ABSTRACT

The Unmanned Vehicle Systems (UVS) are growing in large proportions and are an emerging technology but there are some limitations that still need to be overcome in order to get the best effectiveness. One of the limitations is the reception area for vehicle operation. Most vehicles operate in line-of-sight (LOS). A very significant improvement would be to allow behind line-of-sight (BLOS) operation via wireless networks. This thesis proposes to overcome this difficulties using wireless networks: Wi-Fi and third generation (3G) and fourth generation (4G) of mobile networks to operate an Unmanned Ground Vehicle (UGV), also called a rover. This way, the vehicle will not have a theoretical range operation, turning into a vehicle operated in BLOS. This thesis also includes the study of the reliability and efficiency of wireless heterogeneous networks solution for UGVs operation in real time.

As prove of concept for described objectives, an Unmanned Ground System (UGS) was developed. This system is capable of control and monitoring multiple vehicles through wireless networks: Wi-Fi, 3G and 4G mobile networks. The communication between the vehicle and the operator is made through a ground control station (GCS), an Android application running on a mobile device. Its aim is to centralize all the information from the vehicle(s). Present on the vehicle is a Raspberry Pi (RPI) acting like a proxy, thus enabling the communication between the GCS application and the UGV's board controller, the Ardupilot Mega (APM). The RPI receives telemetry from APM via Micro Air Vehicle Communication Protocol (MAVLink) and captures video through an external camera and sends it all to the GCS.

In order to evaluate the system performance, several tests were done for data collection, namely values of network latency, bitrate, packet error ratio (PER) and signal strength varying with speed, altitude and location of the vehicle.

***Keywords*:** Ardupilot, MAVLink, multiple UGVs, Wireless networks, mobile devices, Raspberry Pi

This page intentionally left in blank

RESUMO

Os *Unmanned Vehicle Systems* (UVS) estão em crescimento exponencial e, como tecnologia emergente, existem algumas limitações que necessitam de ser ultrapassadas de modo a obter uma maior eficácia dos mesmos. Um bom exemplo é o raio de alcance para operação do veículo. Como muitos operam em *Line-of-Sight* (LOS), uma melhoria significativa seria passar a operá--lo B*ehind Line-of-Sight* (BLOS). Esta dissertação tem como objectivo ultrapassar esta limitação, recorrendo a redes sem fios: Wi-Fi e 3ª (3G) e 4ª geração (4G) de redes móveis para operar um *Unmanned Ground Vehicle* (UGV), também chamado rover. Desta forma o veículo não terá, teoricamente, um limite de alcance no seu manuseamento. Esta tese também inclui o estudo da fiabilidade e eficácia da solução das redes sem-fios heterogéneas para operar um UGV em tempo real.

Como prova de conceito foi desenvolvido um *Unmanned Ground System* (UGS). Este tem capacidade de controlar e monitorizar múltiplos veículos através das redes sem-fios: Wi-Fi e redes móveis 3G e 4G. A comunicação entre o veículo e operação é estabelecida através de uma *Ground Control Station* (GCS), nada mais que uma aplicação desenvolvida no sistema operativo Android, executada num dispositivo móvel. No veículo está presente um Raspberry Pi (RPI) a actuar como um *proxy*, permitindo o estabelecimento da ligação entre a GCS e a placa de controlo do veículo, um Ardupilot Mega (APM). O RPI recebe telemetria vinda do APM através do protocolo de comunicação *Micro Air Vehicle Link* (MAVLink), captura vídeo através de uma câmara e envia ambos para a GCS.

Por forma a avaliar a *performance* do sistema, foram executados vários testes para recolha de dados como: latência de rede, ritmo binário, *Packet Error Ratio* (PER) e potência de sinal. Este foram analisados com a variação da velocidade, altitude e localização do veículo.

***Palavras-chave*:** Ardupilot, MAVLink, múltiplos UGVs, redes sem-fios, dispositivos móveis, Raspberry Pi

This page intentionally left in blank

ACKNOWLEDGMENTS

In first place I would like to thank to my family for the opportunity to get the master's degree, especially my mother, father and sister. The transmitted education was central to finish this dissertation successfully, winning all proposed challenges and all the rest showed up.

I want to thank to my girlfriend who was always present and supported me when I needed the most. Her wise words and advices were a great source of motivation to finish this dissertation. Above all, their friendship was essential at various points along this journey.

Of course, this whole project wouldn't be possible without the participation of Professor Pedro Sebastião. Always present and active, he helped me out with technical problems inherited to the project (hardware, software and theoretical questions). Their knowledge served often as a key in solving problems related to the proposed objectives. All the work conditions for the realization of this thesis were possible due to its action in an attempt to obtain and always offer the best and most appropriate resources for the development of the project. I want to acknowledge him not only for this labor issues but also for the support and wise words on personal issues. His advices were also a great help and came always on good time.

Also I want to thank Professor Nuno Souto for his technical advices and transmitted knowledge in all areas covered by the developed work.

I would like to thank the Instituto de Telecomunicações because it was part of this project, by helping with the necessary material and physical space to work.

A special thanks to my good friend and colleague Tiago Saraiva, who helped me out with technical questions and for being always present, giving me support.

Another special thanks to my good friend and colleague Gonçalo Horta for his wise words and advices, both in labor and personal issues.

A acknowledge must be done to Autoeuropa/Atec group, namely to Eugenio Bastos, Ricardo Silva, Carlos Isidro, Paulo Galvoeira and José Peniche, for their availability in terms of time and assigned material.

For last but not less important, I want to thank to Huawei engineers Nuno Bernardo and André Mei for their help, active participation, readiness to help and important material for project development.

This page intentionally left in blank

ACRONYMS & ABBREVIATIONS

2WD	Two–Wheel Drive
3DR	3DRobotics
3G	3rd generation
4G	4th generation
4WD	Four–Wheel Drive
AC	Alternating Current
ACK	Acknowledgment
AP	Access Point
API	Application Programming Interface
APM	Ardupilot Mega
BEC	Battery Eliminator Circuit
BLOS	Beyond Line-of-Sight
BTS	Base Transceiver Station
CAN	Controller Area Network
CANH	CAN-High
CANL	CAN-Low
CDF	Cumulative Distribution Function
CIDR	Classless InterDomain Routing
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA/CD	Carrier-Sense Multiple Access with Collision Detection
DC	Direct Current
DC-HSPA	Dual Cell-HSPA
DLC	Data Length Code
ECU	Engine Control Unit
EDGE	Enhanced Data rates for Global Evolution
EOF	End-Of-Frame
ESC	Electronic Speed Controller
FFMPEG	Fast Forward Moving Pictures Expert Group

FPS	Frames Per Second
GCS	Ground Control Station
GNSS	Global Navigation Satellite System
GPLv3	General Public License version 3
GPRS	General Packet Access Service
GPS	Global Positioning System
GPU	Graphics Processing Unit
GSM	Global System for Mobile Communication
HSPA+	Evolved High-Speed Packet Access
I/O	Input/Output
I^2C	Inter-Integrated Circuit
ICE	Interactive Connectivity Establishment
ID	Identifier
IDE	Identifier Extension
IDE2	Integrated Development Environment
IETF	Internet Engineering Task Force
IFS	InterFrame Space
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IP	Internet Protocol
IRE	Introduction to Research in Engineering
ISO	International Standardization Organization
IT	Information Technology
ITU-R	International Telecommunication Union - Radiocommunication
LAC	Location Area Code
LED	Lightning Emitting Diode
LLC	Logic Level Converter
LOS	Line-of-Sight
LTE	Long Term Evolution
LTE-A	Long Term Evolution Advanced
MANET	Mobile Ad-hoc NETwork
MAV	Micro Aerial Vehicles
MAVLink	Micro Air Vehicle Link

MIMO	Multiple-Input Multiple-Output
MNC	Mobile Network Code
NAT	Network Address Translation
NRZ	Non-Return to Zero
OS	Operating System
OSI	Open Systems Interconnect
OSP	Open-Source Project
P2P	Peer-to-Peer
PCM	Pulse-Code Modulation
PDF	Probability Distribution Function
PER	Packet Error Ratio
PM	Power Module
PPM	Pulse-Position Modulation
PWM	Pulse-Width Modulation
PX4	Pixhawk
RC	Radio Control
RF	Radio Frequency
RPI	Raspberry Pi
RPM	Rotation Per Minute
RTR	Remote Transmission Request
RTT	Round Trip Time
SoC	System on Chip
SOF	Start Of Frame
SPI	Serial Peripheral Interface
STUN	Session Transversal Utilities for Network Address Translation
TCP	Transmission Control Protocol
TURN	Traversal Using Relays around Network Address Translation
UART	Universal Asynchronous Receiver Transmitter
UAV	Unmanned Aerial Vehicles
UBEC	Universal Battery Eliminator Circuit
UDP	User Datagram Protocol
UE	User Equipment
UGS	Unmanned Ground System

UGV	Unmanned Ground Vehicles
UI	User Interface
UMTS	Universal Mobile Telecommunication System
USART	Universal Synchronous Asynchronous Receiver Transmitter
USB	Universal Serial Bus
UST	Unmanned Systems Technology
USV	Unmanned Sea Vehicles
UV	Unmanned Vehicle
UVS	Unmanned Vehicle Systems
VoLTE	Voice over LTE
VPS	Virtual Private Server
VW	Volkswagen
W-CDMA	Wide-band Code-Division Multiple Access
WSE	Wowza Streaming Engine™

Chapter 1
INTRODUCTION

This chapter is directed to all aspects and concepts implied in the realization of this thesis, namely motivation, achieved objectives, scientific contributions and state of the art of the involved technological area. An organized reading is also present in a section relating to the structure of the document.

1.1 Overview

An Unmanned Ground System (UGS) consist on a land system without a human driver aboard that can be controlled remotely. Furthermore if the vehicles have some degree of Artificial Intelligence (AI) they can be self-adaptive to environmental conditions around, smart learning and drive autonomously [1]. In other words, a UGS is composed by the vehicle itself and all the components that are imperatives for its operation [1]. In the military environment, the greatest benefit of the UGSs (and the UVS in general) is the greater protection of the welfare of the operator since he is outside the vehicle [2]. The usage of UGV by the civilian community is growing due to its lower price (also have high prices), easy handling and versatility. For recreation times the remote controlled cars (toys) are a great example. In employment context, there are already some applications, however, are not so well known: land vehicles operating in industrial operations involving risk. An example is drilling in mines where there is risk of collapse, implementation of explosives in quarries, *etc*.

The control of the vehicle is made by GCS or by remote operator [3]. For this to be possible, the vehicle must have an electronic module that performs the orders given by one of the controllers, that is, receive, translate and send orders to the respective actuators, *i.e.* an Engine Control Unit (ECU). These necessary components are quite diverse, as described in [3]:

i. Navigation system: Global Positioning System (GPS), magnetometer, barometer and gyroscope, mostly known by Inertial Measurement Unit (IMU), constitution part of Inertial Navigation System (INS). These allows to monitor the vehicle;

ii. Communication system: they can be RC, satellite or wireless systems. These systems provides the connection between the vehicle and the GCS, even in full autonomy cases. There are no low cost modules with high reliability and bandwidth and long range. That is why military use Beyond Line-of-Sight (BLOS) communications, like satellite communication, that allows long ranges, and civilians use RC in Line-of-Sight (LOS) limiting the range of the operation [4][5]. The military uses Ka band (uplink: 27 – 31GHz; downlink: 17.7 to 21.2GHz; bandwidth: 3500MHz) and civilians the Ku band (uplink: 13.2 to 18.1 GHz; downlink: 10.7 to 12.7GHz; bandwidth: 500MHz) [6][7][8]. In way of avoid the satellite communication complexity and the short range of radio control, cellular networks will be used in UGVs control context: Universal Mobile Telecommunication System (UMTS), Evolved High-speed Packet Access (HSPA+), Long Term Evolution (LTE) and Long Term Evolution Advanced (LTE-A). The

connection between systems will not be affected since high transmission rates and low latencies (Round Trip Time (RTT)) are allowed:

Table 1.1 - Transmission rates, network latencies and bandwidths. *Source: [9]*.

Feature\Technology	**UMTS**	**HSPA**	**HSPA+**	**LTE**	**LTE-A**
Downlink [Mbit/s]	384Kbit/s	14	28	100	1000
Uplink [Mbit/s]	128Kbit/s	5.7	11	50	250
RTT [ms]	~150	< 100	< 50	~10	~ 5
Bandwidth [MHz]	5	3	5	15	20

Thus, real-time video transmission and control of a BLOS UGV can be done with quality and without the referred problems of the other communication types;

iii. Control system: this is closely related with the communication, using it to control the vehicle, sending and receive commands and video. Usually it is a remote control but it can also be a tablet or a similar device. In remote control, an additional display is needed to show the video and monitored parameters; the tablet (or similar) can do it all together.

1.2 Motivation

The concepts of technology and automobile industry were not always consensual. As two distinct concepts that they were, their merger was something hard to imagine and therefore to accomplish. Over the time, both began to integrate and today their separation is unthinkable, even prohibitive due to their correlation.

Any car manufactured in the last 15 years has incorporated a lot of technology in its operating system. The trend is that the more recent the vehicle is the greater the amount of technology embedded in its system, as well as greater diversity and quality. It is quite important to note that this integration process was possible due to a large investment in both areas, not only the economic but also the employment level.

As such, eventually appear Unmanned Ground Vehicles (UGV), one of the subgroups of the Unmanned Vehicles Systems (UVS). Another well-known example are the Unmanned Aerial Vehicles (UAV) or as they are called among the civilian community “drones”.

The largest investor and driver of UGVs was the military branch. They gather a set of favorable conditions for the development of Unmanned Systems Technology (UST): investment funds by the defense departments of governments, restricted and controlled space (aerial, land and

naval) for testing, research teams, specialized staff and workshops for vehicles construction. UST are evolved in such a way that many military institutions have specific for careers for UGV and Unmanned Sea Vehicles (USV) drivers and UAV pilot [11]. Here we can see the impact that this technology had in this business.

Speaking concretely of UGVs, there are many possible applications for military operations, such as: explosives disarming and threats analysis, surveillance and pursuit operations, access to dangerous zones, discovery of landmines, seek for radiation, search and rescue, space expeditions, *etc*. Besides de military field, this technology may also be employed in civil and employment context: agriculture (analysis and treatment of crops), civil engineering inspection, fire combat, hobby, photography and video capture, timber industry (cutting and transport), mining operations, *etc*. Of course that many of these applications can be employed in military branch.

In UGVs control and monitoring the transmission of telemetry and real-time video is mandatory. As the video will be the eyes of the driver, this has to have a minimum acceptable quality. The UGVs used for military operations are controlled via satellite communication because access to this type of communication is very limited, which leads to greater system security. But satellite communication has an associated problem that can, depending on application type, derail the system functioning: significant delay times. The UGV systems for civil applications doesn't use satellite communication because it's not feasible in this context, both in monetary terms and in ease of access. Instead, Radio Control (RC) is used, and the vehicle has to be in LOS, limiting its range. For real-time driving, both these communication schemes are unviable. A possible solution that can remove the disadvantages of each of these communication types is the use of 3G and 4G mobile networks. These technologies allow multiple operators (*e.g.* 1 pilot and 1 camera operator), multi-vehicle control and control of vehicles regardless of the location of vehicles and operators. The problems of long delays in satellite communications detrimental to driving real-time applications and the limited scope of the RC communications are exceeded. However, mobile communications have also some associated problems due to:

- Mobile networks: maximum bit-rate, loss of coverage, handovers;
- Packet network usage: latency.

The notable developments in the market of mobile devices (smartphones and tablets) and its applications has brought a new paradigm for applications that were only developed for

computers, where the user had limited mobility as it needed to have a fixed location with access to the Internet. Mobile devices provides all the necessary components and features for the system architecture: mobility, GPS, Bluetooth module, wireless communication modules (HSPA+, LTE, Wi-Fi), processing capacity, receiving and sending video, friendly User Interface (UI), sensors data, among others. Thus no need to have a processing unit coupled with modules apart. This way, the system will be transparent regarding the location of the vehicle and driver.

The alliance between the automobile industries and mobile networks increasingly efficient, brings a new paradigm: remote driving with easy access. Nowadays the youth community has easy access to driving simulation applications and to racing games through their smartphones or tablets. This fact allows these young people to react in a natural way when faced with real driving situations. An excellent example of a similar situation is the aircraft pilots' case: they have to meet a certain number of flight simulation hours before moving to a real situation.

