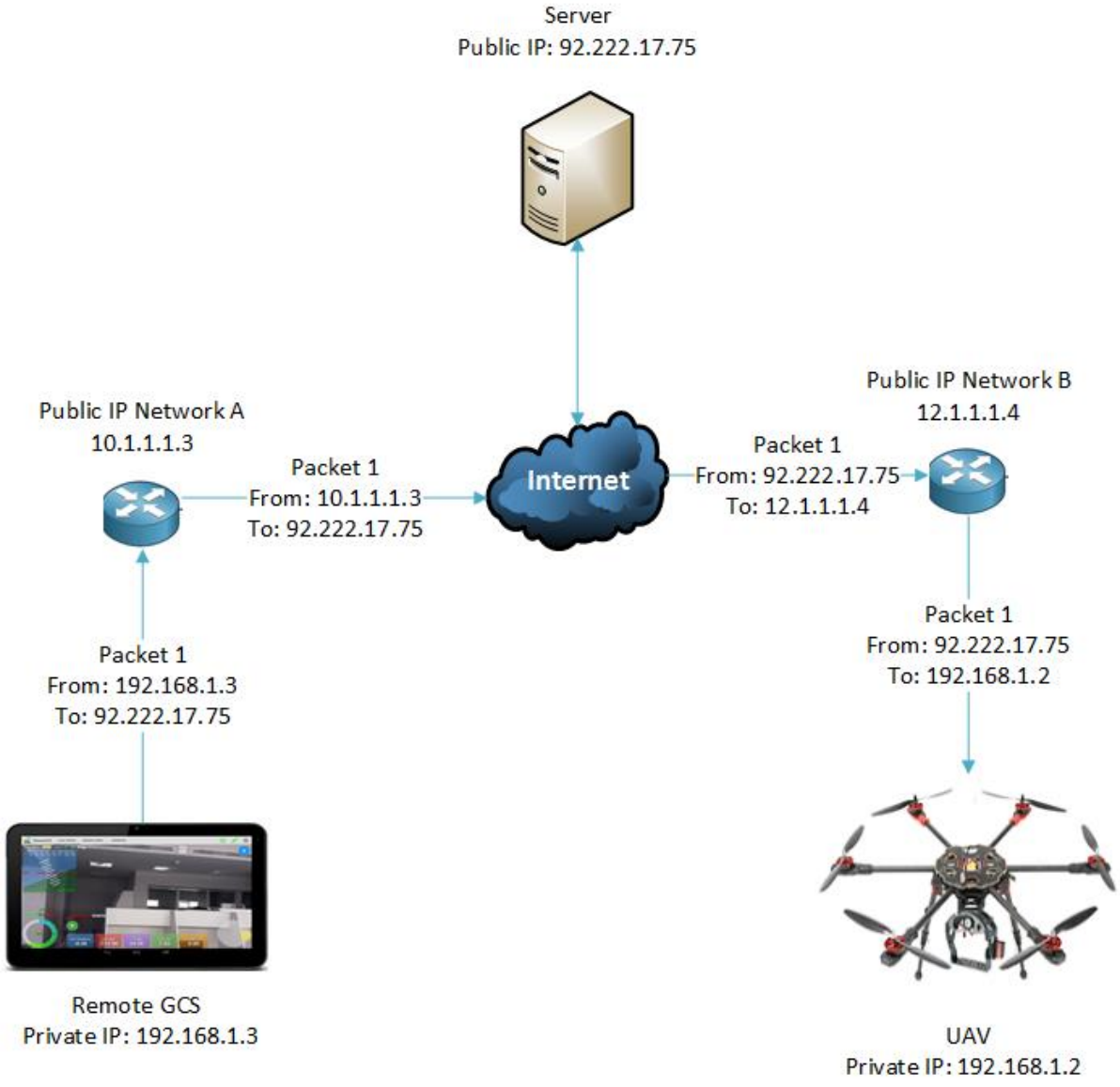

Figure 5-4 *– System NAT Overtake Example*

5.5 MAVLink

MAVlink is the protocol used for communication and data structure for communicate with the Electronic Processing System from the Server to the UV.

The MAVlink packet as it show on ***Figure 1-2,*** each message is sized between 8 and 263 bytes, and it is composed by nine fields. The first byte (STX) indicates the start of a new message, it is characterized by the 0xFF value, which is the same as 254, and the next byte (LEN) is the payload length. Each message has a sequence number, and the SEQ byte is used for that purpose, this field helps in case of packet lost, which can be detected in case of missing one number in the sequence. The SYS value or System ID value, indicates the UV id that the message is sent, the value can be between 0 and 255, which means that a single GCS can control up to 255 vehicles at same time. Next byte is the COMP byte or the Component byte and is the identifier of the component where the message is meant to go, for instance, if the message is to control the UV it has the Pixhawk COMP value, but if is to autonomous flight, it sends another COMP value. The MSG value represents the Payload ID, which is the meaning of the payload. Finally, the last bytes CKA and CKB are for checksum.

MAVLink can be divided in two protocols, waypoints protocol which is the responsible to the autonomous control, driven by waypoints, and the parameters protocol that are the UV direct settings.

5.5.1 Waypoints Protocol

The waypoint protocol objective is to read or write a Mission. A mission is an ensemble of items, and an item is a waypoint. Each waypoint is has an array of parameters such as geo location or the command to execute.

To read a mission as we can see on ***Figure 5-5***, which is the GCS requesting to the Pixhawk a pre-programmed mission, it starts with the GCS sending a WAYPOINT_REQUESTS_LIST, which means that the GCS is asking to the Pixhawk how many items does it has, the Pixhawk replies with WAYPOINT_COUN (N) with the number of items. After this first contact, the GCS is ready to request each item, sending the WAYPOINT_REQUEST (N) which N represents the waypoint sequence number, for instance, in the case of a Mission with 2 items, the GCS sends WAYPOINT_REQUEST (0) and the Pixhawk replies WAYPOINT (0), next the GCS sends WAYPOINT_REQUEST (1) and the Pixhawk replies WAYPOINT (1). In this moment, the GCS already has two waypoints so it knows there is no more waypoints for this

mission and ends the communication sending a WAYPOINT_ACK, informing the Pixhawk that it received all the items for the mission.

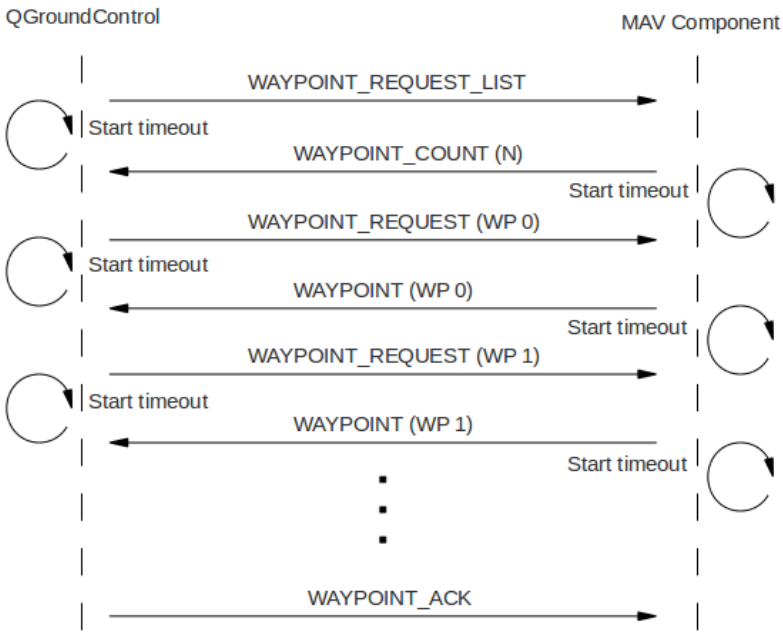

Figure 5-5 *- MAVLink Read Waypoints. Source:* ***Error! Reference source not found.***

In the case of the users intents to clear the actual Mission in has the support from this protocol, it is only need it to send a WAYPOINT_CLEAR_ALL and the Pixhawk replies with the WAYPOINT_ACK

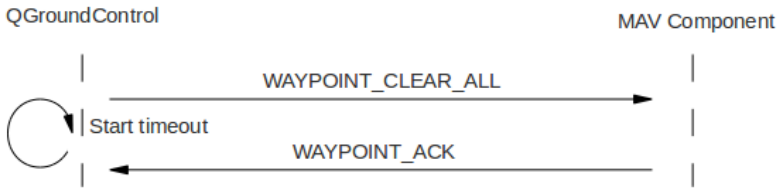

Figure 5-6 *- MAVLink Clear Waypoints. Source:* ***Error! Reference source not found.***

For the write mission as we can see on ***Figure 5-7,*** the intent is the GCS to send a mission and be stored in the Pixhawk, it starts to send a WAYPOINTS_COUNT (N) with the number of items to be stored, and the Pixhawk starts to request to that items, sending the WAYPOINT_REQUEST (N). When the number of items match with the number of

WAYPOINTS_COUNT (N) sent by the GCS the Pixhawk sends the WAYPOINT_ACK message warning the GCS that it received the mission successfully.

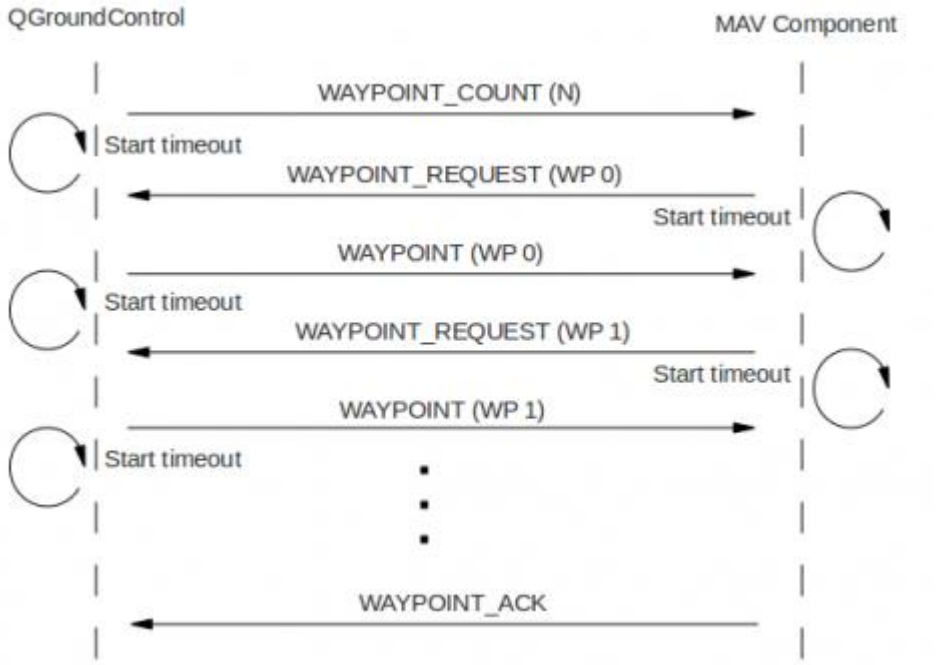

Figure 5-7** - MAVLink Write Waypoints. Source: **Error! Reference source not found.

5.5.2 Parameters Protocol

Not much different from the Waypoints protocol, the parameters protocol allows the user to write and read the UV parameters. A parameter is a UV setting, for example, the UV SYSID is one of the UV parameters.

To read all the parameters, the GCS starts to send and REQUEST_PARAMETERS message, resetting the parameter index to zero, which means that until it gets all the N parameters, the GCS requests the parameters. Pixhawk starts to send the parameter with the value, until it reaches all the parameters. If it misses one or more parameters, the GCS restarts the process until it gets all the parameters, in the case of UAV there are 418 parameters, until the GCS has the 418 parameters it keeps requesting to the Pixhawk. There is no way of requesting one parameter, when requesting the GCS has to request them all.

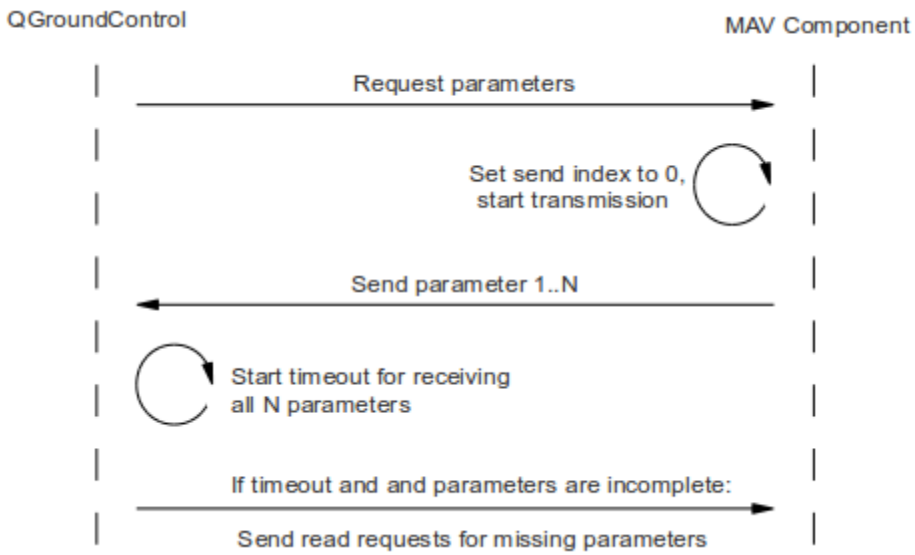

Figure 5-8** - MAVLink Read Parameters, Source:* ***Error! Reference source not found.

When the user changes a parameter, there is the need to write the new value for that parameter, so different from the reading there is no need to write all the parameters, there is only need to write the changed parameter. To accomplish that, the GCS sends to the Pixhawk the changed parameter and the Pixhawk replies with the changed parameter,, if it matches with the parameter sent, the writing is completed.