1.3 State of art

The market of UGVs is rising every day and it brings not only new vehicles but also new software/hardware solutions. For military/security purpose these solutions are classified and little is get to know about them. In the case of civilian use new hardware and software solutions for UGVs emerge at high rate, as proprietary projects or OSPs. These projects were developed by companies or academic groups and includes the whole UGS: vehicle, communication, control and monitoring modules.

1.3.1 Controller boards

Many companies have chosen to develop their own UGVs, making them proprietary UGVs. Autonomous Solutions Inc. is a company that develops ground robotic solutions, i.e., UGV command and control software and technology for farming, mining, governmental and automotive areas worldwide [12]. ASI offers a multi-vehicle and user friendly software for unmanned vehicles command and control – the Mobius [13]. The Nav™ Automation Kit was also developed by ASI and allows a manual control vehicle to be controlled robotically [14]. Its area of action are remote control, teleoperation and full automation.

Mechatroniq Systems Inc. is an American company that designs and produces UVS and all related technologies [15]. The MissionPlanner is a software developed for planning paths and

control and monitoring remotely the Mechatroniq's UGVs [16]. The MEC is a small controller board built around and ARM Cortex A8 processor that provides remote sensing applications and autopilot the vehicle (and others) [17]. Minu is a board with GPS with active antenna, temperature sensor and advanced sensing technology with 9 degrees of freedom and is applied on robotics, vehicle orientation sensing and position, asset tracking, dead reckoning and precision of autonomous vehicle guidance [18].

Sterela developed the 4MOB for applications like monitoring, inspection, external operations and agriculture tasks/construction [19]. It is a UGV with already considerable scale, speed and payload capacity [19]. For control, a radio control is used but route planning is also possible. This vehicle provides video surveillance and teleoperation [19].

Parrot has created the Jumping Sumo, a mini UGV controlled by a mobile application via Wi-Fi connection [20]. This vehicle is targeted to hobbyists, however it is possible to ingrate in other types of applications.

There are many more, those 4 are just an example. Proprietary systems have the advantages of ensuring technical assistance but they are quite directive, not offering flexibility for other kind of applications than that for which they were designed. In turn, the OSPs are very flexible (*i.e.* allow adaptation to the desired application) and have a big worldwide community that provides hardware configurations, software, tutorials, tests results, *etc*. A great example of that is the DIY Drones community [21] that support all kinds of unmanned systems.

The objective of the controller board is, as the name implies, to allow control of the vehicle whether it is manual control, semi-automatic or fully autonomous. However, with the exception of the manual mode, the incorporation of a GCS is required so that all information be centralized. In these cases the controller boards have a GCS, where is possible to send and receive commands and receive and request telemetry information.

It is safe to assume that Ardupilot is the most well-known and used in the world of OSPs [22]. Its versatility is the main reason why it is one of the most used platforms for UVS control. Based on Arduino it is also easy to handle and has a friendly IDE to program [23]. It Is available for download software for desktops that act as a GCS: Mission Planner and APM Planner. Both are able to control and monitor vehicles through Xbee modules. In addition to the described software, there are still GCS applications for Android OS: Andropilot and DroidPlanner [24][25].

Pixhawk was projected to integrate in an easy way with better safety features and to improve reliability compared to present solutions [26]. This platform uses computer-vision algorithms to implement control of the vehicle through on-board image processing [27]. It provides a GCS called Qgroundcontrol that allows vehicle control, parameters setting and missions planning. At the present time, Pixhawk is known as the improvement of APM 2.6 (the most developed version of APM so far).

Openpilot was developed by hobbyists and uses a modified version of FreeRTOS operating system. In addition to providing a GCS called Openpilot GCS, this platform is constituted by CC3D and CopterControl boards [28].

Paparazzi is an autopilot platform that support all configurations of Micro Aerial Vehicles (MAV). Designed to be a low cost solution this platform provides the Paparazzi GCS for monitoring and control (allows pre-programmed missions and others flight modes) through radio systems [29].

Mikrokopter is and autopilot system developed for multi-rotor configurations: quadcopter, hexacopter and octocopter [30]. Provides a GCS, the KopterTool, allowing pre-programmed mission and telemetry transfer. Like Ardupilot, Mikrokopter also have an Android application: BUDwise.

KKMulticopter is aimed at the hobbyists who do capture of aerial images using quadrotors [31]. A very basic system developed for leisure activities. For this reason, it has no GCS, but is provided a tool for firmware transmission to the board [32].

MultiWii was developed by RC hobbyists and uses Arduino framework and Nintendo Wii sensors [33]. It was designed for multi-rotor configurations. It has a GCS, the MultiWii WinGUI and an Android application, the MultiWii EZ-GUI that only allows the change of board parameters [31].

Aeroquad is similar to MultiWii and is also based on Arduino. It was designed for multi-rotor configurations and provides a GCS only for parameters modification [34].

It is important to refer once more that it is possible to shape the board to the desired application. In some definitions of some boards come references to multi-rotor vehicles, hexacopters, *etc*, since this was its original purpose. However, since they are OSPs, it is possible to develop firmware for the desired application and execute it on the board.

Speaking specifically on projects that make use of cellular networks to control and monitoring UGVs, there are some studies prepared, as mentioned in [35] and [36] but none is a commercialized product. Apart from this type of study, there are no known marketable and/or implemented projects that perform control functions and monitoring of a UGV using the cellular network.

1.3.2 Covered areas

1.3.2.1. Control

The control of a UGV can be done via an application software or a remote radio controller. Each one of them can be designed to use a specific type or communication technology and operate in a certain network topology. The most common applications at civilian and OSPs level uses a RC controller and most recently mobile applications via wireless communications (Wi-Fi and mobile). In military case, the control is made via satellite communications and specific software and hardware developed by them self or private companies. For example, Oshkosh Defense Corporation develop the Command Zone™ [37] technology to operate remotely (and also monitoring) a ground vehicle using laptops, on-board displays or digital devices. It is not possible to access more detailed data about military projects, due to its important value in defense.

1.3.2.2. Monitoring

Monitoring must indeed be made using a display where the information is presented. What is unusual is the change in the type of display depending on the technology used, *e.g.* in mobile applications makes sense to make the mobile device display, in cases of use an RC remote usual use a laptop computer or other type of monitor. At military level the monitored parameters are extensive and some with some adjacent complexity, *e.g.* Oshkosh TerraMax is a technology equipped with ground-penetrating radar and mine roller systems [38], and these are complex technologies. In civil use the complexity of the monitoring system will depend on the type of applications: leisure will be a simple system but for industrial it will be a complex system.

1.3.2.3. Mobile applications

In terms of mobile applications for ground vehicles, nothing that offers simultaneously real-time control and monitoring via mobile networks. In military sector there is no knowledge of any type of mobile application directed towards the UGVs.

In the civil branch already begun to appear aimed applications for the control of UGVs, *e.g.* the FreeFlight 3 for Jumping Sumo by Parrot [39]. This application uses the private Wi-Fi network of the vehicle. There are some applications but only for monitoring and control a set of actuators (driving it is not possible). These enter the automobile context and the following are the ones most outstanding:

- "Audi Mileage Tracker" by Audi [40];
- "BMW i remote" by BMW [41];
- "OnStar RemoteLink" by General Motors [42];
- "Chevrolet Volt" by Chevrolet [42];
- "Nissan Leaf app" by Nissan [42];
- "Hyundai Blue Link" by Hyundai [42].

1.3.2.4. Communication

Wireless radio communication for UGVs typically utilized three distinct radios in way to provide the capabilities described in **Table 1.2**. The commands are sent to UGV using wireless technologies, such as Internet or ZigBee [43].

Table 1.2 - Communications capability needs for UGVs. *Source:[44]*.

Radio Capability	**Description**
Data Transmission	Transmits control signals from operator to UGV
Video Transmission	Transmits analog video from UGV to operator
Emergency Stop	Transmits signal to disable UGV

Usually, radio technologies are designed to have a balance between its 3 most relevant requirements: bandwidth, latency and signal propagation [44]. If it was possible to use only one channel, capable of encoding the video and combine voice with data the system, it would be much less complex, reducing size, weight and power consumption.

The transfer of telemetry and video is essential for UGV control and monitoring. **Table 1.3** presents the imperatives for network data traffic in UGVs operation. Many types of

communication and network topologies can be thought for that purpose. In the remaining sections will be presented some points of view within that context.

Table 1.3 – Quality of service requirements in UGV communication. *Source:[45]*.

Data Traffic	Delay Tolerance	Jitter Tolerance	Bandwidth Requirement
Real-Time Sensing	Low	Medium	Low
Store-and-forward sensing (offline storage)	High	High	Low
Command and control	Low	Medium	Low
Real-Time video	Low	Low	Very High
Store-and-forward pictures	High	High	High
Store-and-forward video	High	High	Very High

1.3.3 Application areas

The UGSs technology was and is the target of a major investment [1][2], not only for military but also for civilian use. These systems have several of applications that may not be as intuitive at first sight. UGVs can replace humans and go to places where they can't [46]. For example, UGVs can be used as:

- Monitor and manager for crop conditions during the growing season [47][48];
- Wild fire surveillance, providing sensor data for fire detection and the ability to notify the fire forces in case of fire [49];
- Scientific research of any kind of nature (atmospheric, pollution, environmental, *etc*) [49];
- Boarder interdiction, patrolling the borders [49];
- Law enforcement, patrolling dangerous areas for manned operations [49];
- Disaster and emergency manager, providing real-time surveillance in hazardous environments [49], *e.g.* in coal mines, to avoid disasters and consequently high risk rescue operations (usually the used communication technologies are ZigBee along with RC due to its efficiency and low cost) [50][51];
- Industrial applications, such as surveillance of pipelines or nuclear factory [49][52];
- Detector of chemical agents, explosive, volatile gases and radiation [53];
- Explosives disposal [54].

1.4 Objectives

The aim of this dissertation is to study the implementation of system capable of control and monitor, locally and remotely, a UGV using wireless networks: Wi-Fi and 3G and 4G mobile networks, more specifically UMTS, HSPA+, LTE and LTE-A. In this context the following goals were defined:

- Development of a system for control and monitoring multiple UGVs through an open-source flight controller and Android mobile application;
- Structure a generic and modular system but robust enough, to allow the insertion of new features since this area is in exponential growth, which leads to constant changes;
- Overcome the delay problems related to wireless communications in real-time applications, in this case operate a UGV;
- Development of an Android application that will act as a GCS, centralizing all the telemetry and receive video stream;
- Connection establishment between the GCS app and the vehicle;
- Evaluate the designed system's efficiency through test fields and analysis of collected data.

1.5 Contributions

As explained before in section 1.4 this thesis proposes a low complexity system capable of control and monitoring multiple vehicles (greater focus on UGVs). Designed to be a very complete platform, it is also user-friendly and easy to implement. The system includes the vehicle construction and the development of several applications:

- Remote GCS: Android application for UGV control and monitoring;
- Routing Server: java program that will allow the communication between Remote GCS and the vehicle;
- RPI Camera: shell script that provides the camera to stream video from RPI to Remote GCS.

Note that the RPI is also running other application besides RPI Camera: MAVProxy. This software was developed by Andrew Tridgell and it is open-source. This way the RPI can be seen as a relay server between the vehicle (APM more precisely) and the Remote GCS. The use of this open-source software is connected with the fact that the APM uses Micro Air Vehicle Link (MAVLink) protocol to communicate. As result, a study of MAVLink and UGVs was done.

It is worth mentioning that implementing the waypoint protocol and parameters protocol, both constituent parts of MAVLink, was not an objective. However, they were implemented, making it a more complete project.

As a scientific contribution, a set of 2 articles were elaborated, with special focus on chapters 3 through 8. Once completed, 1 was submitted to a conference (Conftele 2015, *published*) and the other will also be submitted to a conference or international scientific magazine or journal.

1.6 Dissertation structure

In this section the dissertation structure is presented. The dissertation is composed of 8 chapters:

- Chapter 1 – Introduction: current chapter;
- Chapter 2 – Unmanned Ground Systems: presents the main modules of an UGS;
- Chapter 3 – Dimensioning an Unmanned Ground Vehicle, comprises the explanation of the assembly of the vehicle used in this project, *i.e.* its constituent modules and the way they operate.
- Chapter 4 – Boards: explains the boards that are used on the system, its operation mode and characteristics;
- Chapter 5 – Communications: covers all the communications types used, brief explanations and protocols;
- Chapter 6 – Control and Monitoring Applications: presents and explains all applications used to operate the vehicle, mainly the Remote GCS;
- Chapter 7 – Results and Evaluation, shows the results obtained through field tests, treated so that its analysis is easy and as intuitive as possible;
- Chapter 8 – Conclusions and Road Ahead: presents the conclusions to be drawn from the system performance, through all the results and in what way they will matter towards the future.

Besides that, there are 6 annexes at the end of the document that contains a more detailed explanation about subjects that comes up across the document: **Annex A** for raspberry versions and specifications; **Annex B** that contains the structure of a MAVLink message; **Annex C** for describe MAVLink internal protocols; **Annex D** for an explanation about the control mechanisms; **Annex E** that contains a message diagram for a communication example; **Annex F** with field-tests details.

Chapter 2

UNMANNED GROUND SYSTEMS

Here will be specified the main blocks of an Unmanned Ground System and thereafter described point by point. All possible technologies for each block are referred as well as the existing blocks inter-relation in the system.

2.1 Introduction

An UGV is set by several blocks that work together for proper operation of the vehicle. The malfunction of one of these modules has negative implications on the system and may even lead to irreparable damage in case of accident. Each block has its task and has to fulfill it correctly. Therefore, it is not possible to assign valuable degrees to each, because a malfunction of a block has direct implications on the system operation. Almost 100% of the time all modules are working together, waiting for each other's results in order to make decisions or complete its cycle. These blocks are properly identified on **Figure 2.1**.

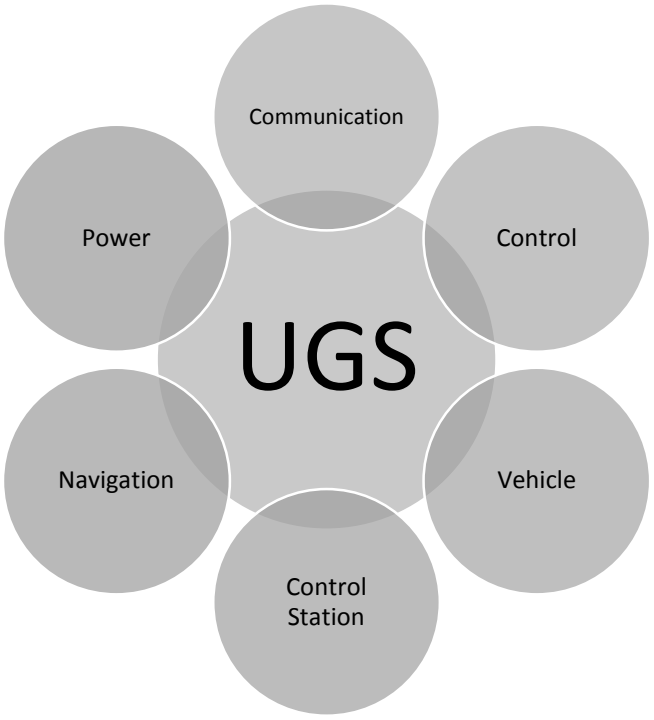


Figure 2.1 – Main blocks of a UGS composition. *Adapted from: [4]*.

2.2 Ground Control Station

The GCS is the physical module through which the user communicates to the vehicle, being able to edit parameters, see telemetry information and send commands [44]. In other words, it can be seen as the cockpit of the vehicle where it is possible for the user to perform all actions as if he was present inside the vehicle [55], the human-robot interaction. GCS can have many forms, sizes and shapes and consists on a sensor station and a pilot (at least) [4]. Normally it is running on a computer (or more) or other(s) device(s) with processing capacity, allowing interaction between the operator(s) and on-board systems [55]. The pilot gives commands and the servo system of the vehicle executes the corresponding actions, this way the vehicle avoids

algorithms and shows some intellect [56]. Therefore, the communication is indispensable for GCS functioning. The GCS allows the operator to determine the future positions of the vehicle – real-time control and mission control. This last element will cause a reduction on reliance of satellite communication for BLOS operation [56].

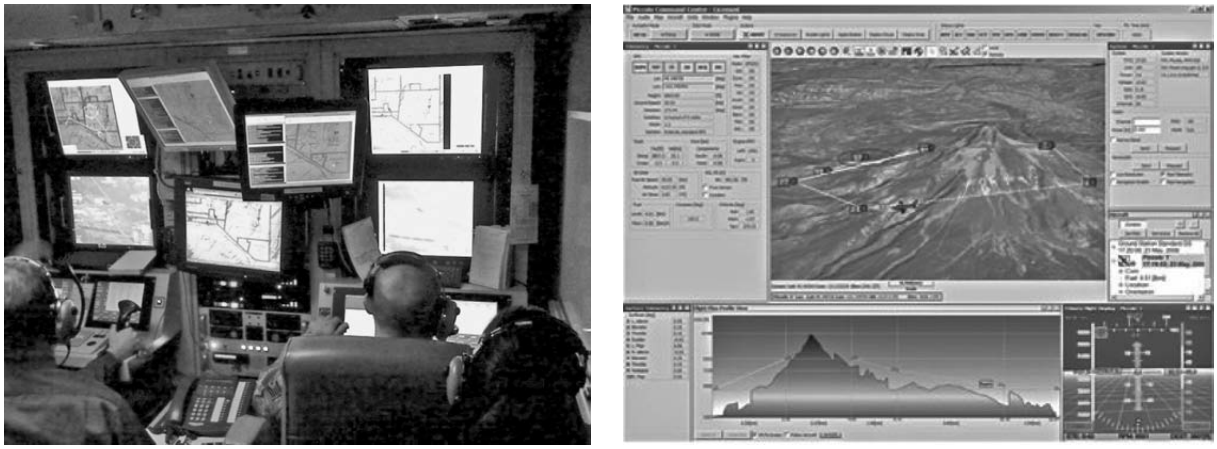


Figure 2.2 – Example of GCSs. *Source: [4]*.

2.3 Communication

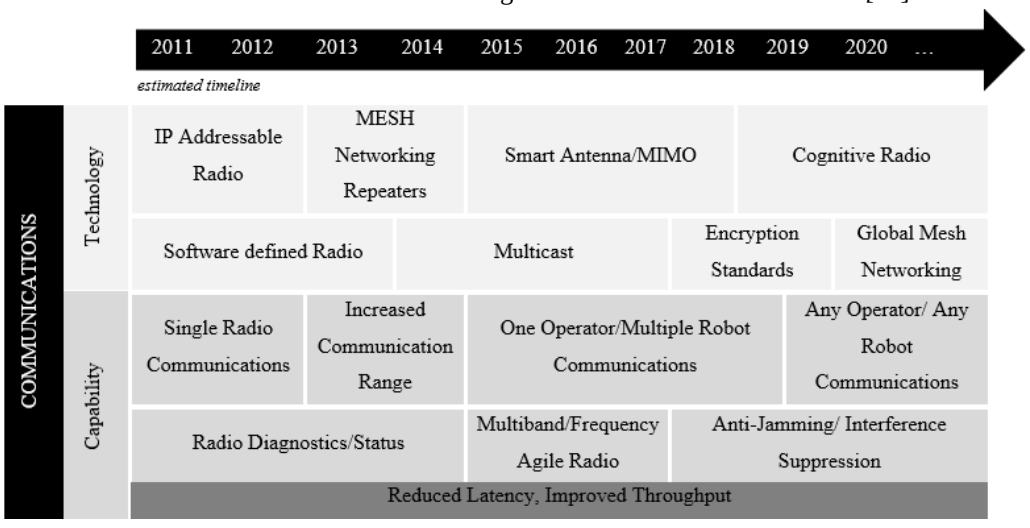
The wireless communication is a mandatory factor to operate an Unmanned Vehicle (UV). This is one of the reasons why improve these systems is a hard task. **Table 1.2** on **Chapter 1** shows that different radio links are used, fact that adds complexity to the system, augmenting the minimum requirements. In order to simplify the communication a single radio channel can be used to transmit video (coded through digitizing) and data and voice combination [44]. The GCS transmits commands, vehicle video and data in a closed loop link (the information is not shared with other networks). The communication link is of extreme importance since the operator must maintain the control of the vehicle all the time. A communication breakdown can lead to dangerous situations, such as the vehicle destruction (and others, depending on the type of application). The implementation of sail safes can be a way to remedy the problem, like return to the start point or home in case of communication loss. However, all current unmanned vehicles are on par with the same communication problems: low availability of transmission channels and the amount of data that they could handle, resilience of Radio Frequency (RF) subsystems against interference and spectrum allocations. [57].

Investments made in UGVs and the increased use mean that its communication system must be developed to support the new capabilities added to new vehicles. With the communication systems evolution, it is expected that they will be more efficient, simple to use, smaller and

lighter. Usually, UGVs use RF communication, being highly integrated into their architecture. That RF communication is usually limited to unlicensed frequency bands, *e.g.* 2.4GHz that are very susceptible to interference with other wireless communications operating at the same frequency band [44]. To avoid interference problems, higher frequency bands can be used, which supports higher data rates and wider bandwidths for real-time operation and video streaming with minimum acceptable quality. However, lower frequencies propagate better than higher frequencies, limiting the vehicle range operation, both in LOS and BLOS. The wireless communications, namely cellular, suffered from big improvements from the last years. Larger bandwidths, higher transmission rates, lower delays and higher coverage are the main improvements of these networks. The integration with tablets, smartphones and other computing platforms and the fact that the infrastructures are already implemented (no need of investment) made this market even more appealing to the world of unmanned systems. It is an inevitable fact that one the operation of unmanned vehicles will be performed through cellular networks. **Table 2.1** represents the communication evolution from 2011 to 2020, both technology and capability improvements.

Teleoperated vehicles require very high transmission rates, low latencies and high reliability communication systems. The 4G of cellular networks, namely the LTE–A, is capable to provide those features when in good/medium conditions (see **Table 1.1**).

Table 2.1 – UGVs communications goals from 2011 to 2020. *Source: [44]*.



2.4 Power

The power subsystem is responsible for supporting the energy requirements of the entire system. The discharge rate is a key element for choosing the right power system. Depending on the vehicle application the power system will vary. The sources go from fuel cell, solid oxide

fuel cell, batteries subsystems through hybrid solutions. Usually, fuel solutions are used on heavy vehicles and cargo vehicles where there needs to be a great responsiveness to the total weight. To this end high torque engines powered by liquid or solid fuel are usually used. However, if fossil fuels are used (traditionally gasoline and diesel), the issue of environmental pollution enters the picture. Hybrid solutions can take an important role in the transition from fossil fuels to system powered by batteries and other renewable energy sources (*e.g.* solar, wind, *etc*). But both solutions have the same challenges: performance is affected by the weight and bulk of expendable fuel. At present, small unmanned vehicles use almost exclusively batteries, namely lithium batteries, since they are rechargeable an environmental-friendly. Small UGVs do not require the torque specifications seen in the heavy ones. Those are the typical combinations, however heavy vehicles can use batteries and small vehicles can use fuel. The selection of the appropriate power source must take into account the UGV application with the features of using each type of power supply (*i.e.* noise of the engine work, maximum duration of the source power, *etc*). For example, if a UGV is used for surveillance or spy, this should be small, silent and have a medium/high energy autonomy – use batteries; however, if the terrain is difficult or the distance to cover is high the batteries may not be enough and fuel subsystems can take the place.

The current unmanned vehicles have minimum requirements related with power suppliers in order to be efficiently operated: they must be reliable, energy dense and rechargeable. Besides that, these power supplies must satisfy a set of safety and environmental rules.

The battery market is in constant evolution and nowadays good performances can be achieve. In fact, at least 90% of civilian branch will use some kind of battery to power his vehicle [58]. There are many kinds of batteries and, again, its usage will depend on the type of application. Lithium batteries dominates this area and are constant present on small UGVs [44], replacing the Nickle Cadmium (NiCad), Lead Acid and Nickle Metal Hybrid (NiMH). They are very popular due to its real high energy density and have a power capacity which allows the vehicle to operate for an acceptable period of time. However, they have to follow a comprehensive and rigorous range of safety and environmental protection rules [44].

The technological advances in power systems, UGS and automotive industry leads to a new kind of vehicles that have begun to be something usual: electric and hybrid vehicles. Due to its batteries and super capacitors they have enough energy to move great distances.

2.5 Navigation

The navigation system of a UGV includes a set of modules that helps on vehicle navigation. Usually, UGV uses a combination of GPS and INS [59]. The INS is composed accelerometers that detects motion and gyroscopes that detects rotations. Through estimations of speed and time intervals between previous locations, the vehicle position can be calculated. This process is known as “dead reckoning”. However, INS has some tendency to an error known as "drift". In order to correct this error a GPS is used and so the INS/GPS combination allows to determine the vehicle current position and orientation. Global Navigation Satellite System (GNSS). Although, GPS receivers can be vulnerable to jamming, as GPS signals can be target of “spoofing” (type of hacking attack), that disrupts the vehicle navigation by sending false signals. That is why the security, availability, robustness and capacity of command links are of utmost importance.

Currently, GPS is the most used GNSS, but there are others. Galileo is the European global satellite-based navigation system and Glonass the Russian GNSS [60]. These 3 GNSS are nevertheless 100% interoperable with each other, giving not only a bigger coverage of available satellites but also a better performance than in the case where only one of the technologies used or a combination of two of them [61][62]. Crossing the triple signal frequencies from each technology with each other will improve the accuracy of the navigation system [63][64].

For teleoperated vehicles the GPS/INS combination is not enough. In addition those vehicles must have a video stream through each the operator will control it. The video will be his eyes, and it is sent through a processing unit. In military branch the radar is also a possibility, making possible to see further and avoid obstacles in motion or stopped. For this, it is most valuable.

2.6 Vehicle

The UGVs don’t have specified configurations or standards. The type of application will dictate the vehicle size, shape, weight and configuration. Main block of an UGV are identified on **Figure 2.3**. Usually an UGV has 4 wheels and can be Two-Wheel Drive (2WD) or Four-Wheel Drive (4WD) (see **Figure 2.4**). There are others who do not have wheels and their locomotion is made using the same mechanism used in tanks and industrial machinery – continuous track, also known as tank tread or caterpillar track (see **Figure 2.5**). There are even cases where legs are used. The engine is responsible for transforming the energy received from the energy module in rotation to the wheels.

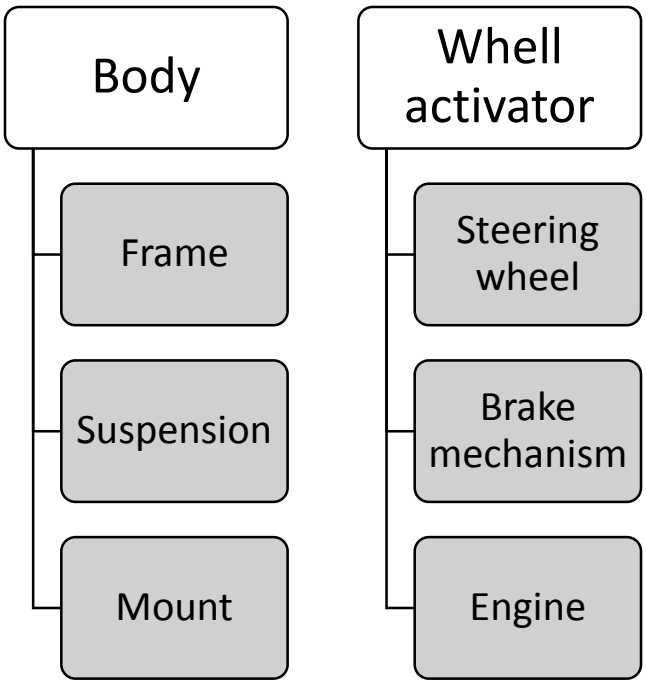


Figure 2.3 – Core modules of an UGV. *Adapted from: [65]*.

Figure 2.4 – Example of a 4WD UGV. *Source: [5]*.

Figure 2.5 – Example of a continuous track UGV. *Source: [5]*.

2.7 Control

UGVs control is based on a board that processes all received commands and sends orders to the respective actuators (control). Usually this board is a unique piece but there is the possibility to have two boards, each one dedicated to a function: one for control and other for processing. The commands that this module receives are usually sent by a traditional RC that operates on commercial band or military band (Ka and Ku bands) or microcomputer user interfaces (*e.g.* keyboards, mice, joysticks and touchscreens). However, the increasing use of mobile devices in the last years (see **Figure 2.6**) as lead to a new paradigm. Mobile devices are becoming more constant in human life and as such, are taking the place of the traditional RC commands.

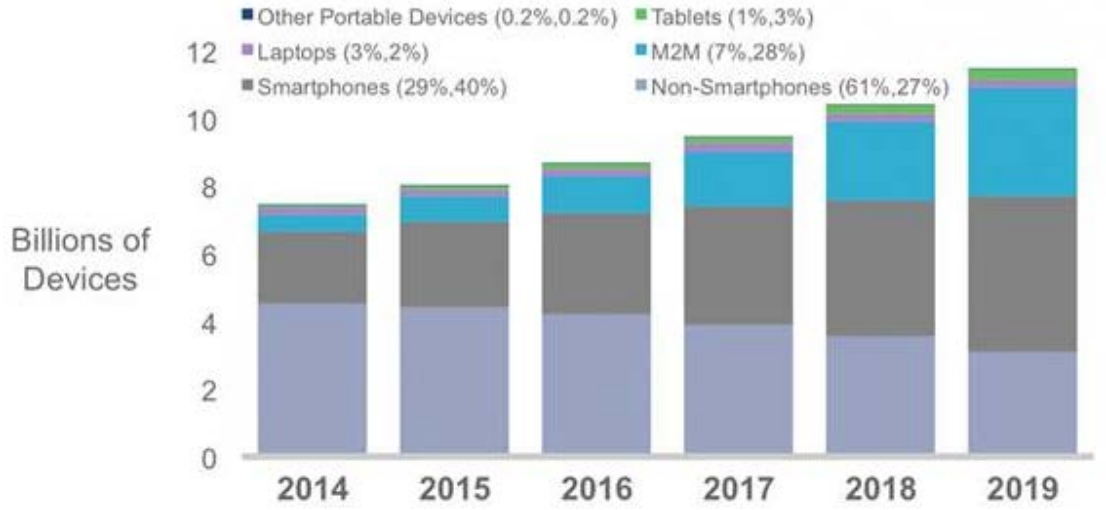


Figure 2.6 – Global mobile devices growth. *Source: [65]*.

The UGV's autonomy is related with the degree that operator has on vehicle operation. It can be quite diverse, as the vehicle can be controlled manually (no autonomy) or be completely autonomous and intelligent. There are several standards that define the levels of autonomy of an UGV. This level will depend on task difficulty, environmental complexity or required operational time [62].

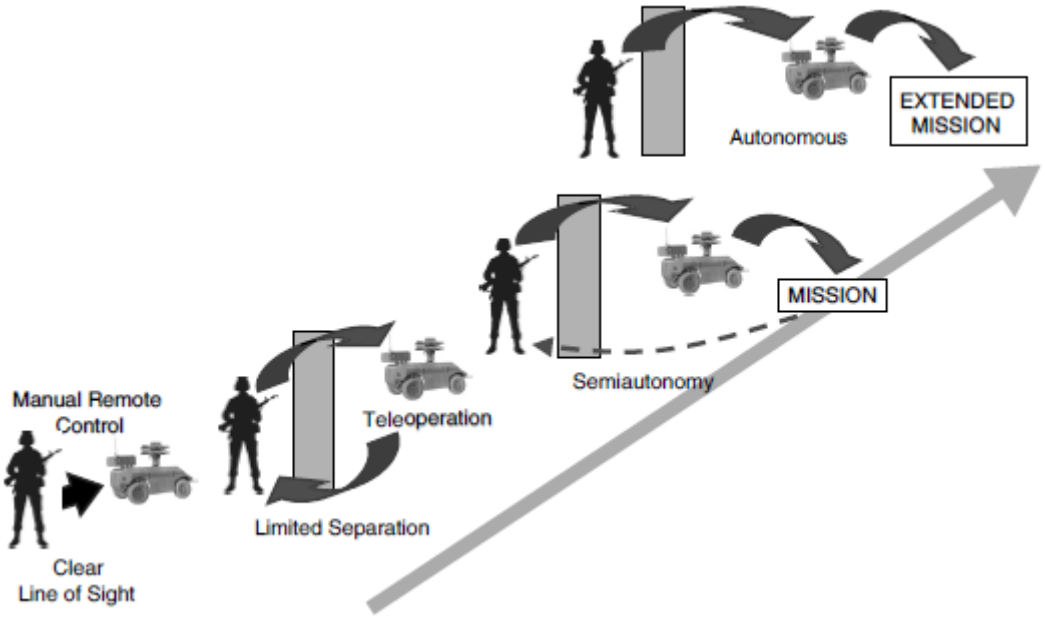


Figure 2.7 – Evolution of UGV control system. *Source: [62]*.