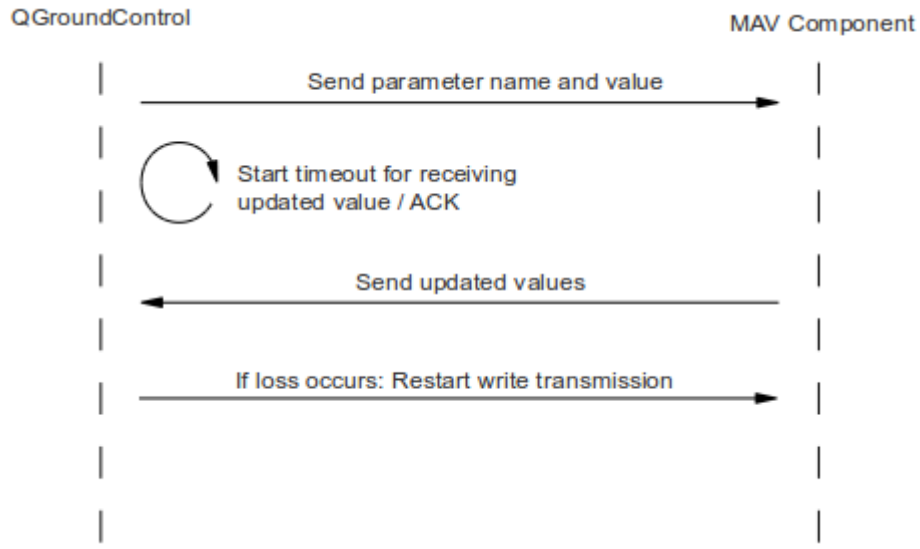

Figure 5-9** - MAVLink Writing Parameters, Source:* ***Error! Reference source not found.

Chapter 6
RESULTS AND EVALUATION

Does the system fulfill the requirements to a real-time operation? What results were collected and how are they related with each other?

6.1 Overview

In order to obtain the results there was the need to have features that allow the analyses of the system, two different forms collect those features: one log file from the Remote GCS, which has the UAV location (latitude and longitude), speed, the system latency (ping). The telemetry bit rates, such as the input bitrate the telemetry messages that the Remote GCS sends to the UAV and the output bitrate the messages that the UAV receives from the Remote GCS, as well as the stream bitrate. Finally the Packet Error Ratio (PER) which is the percentage of packets that the Remote GCS did not received from the UAV.

Another way to collect features is by using the Signal Manager application ***Figure 6-1***, that is an application for android, and it runs on a phone that is going to be placed on the UAV during the flight in order to collect data. The UAV does not need this phone to be operational, the intent is only to collect data. The application handles with the network strengths, collects the Mobile and WIFI Network strength, identifies the Mobile technology and cell location ***Figure 6-2***, which can be useful for analyzing the handover problems, and finally collects the UAV location that can be merged with the location from the Remote GCS to make an accurate location.

Figure 6-2 – *Signal Manager Application*

Figure 6-1 *– Signal Manager Cell Location*

6.2 Ping, PER and Bitrates

In order to calculate the system latency, a simple protocol has been created (***Figure 6-3***). Ping is the time that takes a message to make the route, from the Remote GCS to the UAV back to Remote GCS. A Ping message leave the Remote GCS with a frequency of one second, goes to the server and goes to the UAV and the UAV replies with PING ACK. When the server receives the PING ACK from the UAV and adds some information to the message, the total messages received, and sends the PING ACK to the Remote GCS. The Round Trip Time (RTT) is calculated with the sum of those times (***Formula 1***).

$$RTT[ms] = t_{RemoteGCS/Server} + t_{Server/UAV} + t_{UAV/Server} + t_{Server/RemoteGCS} \quad 1$$

The Remote GCS makes that sum, calculates the Ping, and with the extra info sent by the server it calculates the PER. The PER is the result with the difference between the total messages received from the server with the total messages sent by the Remote GCS (***Formula 2***).

$$PER[\%] = \frac{ErrorPackets * 100}{TotalPacekts} \quad 2$$

The Bitrates are calculated at the end, after the disconnect the Server sends to the Remote GCS the total of messages sent and received to the UAV, and the Remote GCS calculates with the flight time, the input bitrate is the messages received divided by the time and the output bitrate is the messages sent divided by the time.

$$Bitrate\ IN[kbps] = \frac{ReceivedPacekts}{FlightTime} \quad 3$$

$$Bitrate\ OUT[kbps] = \frac{ReceivedPacekts}{FlightTime} \quad 4$$

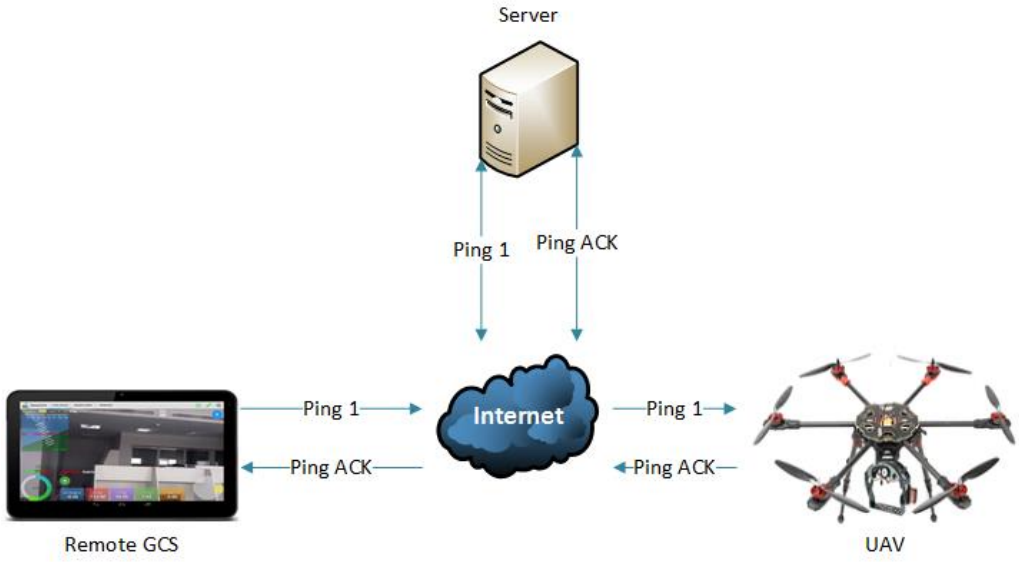

***Figure 6-3** – Ping (RTT) Messages Protocol*

6.3 Scenarios

To test the system, three different scenarios where tested. Performing real flight using all the electronics described in the previous chapters.

The first scenario was Wi-Fi on both terminals, Remote GCS and the UAV. This test was made in a controlled place, with a strong and constant Wi-Fi signal, and all the tests performed with Wi-Fi were performed inside of a house. The other two tests where made outside, with a previous study about the mobile coverage it was accomplish to get two areas where we could get two different types of network, 3G and 4G. As we can see on ***Figure 6-4,*** the base station located near the Remote GCS has the two different technologies: 3G and 4G, allowing the Remote GCS to operate using any technology.

Observing the ***Figure 6-4*** we have the two scenarios tested outside, in red we have the 4G scenario. In this scenario, both the UAV and the Remote GCS were only using 4G to communicate.

The second test, in blue is represented the 3G test, and it was only used 3G technology to communicate on the Remote GCS and the UAV.