As shown in **Figure 2.7**, the trend follows towards vehicles completely autonomous, but this does not mean that they are best suited for all types of applications. Each type of application may require certain features or functionality that a particular navigation mode offers exclusively or in a better condition.

Manual remote control in LOS requires a radio technology for communication. This mode provides direct control of throttle and steering. In teleoperation it is also possible and the user operates the vehicle BLOS using a video stream in real-time (the video will be the eyes of the operator), from which he gives feedback. The semiautonomous mode consists on establishing a pre-programmed mission and the vehicle executes it when ordered to. The vehicle uses its sensor system to guide itself. Full autonomous UGVs perform tasks without the guidance of an operator, reducing his work and increasing the performance in bad or limited communication conditions. In full autonomous mode the vehicle is capable of adapting itself to the surrounding conditions and make decisions without the need of an operator using all its systems.

This page intentionally left in blank

Chapter 3
DIMENSIONING AN UNMANNED GROUND VEHICLE

Dimensioning an UGV is not as simple as it sounds. This chapter describes the main challenges of this process and matches the modules described in Chapter 2 to the material used in the development of the system.

3.1 Introduction

The process of dimensioning an UGS is a complex process, contrary to what one might think. The reason why is due to an exponential increase of unmanned systems and its applications in real-world situations. It has become common to find these systems performing actions that in the past were done by humans or manned systems. This increase leads to a higher level of complexity when talking about dimensioning an UGV.

The major challenges that this process faces are [67]:

- Interoperability: every component has its own features and characteristics as well as associated problems. Thus all system blocks must operate seamlessly with each other and the system itself with other manned and unmanned systems;
- Communications: current UGS demand a high degree of human interaction. As result, those communication links must have protection since the information that flows on it is critical to vehicles operation. Therefore, the reliability of those links are also a very important factor. In case of miscommunication the system has to be prepared to react and activate fail safe mechanisms. However, the fewer failures there are, the more reliable will the system be. This has to contemplate mechanisms that provide reliability to the system or at least improve it in some way (*e.g.* communication protocols, redundancy mechanisms). In general, the challenges at this level are security, spectrum availability, bandwidth and link range;
- Training: the operation of an UGV could be a complex process. In order to improve this process training could be considered as a requirement before operating an UGV in real-world applications which more responsive actions (*e.g.* in the case of aeronautical piles, they have to perform a certain number of flight simulation hours before flying a real aircraft);
- Autonomy: depending on the type of application, the autonomy level will vary. The higher the autonomy level, the greater will be its complexity (for development and implementation). The implementation of autonomy in vehicles will reduce the necessary bandwidth.

3.2 Vehicle main blocks

The vehicle is an integrated part of the developed system. **Figure 3.1***Figure 3.1 – UGV composition and all on-board components.* illustrates the vehicle composition used in this project as well as all

on-board components that are also part of the whole system. Each component is described in more detail on the following sections of the current chapter.

Figure 3.1 – UGV composition and all on-board components.

3.2.1 Frame

A four wheel configuration was chosen whose drive system uses these 4 wheels for locomotion, *i.e.* the vehicle is 4WD. The frame has a suspension system which, together with the 4WD feature with front/mid/rear differential, allows the vehicle to be operated in difficult terrain. It is possible to adjust Toe, Camber and shock position. Camber angle is the angle made by the wheel when viewed from the rear or front of the car [68]. The camber can be positive, neutral or negative as shown in **Figure 3.2**. Toe is related with the direction the wheels are pointing

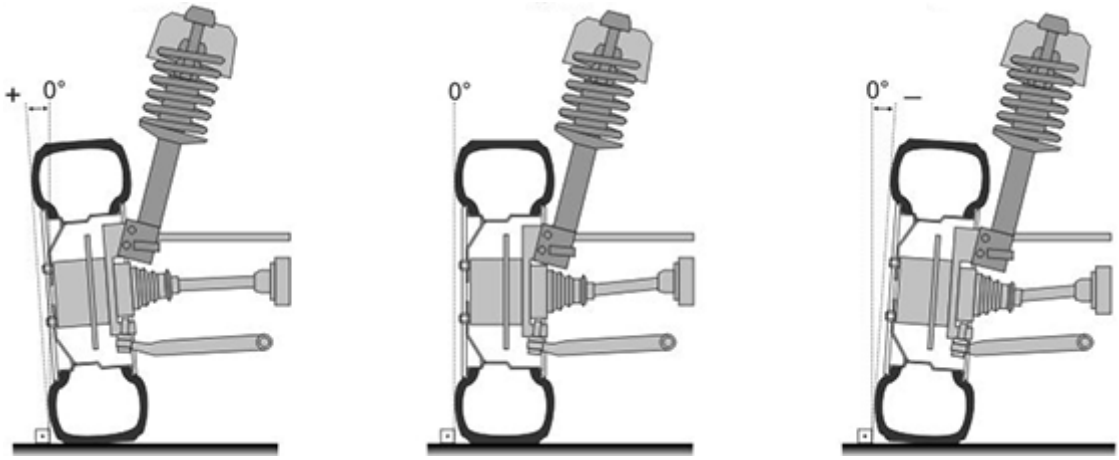


Figure 3.2 – Camber angles. *Source: [68]*.

when viewed from above [69] (see **Figure 3.3**). The tie rod and camber are made of steel, which increases the reliability; the big bore shock is made of aluminum and has 17mm. The frame itself is made by a combination of plastic and aluminum. This part has, by default, own locations for engaging the other vehicle components: motor, batteries, Electronic Speed Controller (ESC) and RC receiver (default communications system). With the cover on, its dimensions are: 305mm width, 200mm height and 530mm length.

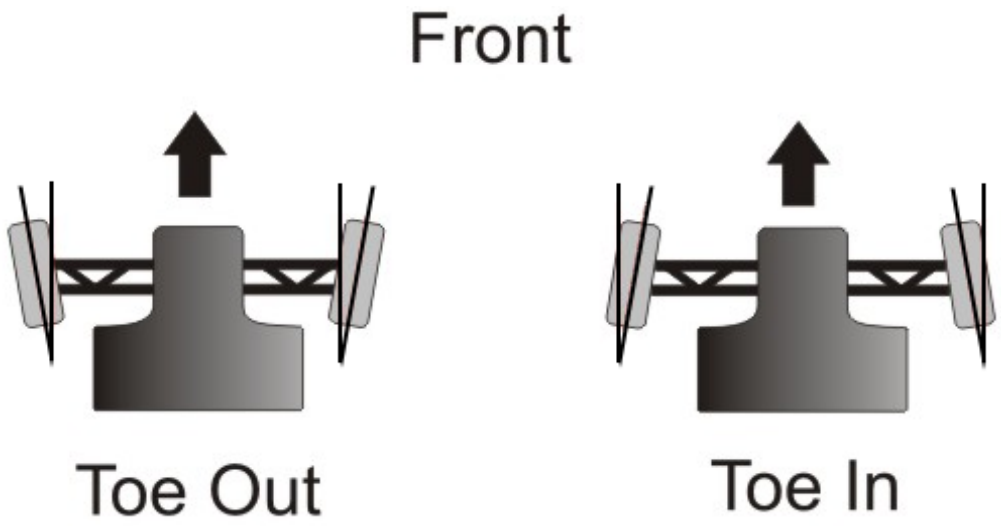


Figure 3.3 – Toe angles. *Source: [70]*.

3.2.2 Motor

When electric schemes are used, it is customary to use one of two types of electric motors: brushed or brushless. In the present case, a sensorless brushless motor was used. To understand how brushless motors work it is important and easier to first perceive how brushed motors (often referred as canned motors) operate. Every brushed motor consists of the following parts [71]:

- Armature: set of the poles, terminals and the commutator. It is basically the rotating portion of the motor.
- Poles: electromagnets formed by cooper wires wrapped around pieces of metal and attached to the armature. Almost all motors have 3 poles to prevent the battery from shorting out (lowering efficiency) and to prevent the motor from getting stuck;
- Commutator: copper section that acts like a switch on the armature, reversing the current to the poles every half rotation. Thus, rotation is maintained by the magnetic fields of each of the poles;
- Brushes: device that is wired to the battery and promote the transfer of electrical power to the armature by touching the commutador;

- Magnets: two permanent magnets with opposite polarity are attached inside the motors case (or ‘can’);
- Armature: set composed by the spinning shaft wrapped by wires, the poles and the commutator.

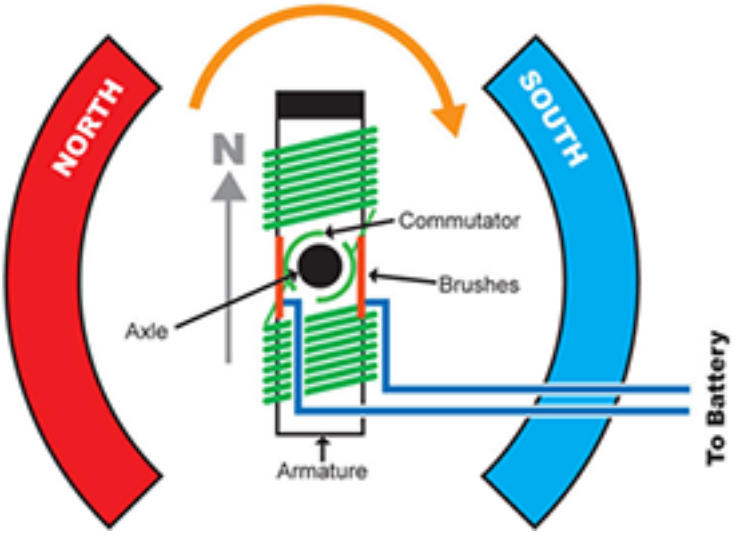


Figure 3.4 – Operation of a traditional brushed RC motor. *Source: [72]*.

In **Figure 3.4** it is possible to identify each of the constitution part of a canned motor and also helps to understant its operation mode. The battery is directly wired to the brushes that make contact with commutator plates as the armature spins, charging a particular pole (electromagnetic) [72]. Applying energy to a electromagnet will cause it to polarize, one end becomes the north pole and the other the south pole. As the north poles of two different magnets are automatic repelled, the armature will spin so its north pole faces the south pole of the ‘can’ magnets. The commutator switchs the polarity of each pole everytime the pole passes a magnet. Like the armature spins, the electrical charge applied on it will flip and the poles will be again repelled, the armature spins again and consequently will rotate the pinion gear and the vehicle’s transmission.

Now comes the explanation for brushless motors. Their construction is similar to the brushed motors, except the poles (electromagnets) are stationary (no need for brushes) and everything is “inside out” [71] as shown is **Figure 3.5**. Here the permanent magnets are placed around the motor shaft becoming part of the spinning portion. This grouping is known as rotor. The coils are a place around the inside of the ‘can’ motor to generate many and differente magnet poles. These motors can have sensors on the rotor that sends signals back to the ESC – sensored

brushless motors. Note that electric motors are rated kV (RPM per Volt). *E.g.* a motor with 3500kV will spin 3500 RPM for each volt applied on it.

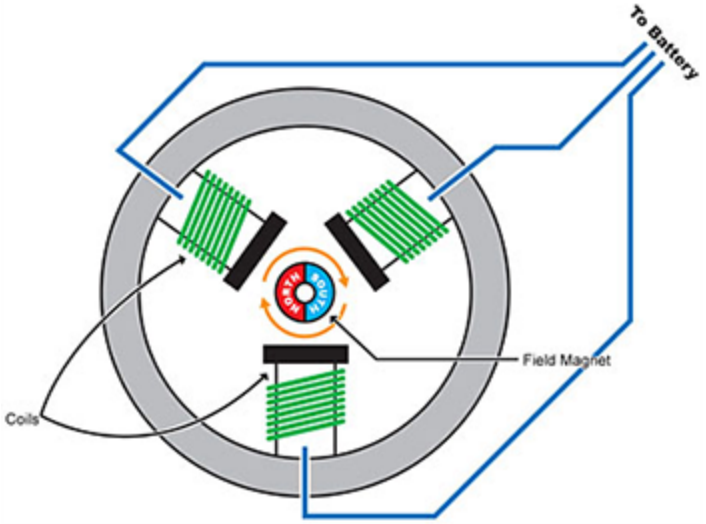


Figure 3.5 – Construction model of a brushless RC electric motor. *Source: [72]*.

Speaking about the advantages and disadvantages of both types of motors, the difference is noticeable. Brushed motors have many associated disadvantages: the brushes and the commutator wear out and must be cleaned from time to time, the friction between the brushes causes to shorter run time, battery life, slows down the motor and lows the power to weight ratio [71].

Table 3.1 – Motor features. *Source [73]*.

Feature	Measure
kV (RPM/v)	2100
Weight (g)	239
Continuous Current (A)	80
Max Current (A)	140
Resistance (ohm)	0
Max Voltage (V)	15 (4S) [1]
Power (W)	2100
Shaft A (mm) [2]	5
Length B (mm) [2]	70
Diameter C (mm) [2]	42
Can Length D (mm) [2]	62
Total Length E (mm) [2]	87

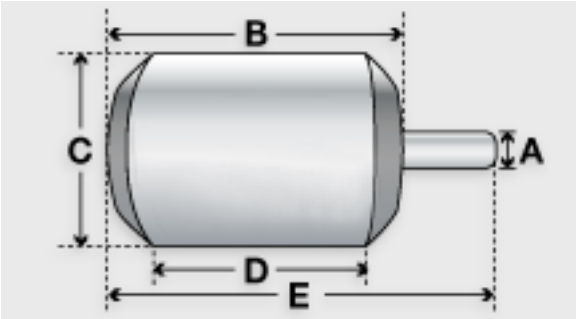


Figure 3.6 – Specification of motor dimensions. *Source: [73]*.

Figure 3.7 – Motor assembled on the vehicle frame..

[1] 4 LiPo battery cells in series.
[2] See Figure 3.6.

The vehicle is provided with a 2100kV brushless motor represented in **Figure 3.7**. Its specifications are set out in **Table 3.1**. Besides those characteristics, this motor is compatible with any sensored or sensorless ESC and is sensor-based which allows and excellent torque and low-speed drivability. In case of failure it is possible to replace the rotor since it is removable. This kind of motors offers an outstanding performance and are resistant: high purity cooper windings, sintered neodymium magnet and aluminum ‘can ’motor. They are designed for vehicles who need good RPM and reliability. A disadvantage of brushless motors are their power consumption. As they are quite rotary they will consume some power when operate in medium/high rotations.

3.2.3 Electronic Speed Controller

The Electronic Speed Controller (ESC) is an electronic circuit responsible for varying the speed, direction and dynamic brake of an electric motor. This circuits transforms the electric current that comes from the power system in a rectangular signal with a specific width that varies with the amount of energy that enters the system. By other words, the ESC converts Direct Current (DC) from battery to Alternating Current (AC) which is required by the motor. The signal is modulated using Pulse-Width Modulation (PWM), *i.e.* the pulse has always the same maximum and minimum amplitude and the duty cycle it is the only variation. That means ESC allows the signal to control the torque and speed of the vehicle. Modern ESCs incorporate an electric component that regulates the voltage for the receiver: a Battery Eliminator Circuit (BEC) or Universal Battery Eliminator Circuit (UBEC). These components convert battery voltage to 5V which is the voltage to which the receiver operates.

There is an ESC for each type of electric motor [74]:

- For brushed motors: the ESC simply limits the amount of energy delivered to the motor. To do this, the ESC sends only short pulses (short pulses, less power, slower the motor spins). The ESC is connected with the motor by two wires. To reverse the motor’s direction, simply swap the wires.
- For brushless motors: the power is controlled in the same way, however the ESC must electronically choose one of the windings in the motor to make it turn. The most recent brushless ESCs are known as sensorless ESCs and determine the position of the motor using the generated voltage in the momentarily unpowered windings. The ESC is

connect with the motor by three wires, like shown in **Figure 3.8**. To reverse the motor's direction, simply swap the two of the wires.

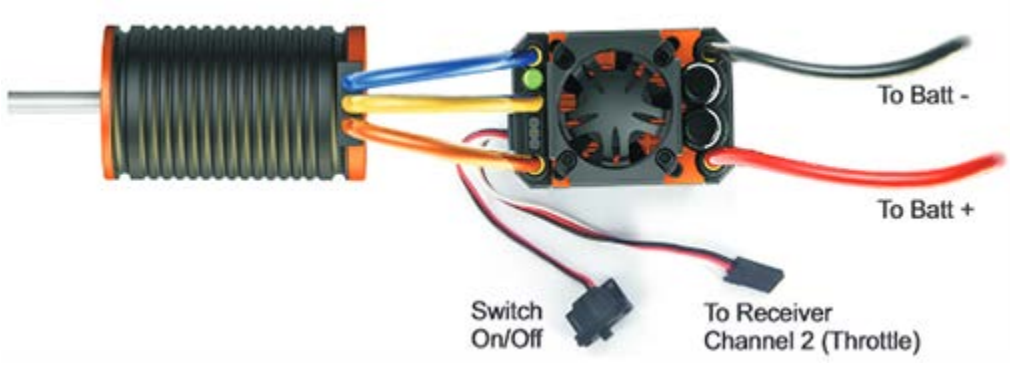


Figure 3.8 – Connection between the ESC and the motor. *Source: [75]*.

The vehicles uses a 120A waterproof and sensorless ESC, which is represented by **Figure 3.9**. **Table 3.2** resumes all its relevant features.

Table 3.2 – ESC features. *Source: [76]*.

Feature	Measure
Continuous Current (A)	120
Burst Current (A)	140
Resistance (ohm)	0
Battery compatibility	2-4 cells LiPo
BEC output (V/A)	6/3
Running mode	Forward with reverse
Weight (g)	141
Length (mm)	43
Width (mm)	39
Height (mm)	33

Figure 3.9 – ESC coupled on the vehicle.

3.2.4 Servo

Servos (or servo motors) are electric rotary motor that control the position, velocity and acceleration of machine part with great precision. They are very used by RC community due to its energetic efficiency, precision, reliability and low price. Its existence eliminates the need to have a complex and expensive control system.

A servo is composed, as shown on **Figure 3.10**, by [77]:

- Small DC motor: heart of a servo that is attached by gears to the control wheel. Moves the gears and the main shaft;

- Drive gears: reduces the motor rotation, gives more power to the main shaft and move the potentiometer with the main shaft;
- Potentiometer: connected to the servo's output, records the motor rotations (its resistance changes);
- Control circuit: regulates with precision how much should the motor rotation and in which direction.

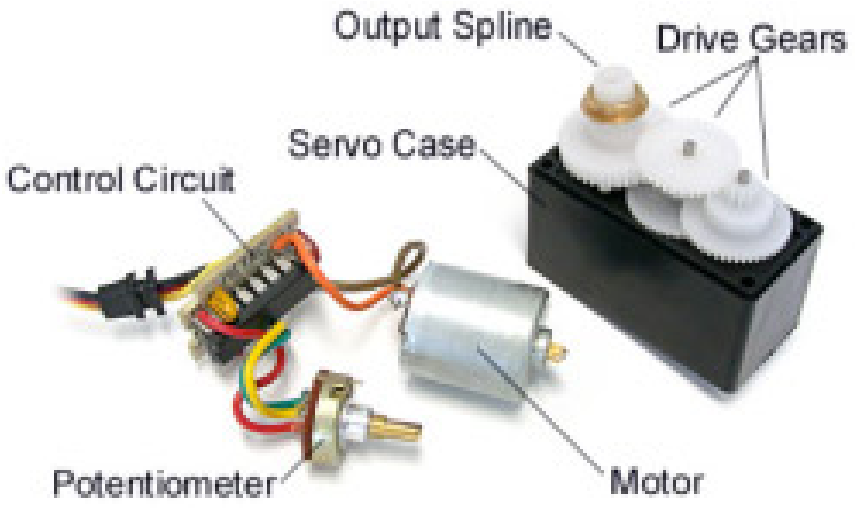


Figure 3.10 – Servo composition. *Source:[78]*.

The servo position is controlled by electric pulses of variable width, also known as PWM. There are 3 important variables to take into account: minimum pulse, maximum pulse and repetition rate. Usually a common servo only turns 90º to each direction (180º in total). The neutral position is where the servo has the same amount of potential rotation in both clockwise and counter clockwise direction [78]. The PWM received by the motor will determine the shaft's position based on pulse duration. When the shaft reaches the desired position the servo holds that position. If an external forces tries to rotate the shaft while the servo is holding a position, the servo will resist from moving out that position. Torque rating is the name given to the maximum amount of force that a servo can exert. The motor expects to receive a pulse every 20ms in order to determine the desired position, even if it is the same as the actual (the pulse must be repeated to instruct the servo to hold) [78]. Example (see **Figure 3.11**): a 1,5ms pulse will make the motor turn to the 90º position: if the pulse is shorter than 1,5ms the motor moves to 0º; otherwise it will move to 180º.

The motor's speed is proportional to the difference between its actual position and the desired position: proportional control [78]. The speed at which the motor will run depends on the necessary speed to accomplish the task: it can turn fast or slowly.

To control the position of the output shaft it uses the PWM technique. The control line feeds the servo with the pulse. This line does not supply power directly to the motor. Inside the servo there is a control chip that receives the pulse and so it does not receive much current.

In this particular case, this component is responsible for the vehicle's steering and can handles 12Kg.

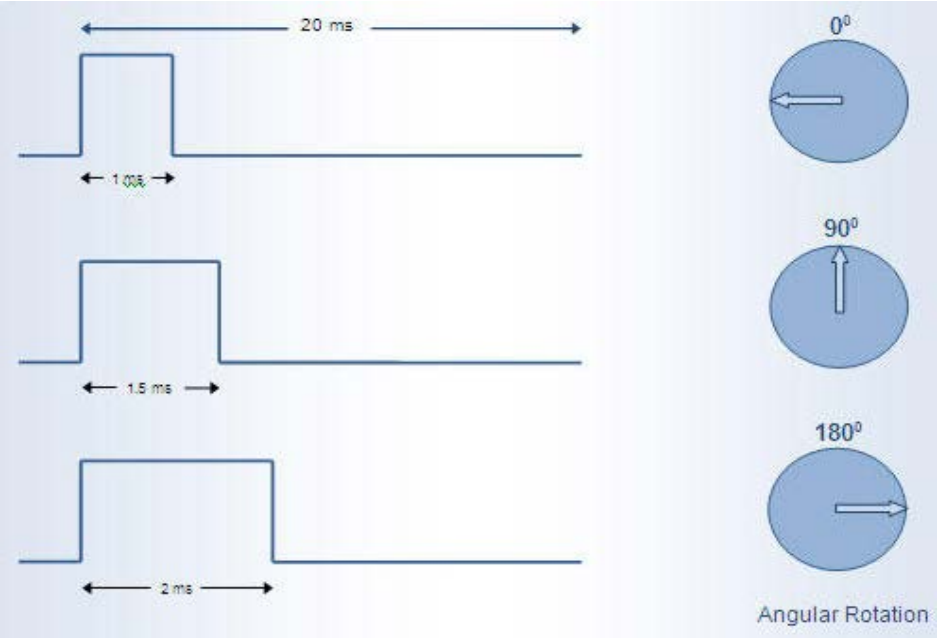


Figure 3.11 – Diagram for variable pulse width control of servo position. *Source: [79]*.

3.2.5 Power source

To power the system, LiPo batteries were chosen. This type of batteries are used in many consumer electronic devices. They gained popularity in RC industry on the last years and now are the 1st choice. The wide array of benefits they offer justifies that fact. However, each user must evaluate and decide if the benefits overweight the drawbacks. LiPo batteries inspire special care and if they are properly met there will not be any problem (at least it should not). Those cares will be described later.

Before using a LiPo battery it is necessary to understand its specifications and associated concepts. Each battery has a label which contains relevant information to know exactly how it should be used. On **Figure 3.12** it is easy to identify the specifications of a LiPo battery. The capacity of a LiPo battery measures how much power the battery can hold. The unit of measure is mAh (milliamp hour). Basically, it gives how much current can be drained from the battery to discharge it on an hour. In other words, determines the usage time of a battery before it recharges (higher capacity leads to higher run time). Usually a motor drain is discussed in

Figure 3.12 – Example of a LiPo battery with labelled specifications. *Source: [80]*.

Amperes (A) and the conversion from mAh to A can be done: 1000mAh = 1A (Ampere) [80]. UAVs do not have a standard capacity value. In turn, the UGVs typically use 5000mAh. It is correct to say that, if possible, batteries with higher capacities should be used. However, we should be bear in mind that the higher the capacity the greater the size and battery weight. Choosing the right LiPo should take it into account the motor and ESC specifications.

The LiPo battery cells as nominal voltage of 3.7V. A battery with 7.4 volts means that it has 2 cells in series (2S battery pack). A three-cell (3S) pack is 11.1V, a four-cell (4S) pack is 14.8V and so on. The voltage determines the vehicle maximum speed. It directly influences the RPM of an electric motor. With 2S LiPo the motor will spin 25900 RPM, with a 3S LiPo will spin 38850 RPM, and so on.

Every battery type has its discharge rating (or C rating). It can be seen as the measure of how fast the battery can discharge without risks. But it is a bit complex to understand because it is not a stand-alone number: it is necessary to know the battery's capacity to figure out the safe current draw. Knowing the capacity it is time for simple math: 20C = 20 x Capacity (in Amperes). Having the battery of **Figure 3.12** as example, C rating = 20 x 5 = 100A. The result of this operation corresponds to the maximum sustained load that a battery can safely handle. If in some case a higher current is provided, the battery will degrade faster or will burst into flames. So, it is very important to charge battery with the right current and voltage values. The variable C rating spoken so far corresponds to continuous current. However, batteries have another "C rating", this time associated with burst rating. As in the previous case the current is continuously supplied, in this case it is applied in 10-second burst [80]. Moreover, the mode of operation is identical. This concept does not comes into play when the vehicle is at steady speed, but when it accelerates. Usually, burst rating is higher than continuous rating. The burst rating is used to compare batteries, not the continuous rating.

LiPo batteries are very powerful but need special care, especially on charge and discharges processes. Reading the instruction manual before use is recommended.

This page intentionally left in blank

Chapter 4
BOARDS

For UV applications the “flight” controller is an essential part without which was not possible to flight or run. Usually only one board with processing and control capabilities is used, but in this case two boards are used, one for each task.

4.1 Introduction

Flight controllers can be considered a major part in setting up a UVS, since it will be through this board that the control and monitoring of the vehicle are made. For this purpose, an Ardupilot Mega (APM) 2.6 was used, running the ArduRover firmware. The APM is the most known flight controller among the hobbyist community because it is an OSP, which means that there is much of information available on matters relating to the board, especially on the Internet. Despite being a very complex component, the APM does not allow connection to a wireless network. In order to promote this mandatory connection is used a Raspberry Pi (RPI) 2 model B. A Universal Serial Bus *(*USB) dongle can be coupled to the RPI and therefore access a wireless network. It also allows coupling of a video camera, capture video and sending it to the operator via a video stream.

In short, instead of having one board performing control and processing tasks, there will be two boards: APM for control and monitoring; RPI for processing and internet access.

4.2 Ardupilot

4.3.1 Overview

APM is a pro-quality IMU autopilot based on Arduino Mega, developed by 3DRobotics (3DR) and licensed by GNU General Public License version 3 (GPLv3) [81]. As referred before, APM is the most known “flight controller” among the hobbyist community. It can fly aircrafts (fixed-wing, multicopters or helicopters) and drive rovers or boats, depending on which firmware is chosen. Since UGV and the term “flight controller” does not fit well, from now on APM refers to the autopilot controller board (or simply autopilot). This autopilot board offers a waypoint based navigation, manual navigation, GPS navigation, camera control and a two way telemetry flux. When used in aircrafts (fixed-wind or multi-rotor) it is capable to automatically stabilize the vehicle. The MAVLink is used to communicate in both directions: inside and outside. This protocol is explained further in Chapter 5.

Until today, there are 4 APM versions. The APM 1.0 is set by 2 boards: the APM board, which actually is an Arduino Mega for Input/Output (I/O) pins and processing; the other one is an IMU shield with all necessary sensors for navigation [82]. Sometime later it was decided to merge the 2 boards of version 1.0 and then form APM 2.0. Thus, APM version 2.0 is the set of an APM board, navigation sensors, processing unit and I/O pins [83]. However, it still needs a shield composed by a magnetometer, a GPS and a slot for micro SD card. Then came the new

versions, 2.5 and 2.6. These APM versions use external a GPS, eliminating the need for shields. The only difference between then is that APM 2.6 uses an external magnetometer, eliminating the existing electromagnetic interferences on APM 2.5 that uses an internal magnetometer [84]. The APM 2.6 was designed to be used with a specific GPS module with on-board compass: the 3DR GPS uBlox LEA-6H (see **Figure 4.1**). In **Table 4.1** are present the main specifications of the APM family.

Figure 4.1 - uBlox LEA-6H GPS module with on-board compass.

Table 4.1 – APM family main specifications. *Source [32][85]*.

Board	APM 1.0	APM 2.x
Processor chip	AT Mega 1280 8-bit AVR AT Mega 328	AT Mega2560 8-bit AVR AT Mega 32U-2 MPU-6000 DMP processor
On-board sensors	3-axis gyroscope, 3 axis accelerometer, barometer, magnetometer (optional)	6-axis MPU-6000 (gyroscope + accelerometer), barometer, magnetometer, GPS
Datalogging memory	2 MB	4MB
Size	40x72x20mm	40x65x10mm

APM 2.6 features an all-in design and supports 8 RC channels and 5 UARTs (Universal Asynchronous Receiver Transmitter): telemetry port, new GPS port, old GPS port, Inter-Integrated Circuit (I^2C) port and power port (see **Figure 4.2**).

Figure 4.2 – APM 2.6 with enclosure. *Source: [86]*.

4.3.2 Schematics and components

APM 2.6 internal composition and schematics is shown by **Figure 4.3**. It can be divided five sections: processing components, sensors, I/O pins and others.

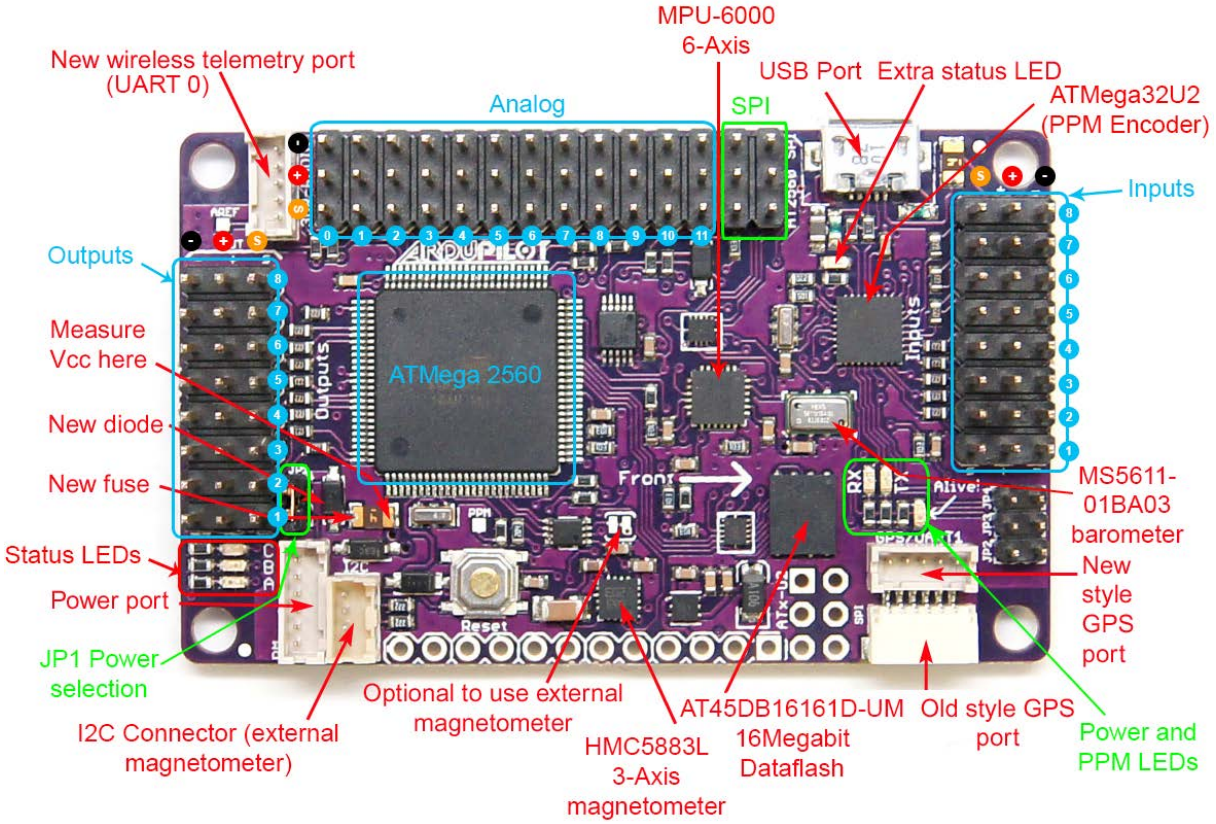


Figure 4.3 – APM 2.6 schematics. *Adapted from: [84]*.