All the four base stations located on the image are dual base stations, as we can see on ***Table 6-2***, which means that they were 3G and 4G, so to get free independent results from different technologies the terminals were filtered to use only a specific technology. However in the 4G route, some 4G to 3G handovers were made in order to test the technology handover.

Figure 6-4 *- Flight routes with the Base Station Locations*

6.4 Results Analysis

In order to have a good analysis about the system, several features were collected on data tests. Such as: RTT; Speed; Packet Error Ratio (PER); Received Power (Prx); Cell ID that are for handovers analysis; Bitrate in, which are the messages that came from the UAV to the Remote GCS. Bitrate out, which are the messages that goes from the Remote GCS to the UAV and finally the Bitrate Video, which is the bitrate from the Video to the Remote GCS.

The first feature analysed is the speed, as it can be seen on ***Figure 6-5*** in was only measure for the 3G and 4G test because on the Wi-Fi scenario the UAV did not perform any flight, so there is no speed data.

As we can see on ***Figure 6-5*** the UAV reached speeds of maximum of 24 m/s or 86 km/h in the 3G scenario. This was intent to verify if the speed influence the system performance, as

we can see on ***Figure 6-6*** it did not made any difference, the RTT stayed the same. So for both scenarios we can conclude that speed is not a direct feature for make the system viable. It can be proved that the system can handle high and low speeds.

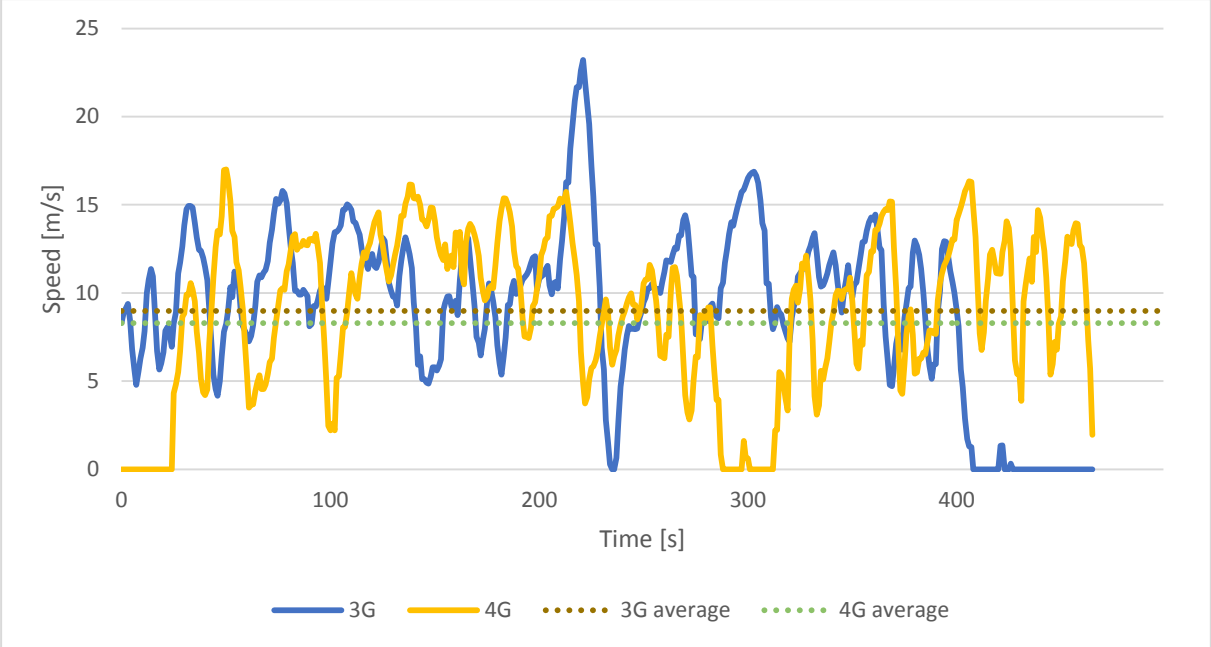

Figure 6-5 - *Speed vs Time Graphic*

In terms of RTT, which is the time that a message takes from the Remote GCS to the UAV and return, we can see on ***Figure 6-6*** that the best scenario is Wi-Fi with the average of 56 ms, which is the best case scenario. For the 3G is a bit higher with 100 ms and for the 4G with 70 ms.

Even with the highs of 300 ms, the system can handles well with RTTs of 500 ms, higher than that makes it instable. So we can conclude based on the RTT analysis that the system can handle even with high RTT values.

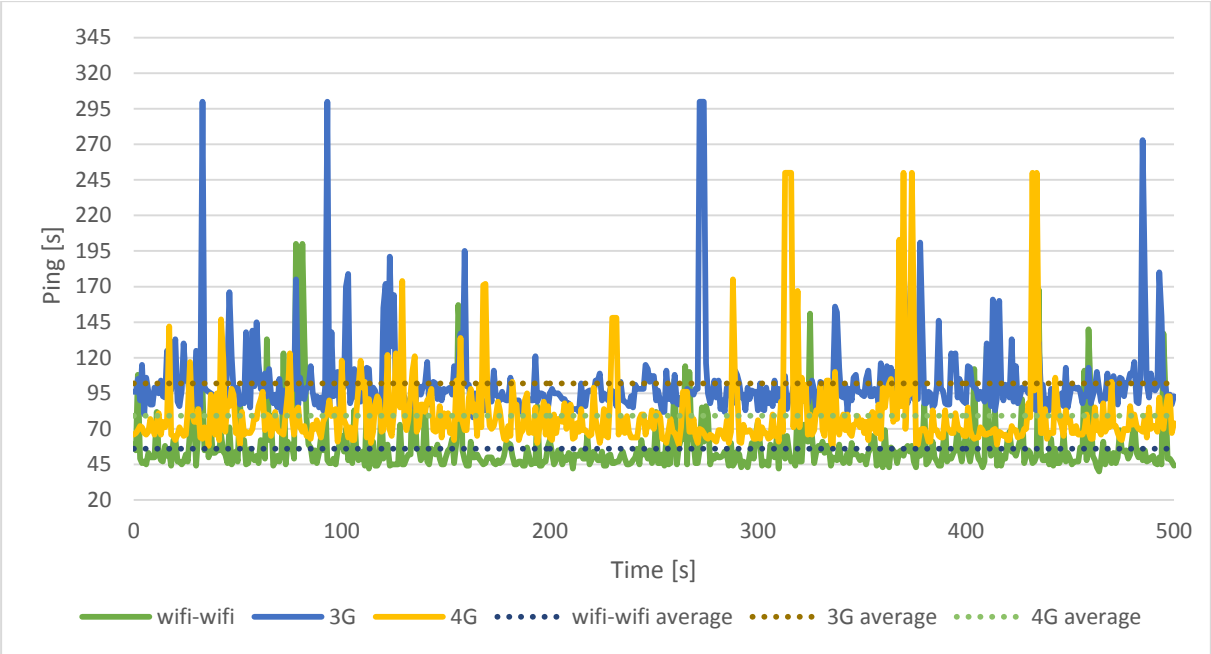

Figure 6-6 *- RTT vs Time Graphic*

To compare different scenarios, analysis and comparisons of empiric parameters of RTT samples were made. The delay, D, for a given percentage of samples, d_p, is obtained through the RTT Cumulative Probability Density Function (CPDF), $F_D(d_p) = 50\%$ and $F_D(d_p) = 90\%$:

$$F_D(d_p) = \int_0^{d_p} p_D(d)\, d_d \qquad 5$$

Where $p_D(d)$ is the RTT Probability Density Function (PDF) and the dispersion of results is obtained using the standard deviation:

$$\sigma_D = \sqrt{\mathrm{E}[D^2] - \mathrm{E}[D]^2} \qquad 6$$

Where $\mathrm{E}[D]$ represents the average value of RTT:

$$m_D = \mathrm{E}[D] = \int_0^{+\infty} D \cdot p_D(d)\, d_d \qquad 7$$

In addition, $\mathrm{E}[D^2]$ represents the second order moment:

$$\mathrm{E}[D^2] = \int_0^{+\infty} D^2 \cdot p_D(d)\, d_d \qquad 8$$

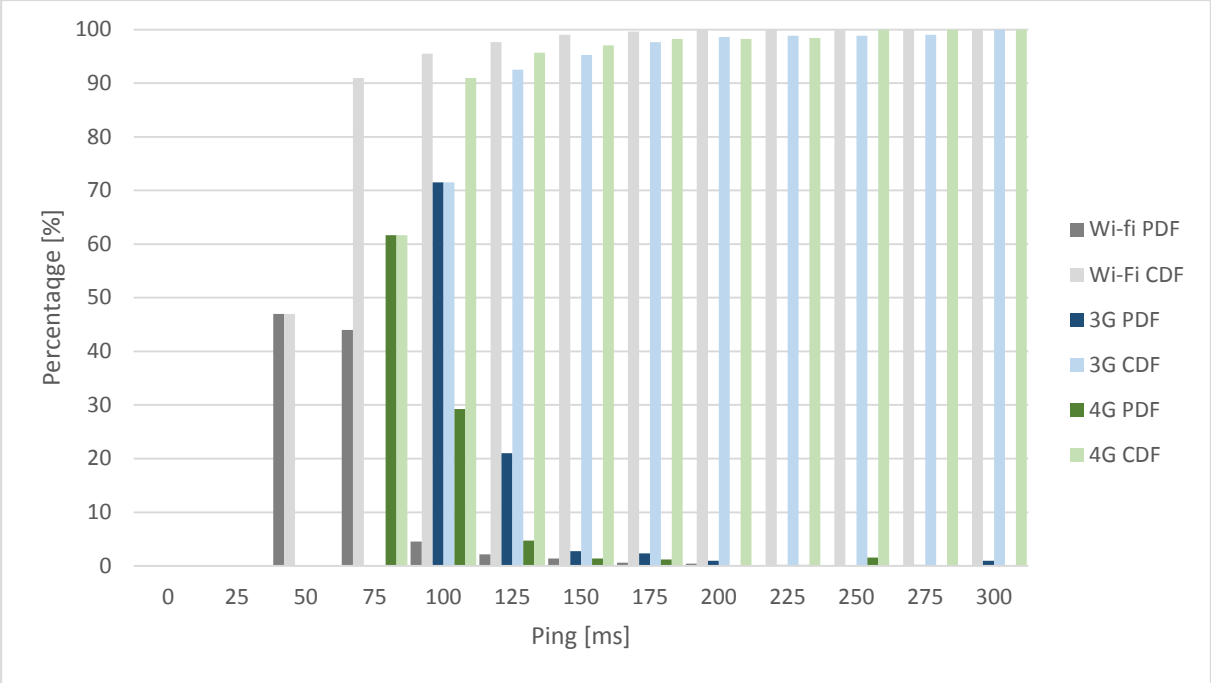

***Figure 6-7** - Cumulative Distribution Function of RTT*

Scenario	50%(a)	90%(a)	m_D	σ_D
Wi-Fi	55	75	56	19
4G	75	100	79	27
3G	85	125	101	28

a. The delay for a given percentage of samples for each scenario

***Table 6-1** - RTT Performance Indicators for Each Scenario*

By observing the data summarized in ***Table 6-1*** and the ***Figure 6-7,*** it can be seen that the Wi-Fi scenario is the most advantageous, however 4G and 3G scenarios also have very interesting results and allows a real-time control. All these results show dispersion in the same order of magnitude despite the Wi-Fi scenario having a delay in average much lower than 3G and 4G scenarios. 4G scenario comparing with 3G have better results, but comparing with the

Wi-Fi, are a bit higher which lead us to conclude that cellular network is the main responsible to increase delays.

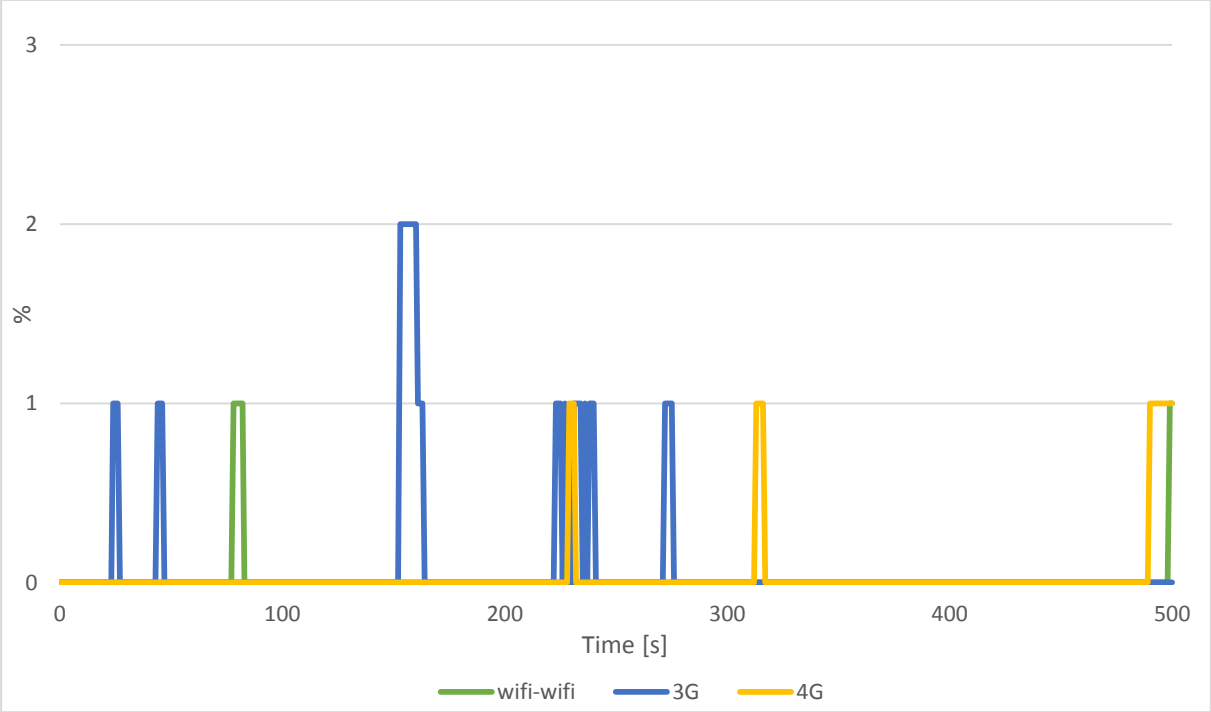

***Figure 6-8** - PER vs Time Graphic*

In terms of Packet Error Rate, the system behaved well to the tests, having an insignificant value on both scenarios, with a PER on an average of 1%, which means that 99% of the packets arrived without any errors. However some conclusions can be taken from the ***Figure 6-8***, that the PER is directly connected with higher values of RTT, when the RTT has higher values the PER rises, but with an insignificant value.