4.2.2.1 Processing

The Atmel ATMega 2560 is a 8-bit AVR (modified Harvard architecture) microcontroller with 8KB SRAM, 256KB flash memory and 4KB EEPROM [87]. It operates at a maxium of 16 MHz, powered by 2.7V to 5.5V and have 86 general purpose I/O lines, 4 Universal Synchronous Asynchronous Receiver Transmitter (USART), PWM and 16-channel 10-bit A/D converter [87]. The ATMega 32U2 is used as a PPM (Pulse-Position Modulation) enconder, converting the radio PCM (Pulse-Code Modulation) signal to PPM for the ATMega 2560 [88]. This chip is also responsible to control the USB connection to the ATMega 2560. ATMega32U2 chip is a Atmel 8-bit AVR microcontroller with real-while-write capabilities, a 32KB flash memory, 1KB EEPROM, 1 KB SRAM, 22 general purpose I/O lines and one USART [89].

The Atmel AT45DB161D is a 16Mb that provides a Data flash memory. Is powered by 3.3V and is used for data logs [90].

4.2.2.2 Sensors

The IMU sensors are present on a MPU-6000 6-Axis I^2C MotionTracking chip that combines a 3-axis accelerometer and a 3-axis gyroscope. It has an operating voltage from 2.375V to 3.46 and is responsible for processing all sensor data, increasing the efficiency compared to cases where the processing is done by the main system [32]. The I^2C serial bus allows to compute data from any other sensor [91].

A Honeywell HMC5883L multi-chip module is used as magnetometer (or compass). This 3-axis digital compass is powered from 2.16V to 3.6V, has a low power consumption of 100μA and provides an I^2C serial bus to communicate with the microcontroller [92]. As referred before, APM 2.6 uses an external compass to avoid interferences as happens on APM versions 2.0 and 2.5 (on-board compass) [32].

Previously it was stated that APM 2.6 is designed to be used with uBlox LEA-6H GPS (Galileo ready) module. Both 2.0 and 2.5 versions of APM make use of a MediaTek MT3329 GPS. The uBlox GPS module features an on-board compass and offers a superior performance due to its newer chip and larger antenna [93]. This anti-jamming navigation module have a maximum speed of 500m/s, 5Hz update rate, high sensibility up to -162dBm, 2.5m horizontal accuracy without aid, 0.5 degrees of heading accuracy and is power from 2.7V to 3.6V [94].

Atmospheric pressure is measured using a MS5611-01BA03 module. This high-resolution and high performance barometric pressure sensor can easily perform altitude readings from a range between 10mBar and 1200mBar. It has an Serial Peripheral Interface (SPI) and a I^2C serial ports and operates up to 20MHz with a power supply of 1.8V to 3.6V [32][95].

4.2.2.3 Powering up APM board

APM kit comprises an APM Power Module (PM) (see **Figure 4.4***Figure 4.4 – APM power module.* Source: [97].) that supplies 2.25A at 5.37V [96]. This module is responsible for supplying energy to the system and to the RC receiver (without servos). To power up the RC receiver, any APM input can be used (+5V and ground). A dedicated power supplied is required to supply additional servos, because APM only controls servos (does not supply them). It also monitors the status of a LiPo battery, its voltage and current.

To provide additional energy to the APM is first necessary to understand its power circuit. Observing **Figure 4.5**, it is possible to see that APM has two separated positive rails (red lines) that power the sensor pins and I/O. The use of these scheme is due to the fact that APM has to

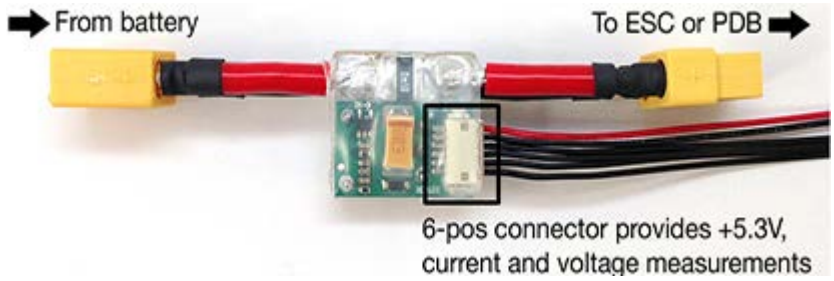


Figure 4.4 – APM power module. *Source: [97]*.

provide power to its internal components and external systems (*e.g.* RC receiver, telemetry modules) and also because it connects servos and other devices. These components can, in some cases, draw too much current and burnout the APM. The separated positive-rail will avoid that: on rail for the Outputs (servos and ESCs) supplied by and independent power source; the other rail for the Inputs and the Analogs (RC receivers and sensors) supplied by the PM [96]. In cases that servos are not used the Outputs can be powered through the PM, placing a jumper between two pins (JP1 in **Figure 4.5**).

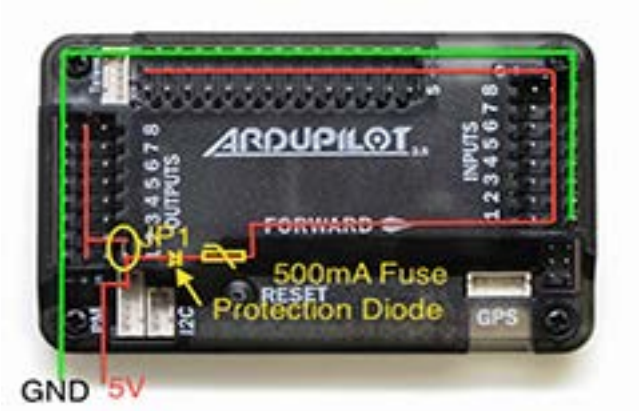


Figure 4.5 – APM 2.6 dual positive rails. *Source: [96]*.

4.2.2.4 Digital and Analog I/O

APM 2.6 has total of 16 digital I/0 pins, 8 for each case. Depending on the installed firmware these pins will have different functions. In the case of ArduRover, usually only 4 channels are used, as shown in **Figure 4.6***Figure 4.6 – RC input channels for ArduRover.* Adapter from: [99].: channel 1 for steering, channel 3 for throttle, channel 7 for waypoint record and channel 8 to switch navigation mode. The other inputs can be used to perform other kind of actions defined by the user. As the outputs, only two channels are used. The connection will depend on the vehicle [98]:

- For cars: channel 1 for servo and channel 3 for ESC/Motor;
- Skid steer (like thanks): channel 1 for left motor and channel 3 for right motor.

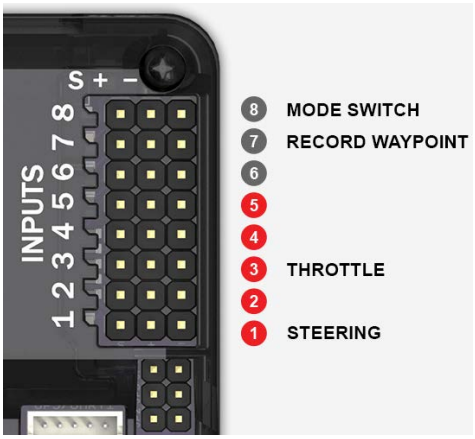


Figure 4.6 – RC input channels for ArduRover. *Adapter from: [99]*.

The analog pins can be used for any general analog input (*e.g.* airspeed and sonar) and take up to 5V [91]. They have a positive-rail ("+") that is connected with the Inputs positive rail, a common negative-rail that they share with both Outputs and Inputs and individual pins for signal ("S") to connect sensors and servos. **Figure 4.7** shows with some detail the utility of each pin.

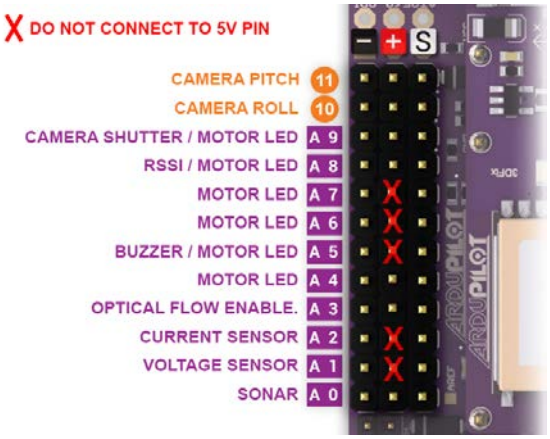


Figure 4.7 – APM 2.6 Analog pins. *Adapted from: [100]*.

4.2.2.5 Others

The UARTs (or serial ports) are interfaces that allow a simple connection between APM and the external devices. The PM port allows to connect a PM and monitor a battery; the I^2C serial bus provides a connection to multiple peripherals devices on an easy way; GPS interface (both old and new) is the input for GPS module; Telemetry port allows to connect a device for a 2-way telemetry flux and control [101].

APM has a micro USB port that provides a direct connection to a computer.

The Lightning Emitting Diodes (LEDs) indicates the current state of the vehicle, autopilot and GPS. They are properly identified on **Figure 4.3** and its meaning described on **Table 4.2**.

Table 4.2 – APM LEDs behaviour and meaning. *Source:[102]*.

LED	Behavior
A (red)	Solid = armed, motor(s) will spin when throttle raised Single Blink = disarmed, motor(s) will not spin Double Blink = disarmed, motor(s) will not spin, cannot arm because of failure in pre-arm checks
B (yellow)	Only flashes along with A and B during calibration or as part of the in-flight auto trim feature
C (blue)	Solid = GPS working, 3D lock Blinking = GPS working, no lock OFF = GPS not attached or not working
Power	ON when powered
PPM/Serial	Flashes when data is being transmitted

4.3 Raspberry Pi

4.3.1 Overview

Raspberry Pi (RPI) is a credit-card sized computer developed in United Kingdom by RPI Foundation. Originally is was setup to promote the basic computer science teaching in schools [103]. In order to use this low cost minicomputer, a computer monitor or a TV can be plugged as well as a keyboard and a mouse. The big amount of interfaces allows the users to have a lot of features and possible applications: USB ports, Camera Serial Interface (CSI) and Display Serial Interface (DSI) to connect a camera and external display, respectively and General Purpose Input/Output (GPIO) pins to attach other devices.

It all begins on February 2012 with the release of RPI model B. The product became so popular that in April 2013 the model A was release. On July 2014, the model B comes up and immediately afterwards in February 2015 RPI Generation 2 of model B (RPI 2) arises. Comparing all models, there are significant differences, with the exception of these last two. Despite the few differences, they are quite significant and cause an exponential increase in the capacity and performance of the device. Increasing the frequency of the Central Processing Unit (CPU) and the available Read Access Memory (RAM) enabled the expansion of the RPI for projects and applications where it was previously not possible. The **Table A.1** present on **Annex A** shows all the important specifications of RPI versions, according with [104][105].

RPI is capable to perform normal tasks of an ordinary desktop: browsing, playing High-Definition (HD) video, gamming, *etc*. RPI Foundation provides two OS for download that RPI2

can run: Raspian (Linux distribution based on Debian Wheezy) and OpenELEC. All the others OS like Pidora, RASPBMC and RISC OS can only run in older RPI versions. Microsoft also released a Windows 10 version that supports RPI 2. This was a great improvement, since Windows was not present on RPI's market [106].

The RPI Foundation recommends Python programming language for learners and Scratch for young kids. However, any language compiled for ARMv6 can be used, with no restriction on Python. RPI has other installed languages like C, C++, Java, Pearl, Ruby and Scratch [107].

The RPI 2 was chosen as processing unit of the developed system. The reasons that support this decision are very simply: low cost, high performance, capability to attach external devices (*e.g.* Wi-Fi and 3G/4G dongles) and the great support provided throughout the RPI community and hobbyists. It operates at 900MHz by default but can be overclocked to 1000MHz and its voltage range is between 4.75V and 5.25V [108]. This minicomputer is powered by 5V micro USB. The power requirements will depend on the type of applications. A 1.2A power supply should handle RPI for most applications; if the 4 USB ports are in used, a 2.5 power supply is needed [109].

4.3.2 Schematics and components

Inside a RPI 2, 12 main components can be founded as shown in **Figure 4.8**.

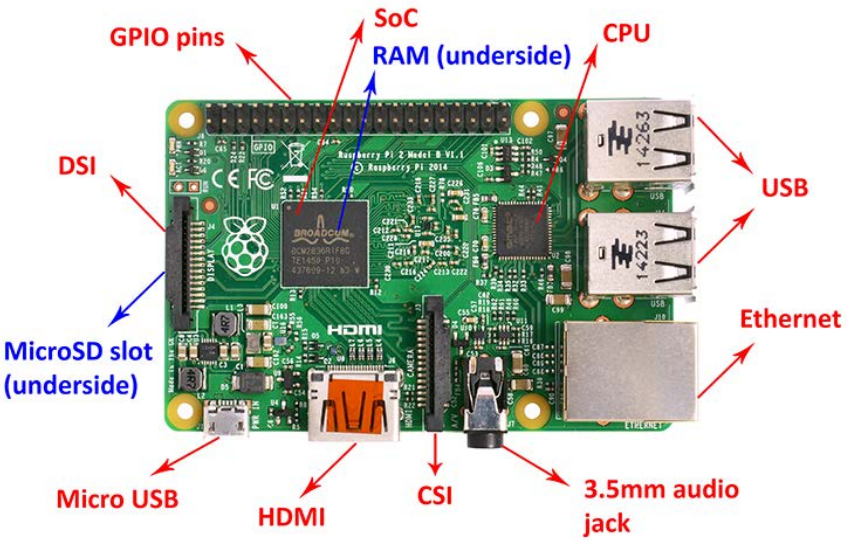


Figure 4.8 – RPI 2 top schematics. *Adapted from: [110]*.

Based on model B, the RPI 2 has four USB ports where can be attached the most usual USB devices, from keyboards, mouses, USB storage, USB Wi-Fi/3G/4G dongles, *etc*. Thus, many functionalities can be added on an easy way. The ethernet port provides internet access on an easy way, just needing to connect to an Ethernet cable.

From the multimedia's point of view it has a 3.5mm audio jack where can headphones can be connected, speakers, *etc*, and a HDMI port where the user can connect a TV or a computer monitor. There are also 2 interesting intefaces: the CSI and the DSI. The CSI is a connection point to a camera and the DSI to a display.

As SoC, the RPI 2 presents a new face to its predecessors: the Broadcom BCM2836 replaces the Broadcom BCM2835. This specific part consists a single chip composed by a set of other components (GPU, CPU, USB controller, RAM, *etc.*) that allow a computer to run. Instead of using those components as spare parts, they are used together as one [111]. The older versions use a single-core ARM1176JZF-S running at 700MHz and Videocore 4 GPU. The new version was awarded with a quad-core ARM Cortex-a7 CPU running at 900MHz and the same Videocore 4 GPU [112]. The CPU can run happily at 1000MHz, just needs to be overclocked. To improve its function in that cases, heat sinks can be used to dissipate the heat.