We can conclude that the system is capable enough to resist to the adversities imposed by the network.

In terms of Received Power (Ptx), as we can see on ***Figure 6-9*** that Wi-Fi achieves higher power than the other two scenarios. Like it was mentioned the Wi-Fi test was made in perfect conditions inside of a building with constant Wi-Fi power, and the variation on the power it is only consequence of the movement of the UAV through the building.

In terms of the mobile tests the results are similar, but with some differences. The 4G scenario achieved higher Ptx values because it was near the base stations, however because it used the 4G technology had more handovers than the 3G scenario, as we can see on **Figure 6-10,** this can explain the oscillation on the values.

A good example that the system is robust enough is in the 3G scenario, the Ptx has the value of -90 dBm on 200s, and as we can see on ***Figure 6-6,*** the system did not have higher RTT values because of the Ptx value.

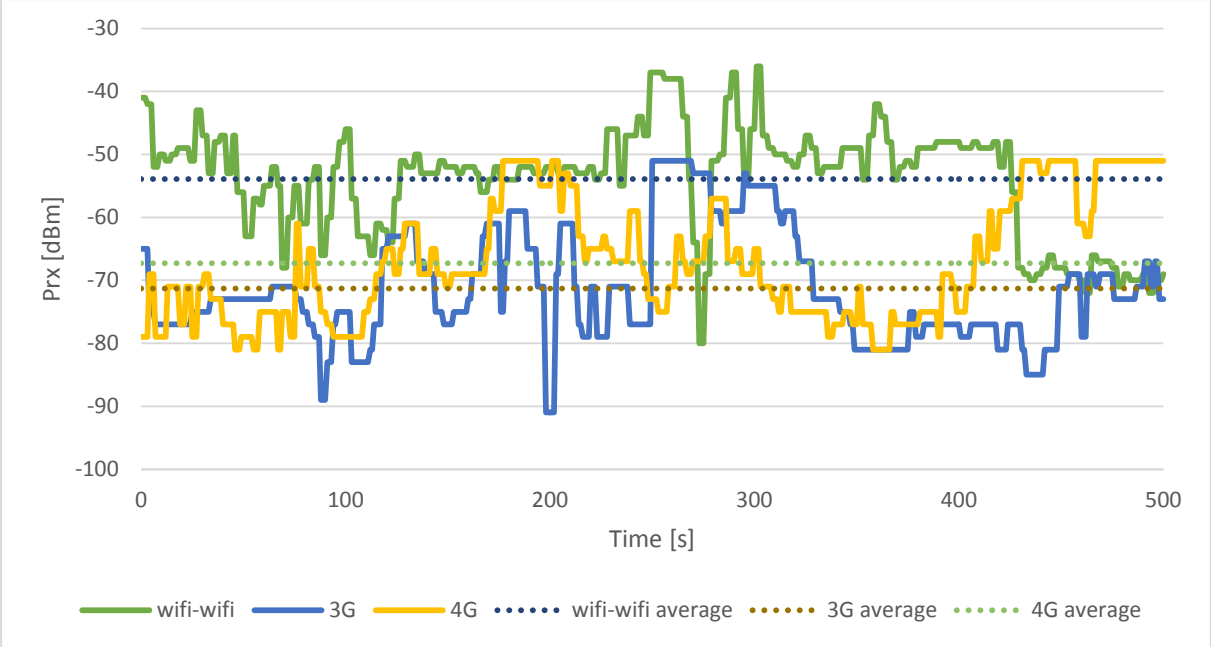

Figure 6-9 *- Received Power vs Time Graphic*

Handover is the name given when a mobile device switches from one cell to another cell, and to test our system is important to analyze this feature.

As we can see on ***Figure 6-10*** we have the cell ID comparing with the time, the cell ID is not relevant for the analysis, it is only significant for the base station location. Doing a further analysis on the figure, we can conclude that 4G performs more handovers than the 3G. In numbers, 4G performs 16 handovers and 3G performs 6. This happens because of the network architecture, 4G more known as LTE, only had two base stations on the test location, so in order to keep the system alive it made some handovers to the 3G technology, as we can see on ***Figure 6-9***, some values are closer to the 3G, because of that as well.

It is important to separate the 3G technology used. on the 3G scenario only HSPA+ and UMTS was used, so all the values are based on those technologies. On the 4G scenario most of the technology used was LTE, however in certain parts of the route it was impossible to use only LTE, so some handovers were made to other technologies. LTE architecture makes the system faster, as it was already proved looking the RTT and PER results.

It is interesting to analyze the difference in handovers on the 4G scenario between technologies. When the handover is made between LTE and UMTS the Ptx has a less

decrease than the LTE and HSPA+ handover. On ***Figure 6-10*** this can be seen at 250s, which is a LTE to UMTS handover, and the 300s which is a HSPA+ to LTE handover. This happens because UMTS belongs to the same group as the LTE, the 3rd Generation Partnership Project (3GPP) and the LTE architecture handles the UMTS faster and more efficient than other technologies such as the HSPA+. However these variations are not significant at all to disturb the functionality of the system, analyzing both the ***Figure 6-11*** and ***Figure 6-10***, and in terms of RTT, it is not noticed, so we can conclude that the system handles well the handovers.

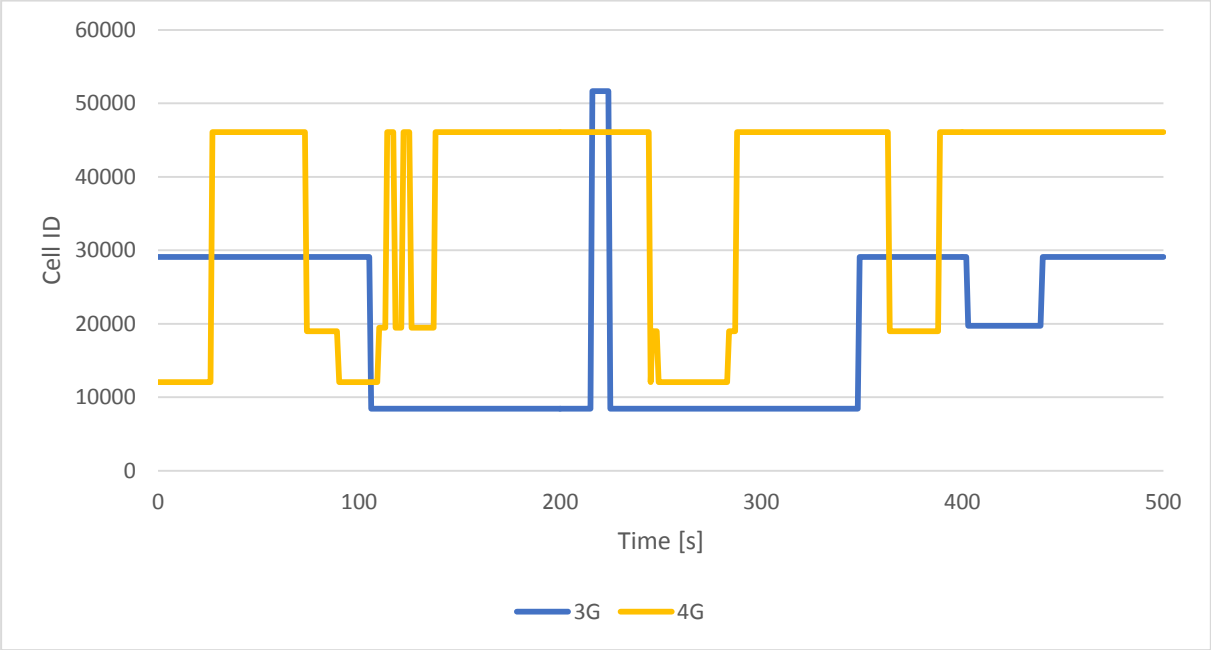

Figure 6-10 - *Mobile Cells vs Time Graphic*

Cell ID	LAC (a)	MNC (b)	Carrier	Technology
29073	4100	3	NOS	UMTS/HSPA+
8421	4100	3	NOS	UMTS/HSPA+
51671	4100	3	NOS	UMTS/HSPA+
19723	4100	3	NOS	UMTS/HSPA+
12053	1	3	NOS	LTE
46103	1	3	NOS	LTE
18965	4100	3	NOS	UMTS/HSPA+
12053	1	3	NOS	LTE

a. Location Area Identity (LAC)
b. Mobile Network Code (MNC)

Table 6-2 - *Cells Information*

In terms of Bitrates, we can analyze **Figure *6-11***, ***Figure 6-12*** and ***Figure 6-13*** together. Starting for the Outgoing Bitrate or the messages that the Remote GCS sends to the UAV, we can conclude that the system follows the same logic. With Wi-Fi it has a higher bit rate than the others, however as we can see on the ***Table 6-3*** the average value is not that much different we are talking of 200 bytes of difference, and as we remember 200 byes is the size of one MAVlink message, so it is not a significant difference.

In terms of Incoming Bitrate or the messages that the Remote GCS receives from the UAV, again the results follow the same logic, Wi-Fi with better results and 4G with slight better results than the 3G, however the average value converge to the same value, so both scenarios are good.

Finally the Video Bitrate, which is the Bitrate of the video that the Remote GCS receives. There is a difference on this feature, the 3G and 4G scenario the Video Bitrate can achieve higher values than the Wi-Fi scenario, those differences are not explained by the network itself but can be explained by the type of video recording, on the Wi-Fi scenario (indoor) there was less objects to record, less FPS changes which means the image was smaller. In the outdoor scenarios (3G and 4G), there was a constant movement of the UAV, with lots of vibration and obstacles to record, which made the frames larger, meaning larger images and more bitrate necessary to transmit. However, in both three scenarios the video quality was good, providing the real time control possible, even with some buffering, it never put the UAV control in cause.