At last but not least, the GPIO pins. These pins allows the RPI to connect with the external world. On an easy way, these can be seen as switches that the user can turn on/off (input) or the RPI turn on/off (output). The inputs can be data from a sensor or a signal from other device; the outputs can be anything (*e.g.* turn a led on a led, send a signal to other device). The RPI 2 has 40 dedicated pins: 26 are GPIO pins and the others are power and ground pins and 2 ID EEPROM pins [113]. As shown in **Figure 4.9***Figure 4.9 – RPI2 GPIO header.* Source: [114]., pins 8 and 10 are UARTs, pin 12 is PCM, pins 3 and 5 are I^2C and pins 19, 21, 23, 24 and 26 are SPI. Note that the high voltage of these pins is 3.3V!

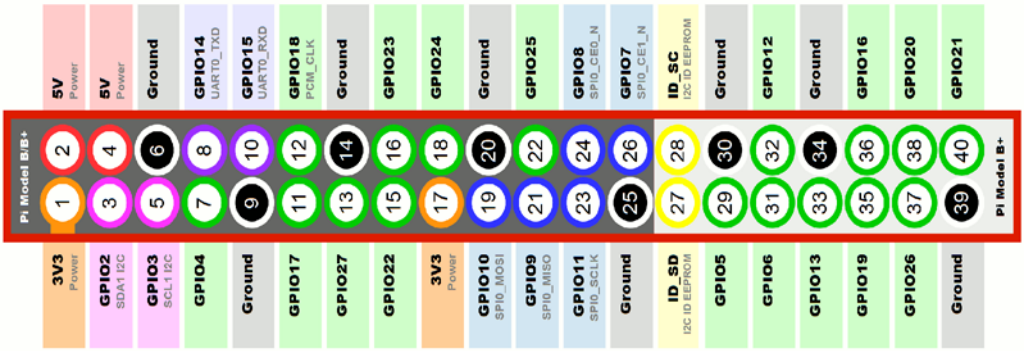


Figure 4.9 – RPI2 GPIO header. *Source: [114].*

4.3.3 Video camera

Currently, RPI has 2 official cameras (see **Figure 4.10**): the RPI camera and the Pi NoIR (NoIR = **No I**nfra**r**ed). Both cameras has a 15cm ribbon cable that connects to the RPI's CSI port. The RPI camera module supports 1080p30, 720p60 and VGA90 video modes, compressed in H.264 with its 5megapixel camera. It can be used to photographs and HD video and is compatible with

all RPI models [115]. Pi NoIR is also compatible with all RPI versions and has only one difference compared with the regular camera which is not implements an infrared filter [116]. The infrared lightning makes possible to see in the dark. In both cases, MMAL and V4L APIs provide access to the cameras and there are many libraries created for it, *e.g.* Picamera Python library [117].

Figure 4.10 – RPI regular camera and Pi NoIR, respectively. *Source: [118][119]*.

4.3.4 Connection to the APM

As referred before, APM works at 5V and the RPI's GPIO pins at 3.3V. Thus, there is a need to convert this values in order to avoid any of the equipment (in this case the RPI because it works at a lower voltage) from being damaged. A Logic Level Converter (LLC) was the adopted solution. This small circuit that can convert 4 I/O lines in both ways: low-to-high and high-to-low. **Figure 4.11***Figure 4.11 – Connection between APM 2.6 and the RPI 2.* shows how the connection is accomplished.

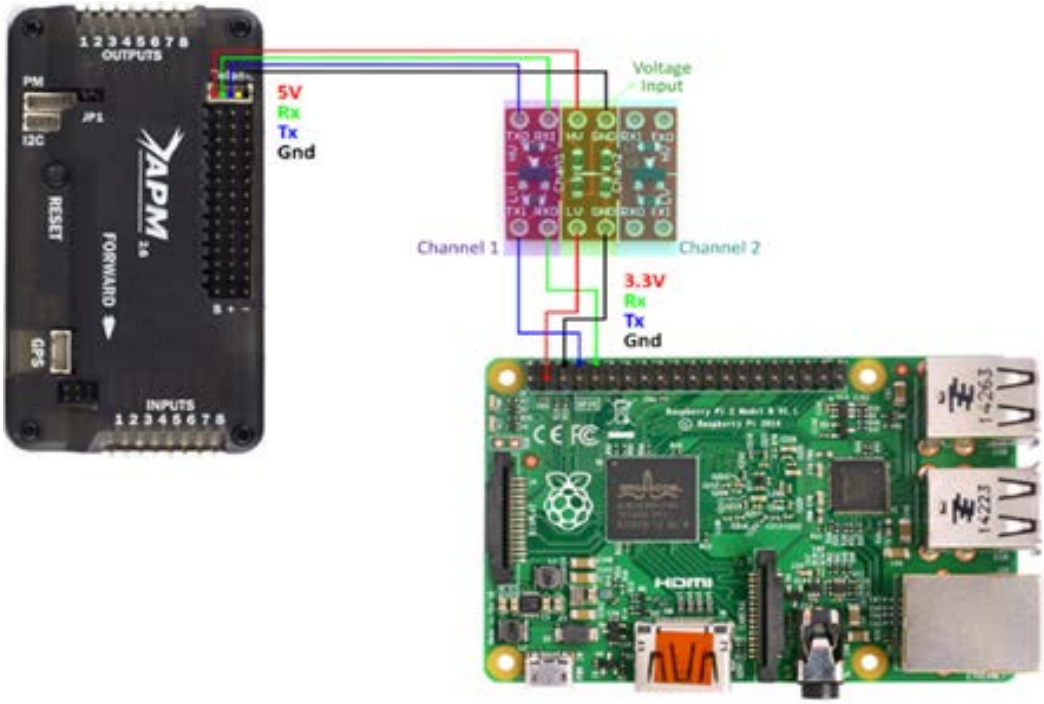


Figure 4.11 – Connection between APM 2.6 and the RPI 2.

This page intentionally left in blank

Chapter 5
COMMUNICATIONS

This chapter contains all the communication part made by developed and/or installed components. The focus is on wireless communications and the vehicle communication protocols. -

5.1 Introduction

This chapter can be divided in two main topics: the MAVLink for APM bi-directional communication and the communication between RPI and the GCS through wireless networks. They are designated as on-board communication and out-board communication, respectively, and are described below. Note that the initial objective was the implementation of the system in a vehicle true to scale, *i.e.* an ordinary car. As such, a study was made on the Controller Area Network (CAN) communication protocol, which will also be presented, although it was not used.

5.2 On-board communication

5.2.1 Micro Air Vehicle Link

MAVLink is an application level communication protocol designed by Lorenz Meier in 2009. Originally built for air vehicles, it allows the communication between a GCS and the autopilot (*e.g.* Ardupilot Mega). It is a header-only message marshalling library that defines a set of telemetry messages, the encapsulation method and internal communication to transmit the information over a serial communication channel, wired or wireless [120].

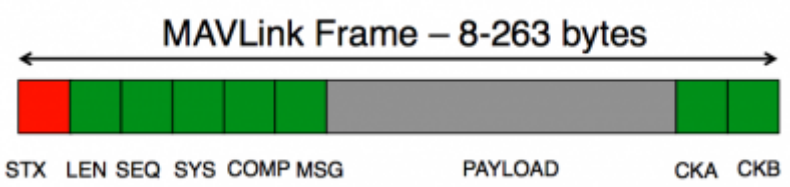


Figure 5.1 – Structure of a MAVLink message. *Source:[121]*.

A MAVLink message is simply a stream of bytes encoded by the GCS and sent to the autopilot (MAVLink compatible). The referred encoding is nothing more than putting the packet within a data structure and sent it through the channel with some error correction. A MAVLink packet has a minimum length of 8 bytes and a maximum length of 263 bytes. The structure of MAVLink message is represented by **Figure 5.1***Figure 5.1 – Structure of a MAVLink message.* Source:[121].. CAN was one of inspirations for this anatomy. The **Table B.1** of **Annex B** resumes the content of a MAVLink packet as well as the value and description of each one, according with [121]. It is open-source code that allows to define new messages using XML files. The default MAVLink messages type can be found in [122].

Analyzing the structure of MAVLink messages, the most important fields are:

- System ID: indicates the source of the message (*e.g.* the GCS). The message is sent to the autopilot board via USB or telemetry port and the software does regular checks to be sure that the message is for itself;
- Component ID: also known as the subsystem within the system, *e.g.* IMU sensors, GPS, *etc*. It is needed because multiple components could send multiple messages from the same system. It also allow to address a message for a specific component [123];
- Message ID: identifies the message in question in order to know how to proceed: decode and execute the right action;
- Payload: the juice of the message, the real goal.

The software evaluates if the message is valid by checking the checksum: if it is corrupted, the message is discarded. It is due to this fact that the baud rate for telemetry is 57.6bps (not the traditional 115.2bps). The lower the baud rate the less prone to errors the channel will be. However, this leads to a lower update of the telemetry information for the GCS. In theory, with 57.6bps of baud rate gives a 1.6km radius coverage with a 3DR telemetry radio [124].

At the end of each MAVLink packet are defined two bytes of checksum. This error checking algorithm uses the ITU X25/SAE AS-4 hash and the calculus are made using the header (excluding the byte, STX, and the data) [121].

In order to send controls to the autopilot, the RC_CHANNELS_RAW (#35) message is used. This message allows to overwrite the RC channels and thus control the autopilots output. To performed pre-programmed missions and perform actions with autopilot's parameters, MAVLink has internal protocols to do that: waypoints protocol [125] and parameters protocol [126], respectively. Both are designed to work like a state machine, which means the system is in only one state at a time. Its state could change when an operation is concluded or fails (see **Annex C** for a detailed explanation). There are MAVLink messages that are always being transmitted, regardless the state of the machine, with a frequency of 1Hz [122]: the HEARTBEAT (#0) allows it to know if the connection is still active or some failure occurred; the SYS_STATUS (#1) indicates the general system state; and the AUTOPILOT_VERSION (#148) informs about the version and capability of autopilot software.

5.2.2 Controller Area Network

5.2.2.1 Description

The Controller Area Network (CAN) is a serial bus communication protocol developed by Robert Bosch in 1983. CAN is a message-based protocol originally designed for in-vehicles networks, but nowadays it can be found in many other contexts of networked embedded control [127][128]. In automotive, electronic devices are connected via CAN with bitrates up to 1Mbit/s [127]. CAN is used in real-time applications to offer an efficient and reliable communication between electronic devices: sensors, actuators, controllers, *etc*. [127]. CAN does not send large blocks of data from node A to node B (point-to-point) like a traditional network with a supervision of a master (*e.g.* Ethernet, USB). Instead, it is a message broadcaster service with message addressing [129]. This means that all nodes receive all messages but only the destiny nodes will really read them (see **Figure 5.2**). Since it has a multi-master architecture, each node can access the bus autonomously [127]. Before CAN, automotive industry used wired point-to--point connections between all electronic devices, resulting on an expensive and heavy bulk wire hardness (see **Figure 5.3**) [131]. Another important feature of CAN is that the transmission does not depend on the type of nodes. Thus, new nodes can be added and the system will work normally without the need of change software and/or hardware.

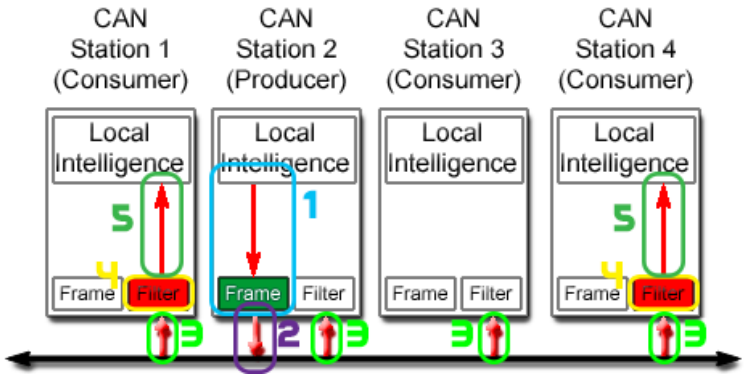


Figure 5.2 - Local filtering by CAN hardware. *Adapted from: [130]*.

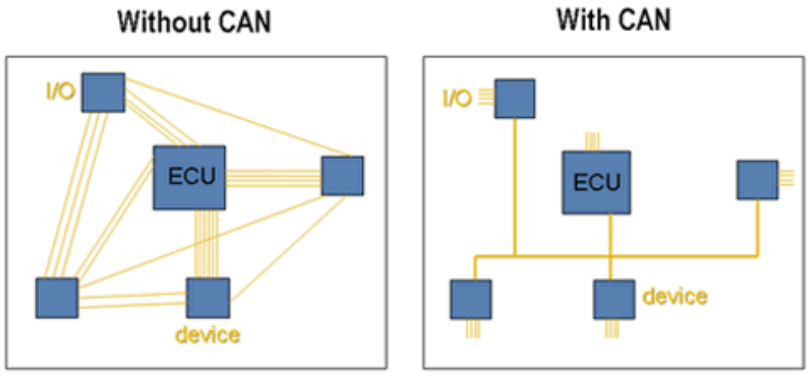


Figure 5.3 - Difference between a system without and with CAN. *Source: [131]*.

5.2.2.2 CAN layers

The CAN protocol is a standard defined by International Standardization Organization (ISO) in ISO-1898:2003 **Error! Bookmark not defined.**. This document specifies how information is transmitted between devices on a network conforms Open Systems Interconnect (OSI) model. In order to achieve transparency and flexibility CAN was divided into layers according to the OSI model [127][128][130]:

- Data-link layer: have the Medium Access Control (MAC) sublayer and the Logical Link Control (LLC) sublayer. The MAC can be seen as the kernel of the CAN, since it presents the messages received from the LLC and accepts the messages to be transmitted to the LLC. Its main concerns are Message Framing, Arbitration, Acknowledgement, Error Detection and Signaling. It is supervised by a self-checking mechanism – Fault Confinement – that distinguishes real failures from small disturbances.

  The LLC sublayer is responsible for Overload Notification, Message Filtering and Recovery Management;
- Physical layer: defines the method used to transmit the signals and deals with the Synchronization, Bit Encoding and Bit Timing. Driver/receiver features for physical layer are not defined in order to optimize the transmission medium and signal level implementations for a specific application.

The aforementioned responsibilities of each layer and sublayer are visible on **Figure 5.4**. In the OSI reference model, the data-link and physical layers correspond to the two lowermost layers.

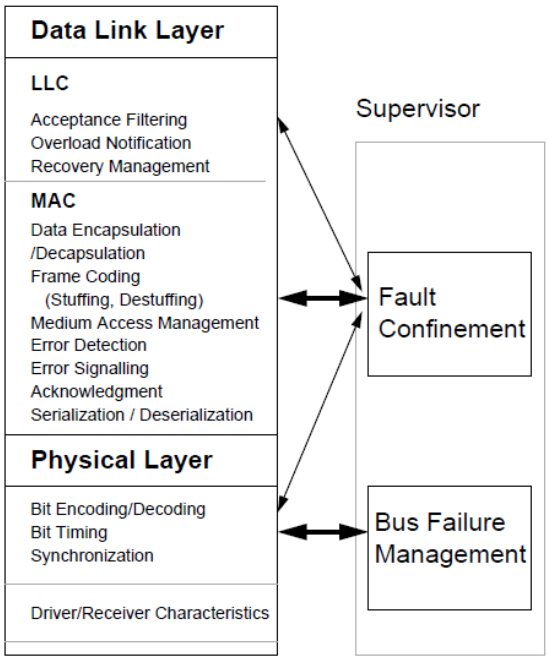


Figure 5.4 – Layered architecture of CAN according to the OSI reference model. *Source: [127]*.