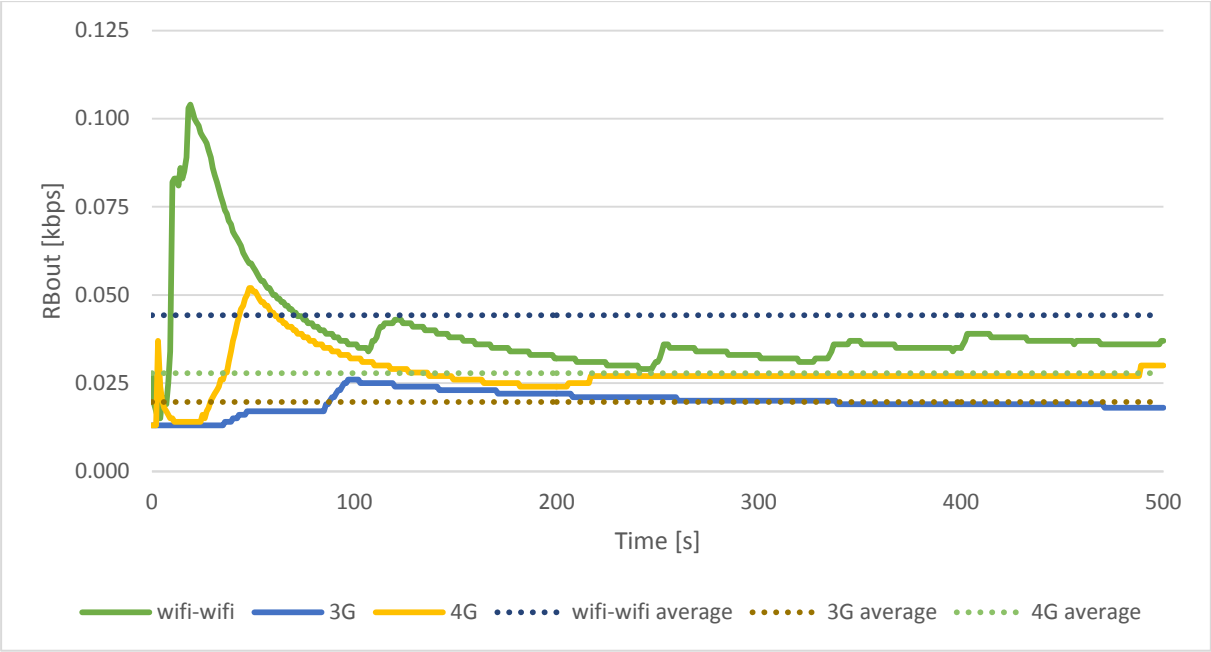

Figure 6-11 *- Outgoing Bitrate vs Time Graphic*

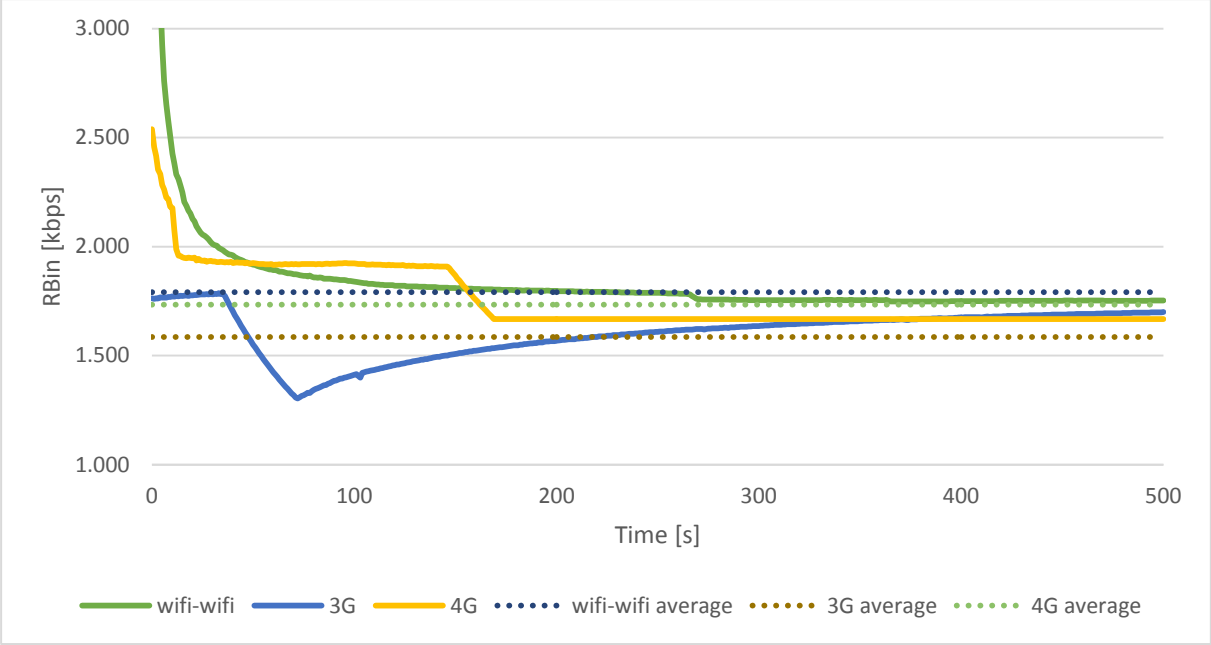

Figure 6-12 *- Incoming Bitrate vs Time Graphic*

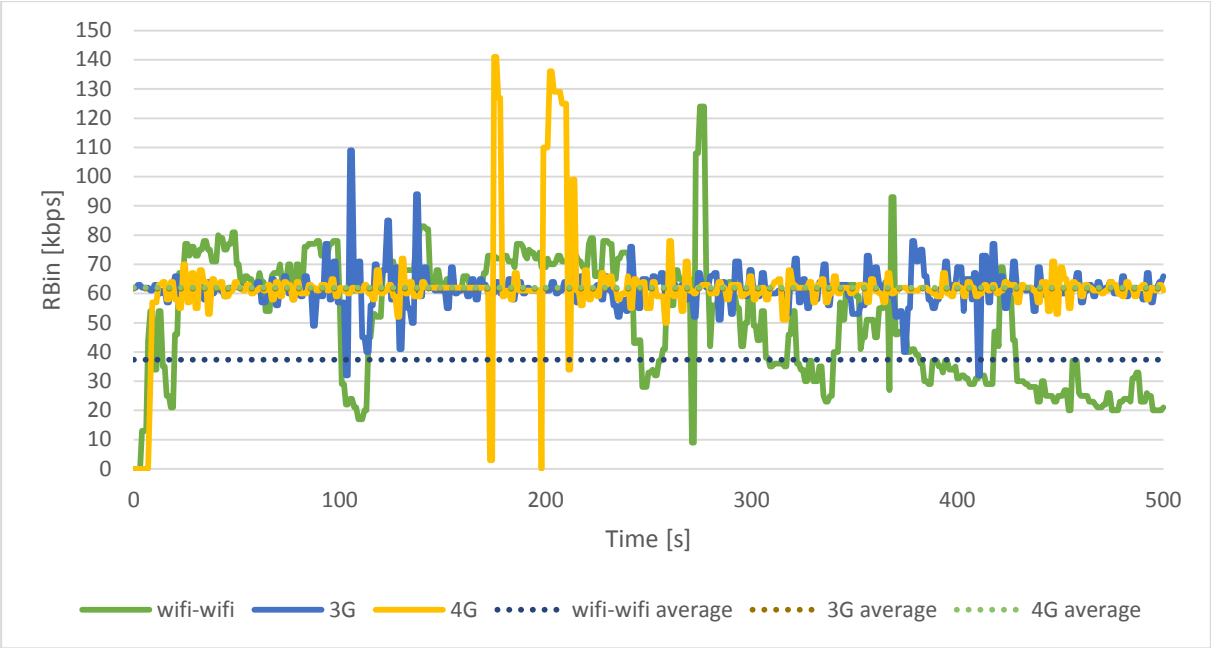

Figure 6-13 *- Video Bitrate vs Time Graphic*

Scenario	RTT [ms]	Bitrate Out [kbps]	Bitrate In [kbps]	Bitrate Video [kbps]	Speed [m/s]	Received Power [dBm]	Handovers
m_DWi-Fi	56	0.04	1.79	37.36	-	-53	-
m_D 4G	79	0.02	1.73	62.04	8.3	-67	16
m_D 3G	101	0.01	1.52	61.64	9	-71	6

Table 6-3 *- Results Average Table*

Chapter 7
CONCLUSIONS AND FUTURE WORK

Were all the goals achieved? What conclusions can be taken after all the developed work? What improvements can be done in the future?

7.1 Conclusion

The main goal of this thesis was to build and test a system capable of controlling a UAV using the heterogeneous networks.

Various applications were developed to achieve this goal, first the Remote GCS application that is capable not only to control a UAV but also to control other types of UVs. In addition, a server capable of managing several UVs at a time, and provides the solution for the NAT problem.

A new protocol was developed in order to transport the communications through the server; UDP was used for transportation. However, some messages required some confirmation because they are critical to the performance of the system, such us the Throttle and the steering control. An electronic processing system was developed in order to make all the software work, and it has to be place on the UV. For all of those components several conclusions can be taken:

- Remote GCS is user friendly and easy to control. The video has almost no delay, even with handovers, the video does not suffer any long buffering, even if that happens it automatic restores to the actual time. In terms of commands even with the worst case scenario (3G) it is still functional, and provides an instantaneous command, which is important to the control of the UV.
- The Server provides a full control to the UVs, allowing to overtake the NAT problem, which is a simple way that eliminates the delay of using others solutions.
- The use of the Raspberry 2 had a big improvement, it made the system processing way faster in order to handle the control and the video on the same component.
- Average RTT values of 70/100 ms allow a perfect control system on both 3G and 4G technology, using Wi-Fi 40/50 ms.
- From the test scenarios, we can conclude that using Wi-Fi the system handles really well, however it was the best scenario possible, with no other variables on the system. When tested outside the (4G) achieved results much better than the 3G in all the tests.
- Handovers do not affect the system, as we can see on Chapter 6 several handovers were made and the system was capable to sustain the communication without suffering any consequence.

- 3G, which was the worst possible scenario, was still a viable option to control the UV, which we can conclude that all the scenarios are viable option to control safely any UV.

7.2 Future Work

For the future work, some new features and improvements can be made in order to improve the system:

- Implementation of the First Person View (FPV), which allows the user to see what the UAV see, with the use of oculus rift technology.
- Implementation of new fail-safes, in order to protect the UAV integrity.
- Improve the Remote GCS application visually to be friendly to other UVs.
- Creation of a database to store all the media and data collected from the UVs.
- Creation of a UAV using new and light materials, and make it cheaper and more affordable.
- Creation of a better security system of the messages.