5.2.2.3 Message frame of Standard CAN and Extended CAN

The main difference between Standard CAN and Extended CAN are the number of bits of the identifier. The Standard CAN have a 11-bits identifier that provides a total of 2^{11} = 2048 different message identifiers, whereas the extended CAN with its 29-bit identifier provides 2^{29} = 537 million identifier [128]. **Figure 5.5** represents the frame of a Standard CAN message **Error! Bookmark not defined.**. The Start-Of-Frame (SOF) indicates the start of the message and is used to synchronize the nodes. To define the message priority the Identifier it is used. The Remote Transmission Request (RTR) is dominant when another node requires information. The identifier determines the target node, since all nodes will receive the request. In order to distinguish Standard CAN frame from Extended CAN frame, the Identifier Extension (IDE) bit assumes a specific value: dominant for Standard CAN frame and recessive for Extended CAN. The reserved bit (r0) is not used until today, it exists for future implementations. To know the number of data bytes being transmitted, it should be noted the Data Length Code (DLC). The Cyclic Redundancy Check (CRC) contains the checksum that guarantees the system integrity by error detection. All nodes ("interested" on message or not) that correctly receives a message sends an Acknowledgment (ACK) dominant bit. The transmitting node sends a recessive bit and checks the bus looking for an ACK: if no ACK is detected (Acknowledgment Error signal) it retransmits the message. The end of the frame is marked by End-Of-Frame (EOF). The InterFrame Space (IFS) is the number of bits separating consecutive messages.

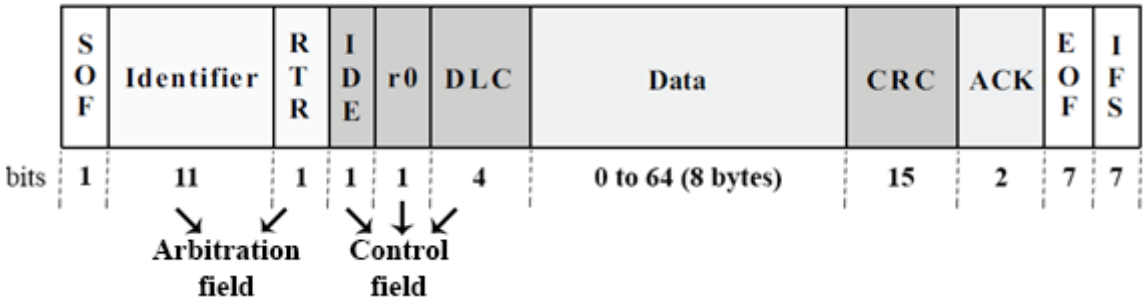


Figure 5.5 – Standard CAN message frame. *Adapted from [129]*.

The **Figure 5.6** represents the Extended CAN message frame that is quite identical to the Standard CAN frame. Although, some differences can be found. The Substitute Remote Request (SRR) field replaces the "old" RTR. The IDE bit is now recessive to indicate an Extended CAN frame. At last, there is one more reserve bit (r1). Basically, the main difference is the size of the identifier field.

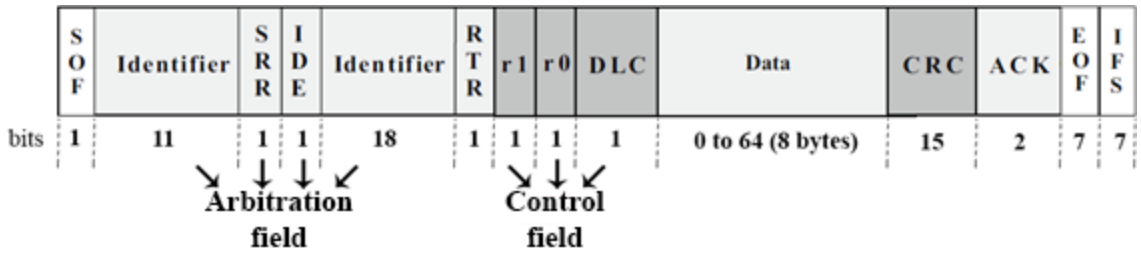


Figure 5.6 – Extended CAN message frame. *Adapted from: [129]*.

5.2.2.4 The CAN Bus

CAN have several different physical layers that can be used, the most common being the High-Speed/FD CAN, defined in ISO 11898-2 (see **Figure 5.7***Figure 5.7 – CAN physical layer standards. Source: [133].*). This standard defines a maximum data rate of 1Mbit/s on a bus with 40m and a maximum of 30 nodes, and a twisted-pair cable. The Low-Speed CAN is also very used but the transceivers only support a maximum of 125Kbit/s (defined in ISO 11898-2). Usually, High-Speed CAN is used by antilock brake systems, engine control modules and emissions systems and, CAN Low-Speed transceivers are used in convenience areas [127].

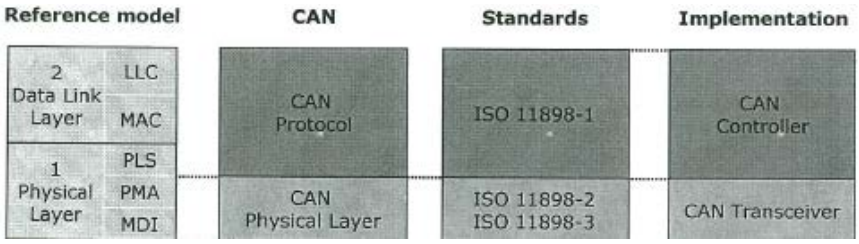


Figure 5.7 – CAN physical layer standards. *Source: [133]*.

The transmission of signals in a CAN network is a differential signal transmission in order to improve noise immunity. Thus, the CAN bus has two lines: CAN-High (CANH) and CAN-Low (CANL). Both signal lines of CAN bus are stabilized at ≈ 2.5V when in recessive mode. Transition to the dominant state leads the CANH to ≈ 3.5V and CANL to ≈ 1.5V, as shown in **Figure 5.8**. Thus, a 2V differential signal is created [132].

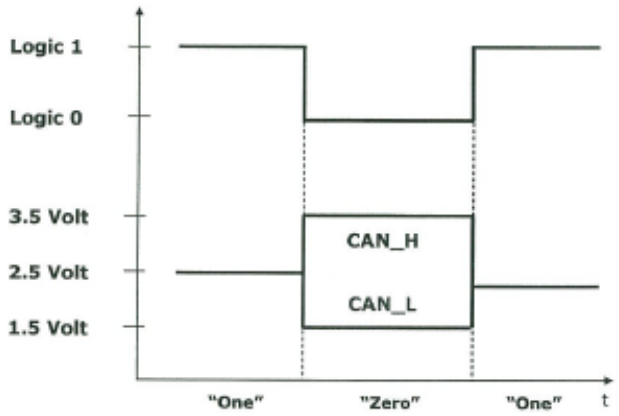


Figure 5.8 – High-Speed CAN signal lines. *Source: [133]*.

5.2.2.5 Arbitration

CAN listens to the network in order to avoid collisions, being a Carrier-Sense Multiple Access with Collision Detection (CSMA/CD) protocol and arbitration on message priority (AMP). By other words, each node must wait a specific time period before transmitting and that the collisions are resolved via a bit-wise arbitration based on messages' priority (defined on identifier field). Priority is inversely proportional to the identifier: the highest identifier always have guaranteed access to the bus, as shown by **Figure 5.9**.

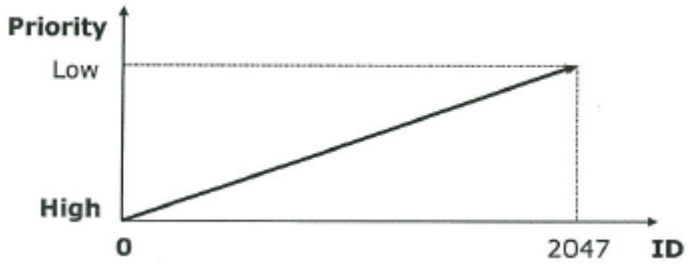


Figure 5.9 – Relation between Message ID and priority. *Source: [133].*

Bus access is a random and driven event. When 2 nodes try to access it simultaneously the conflict is resolved using bit-wise arbitration. This is also a nondestructive task, since the message of the winning node is protected against destruction or corruption by another node. If a node sends the last identifier bit as zero (dominant) and the other nodes sends one (recessive), it retains the bus access. All the recessive nodes become receivers until the bus is available again. These behaviors can be seen in **Figure 5.10**.

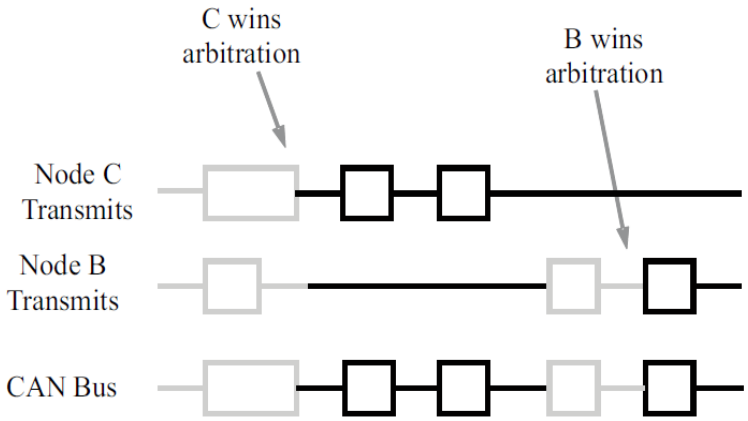


Figure 5.10 – Arbitration on a CAN bus. *Source: [128].*

5.2.2.6 Implementation: main challenges

Implementing the CAN protocol on a system is no task. When a car is developed, the CAN messages associated with it are unique or at least they should be. For the general people it is

hard to access these messages and know exactly the action they will perform. The manufactures are the only ones who have those specifications because they define them. As it is classified information, new options had to be thought out: reverse engineering of CAN messages, using electronic signal measure devices and a repeating patter; install auxiliary modules that provide direct access to the actuators without entering the CAN bus. Both ideas have been tried but without success. Reverse engineering of a binary signal without a reference is a hard and time consuming job. Adding to this the fact that some messages are generated by the controlling using random variables and other through sensors activated by external forces.

Develop auxiliary modules to execute physical actions directly on the actuators, *e.g.* a servomotor to interact with the steering motor. However, this solution deviates slightly from the initial objective of the project. Furthermore, the development of these modules would not be easy, since there are almost inaccessible areas where the device would have to be assembled.

In order to complete the project, and derived to its deadline, a model car was chosen, as already described.

5.3 Out-board communication

5.3.1 Wireless networks

A wireless network is required due to the absence of the operator on the vehicle. The telemetry and video must be transferred remotely to ensure the control and monitoring of the vehicle. The real-time control implies that the video has a minimum delay possible. By its turn, telemetry messages, namely control (throttle and steering), must have a reliable scheme to implement reliability – the loss of a message may cause an accident.

Telemetry messages have a maximum size of 263 bytes and are sent from a range of frequencies from 1 to 20Hz (can go up to 50Hz but it is not recommended). These messages are the readings from the sensors and so the referred frequency is the sensor's reading frequency. Thus, telemetry requires a maximum bitrate of 5Mbit/s. Meanwhile, the video streams requires a bit more than the telemetry messages. This bitrate will depend on the video quality measured by the number of Frames-Per-Second (FPS) and video encapsulation algorithm.

Cellular networks only supported voice in the first instance. The Global System for Mobile Communication (GSM) was originally designed as a circuit switched system for voice telephony and to operate in 900MHz (and soon in 1800MHz) [134]. The General Packet Access

Service (GPRS) was introduced to provide access to packet switched networks. It offers 14Kbit/s on uplink and 40Kbit/s on downlink and latencies up to 1 second [135][136][137]. Further, Enhanced Data rates for Global Evolution (EDGE) was implemented to reduce latency and improve spectral efficiency, providing bit rate 270Kbit/s and 100ms latencies, thus meeting the ITU's IMT-2000 requirements for 3G technology [137][138]. EDGE become known as the 2.5G, the bridge between 2G and 3G [138].

The 3G mobile networks can with the UMTS, developed by 3rd Generation Partnership Project (3GPP) to offer higher bandwidths than its successor GSM [133]. UMTS is based on Wide-band Code-Division Multiple Access (W-CDMA) and has 5MHz channel bandwidth [139][140]. It is capable to offer a 340Kbit/s data rate and low latencies around 100ms [9][141]. Nowadays, with network improvements and UMTS new releases, it is possible to achieve 2Mbit/s using UMTS networks.

The original scheme of UMTS was improved, emerging HSPA [142]. This is the pair of High-Speed Download Packet Access (HSDPA) and the uplink equivalent (HSUPA) [142][143]. With the later enhancements, HSPA saw its latencies down to 100ms and its data rate increase: 14.4Mbit/s for HSDPA and 5.7Mbit/s to HSUPA [9][141][143].

The 3G networks have another technologies that result from the enhancement of HSPA: HSPA+ and Dual Cell-HSPA (DC-HSPA) [144]. The first one, not only decreases its latency to 50ms but also augment its data rate up to 28Mbit/s and 42Mbit/s (with a single 5MHz carrier), especially due to the higher order modulation and the Multiple-Input Multiple-Output MIMO antenna scheme used in downlink [9][141][145]. By its turn, DC-HSPA introduces a new scheme of aggregation of two adjacent bands (two 5MHz carriers) resulting on a 10MHz bandwidth and, theoretically, doubles the sustained data rate to 84Mbit/s [145].

The LTE is the result of the enhancement of the HSPA+ and it is known as 3.99G technology because it almost fulfils the requirements for a 4G standard [141]. LTE's idea is to enable higher speeds along with lower latencies, both in big quantity [141]. This purely IP-based technology provides 100Mbit/s to downlink, 50Mbit/s for uplink and a maximum packet latency of 10ms [9][141][146]. Since LTE is based on IP (supports IPv4 and IPv6), the data sent by the User Equipment (UE) is supported by IP protocol and, this way the GSM (voice component) disappears form core network. As there was no provision for voice, the Voice over LTE (VoLTE) was added [141].

Finally comes the 4G of mobile communications – LTE-A. This technology fulfils the requirements proposed by International Telecommunication Union - Radiocommunication (ITU-R) for a 4G standard [146][147]. LTE-A network is capable to provide high data rates, 3Gbit/s for downlink and 1.5Gbit/s for uplink, with a maximum latency of 5ms [9][146][148]. This 4G technology increases both network capacity (allows more users) and spectral efficiency [149].

5.3.2 IP-based communication problems

Along the time, more and more Internet connections from different devices were made, leading to a shortage of IPv4 addresses. In order to resolve that problem in short time, Internet Engineering Task Force (IETF) introduced Classless InterDomain Routing (CIDR) and Network Address Translation (NAT) [150]. However, this is a temporary solution to effect the transition from IPv4 to IPv6, which was not cleaned soon as their implementation is time-consuming. Whereas IPv4 uses a 32-bit address, IPv6 is based on a 128-bit address. The number of IP addresses available is therefore increased from 2^{32} = 4 billion to $2^{128} \approx 3.4 \times 10^{38}$ [151]. The presence of NATs on the network is to create sub-networks and, thus, have more IPv4 address available.

All networks that make use of NATs have the same problem when facing a Peer-to-Peer (P2P) connection – knowing each other IP address to stablish the communication. The NAT creates a private network and acts like an interface, mediating the connection between that network and the public network [151]. NAT maps a public IP address to the private network, allowing multiple machines to be connected to that network. Thus, an external machine cannot communicate directly with an internal machine [152]. To do that, NAT changes the IP address and port number of incoming/outcoming packets to its reflexive IP address [153][154].

Besides the presented disadvantages, it has other negative impacts: requires the implementation of NAT traversal protocols and diversity of NAT types [155].The existing types of NAT are: full cone NAT, symmetric NAT, restricted cone NAT and port restricted cone NAT [155]. Regarding NAT traversal protocols, these are: STUN defined on RFC 5389 [156], TURN defined on RFC 5766 [157] and ICE defined on RFC 5245 [158].

5.3.3 Routing Server

To solve NAT's problems, an alternative solution was implemented: a routing server with public IP address. Its purpose is simple: as its IP address is public, every peer knows it. Thus, every peer can send its packets to the server that will route them to the other peer.

This routing server avoids the complexity associated with the implementation of NAT traversal protocols and reduces the amount of transmitted data, since this data does not need to be encapsulated in other specific packets of these protocols. However, this implementation entails other problems which can be considered tolerable by the system in an initial phase of testing and feasibility study. These problems are security and communication delays. With the current state of communications the "additional delays", *i.e.* the use of a third part, not being a direct communication between peers, is not a worrying factor. As for the safety factor, data protection mechanisms and safeguards against intruders were not added. As the complexity of the system is a little high and meeting deadlines is imperative, it was necessary to make a choice and consequent exclusion of certain factors. Since this is not a commercial product, the choice was not to implement security, as it would take time to develop.

To execute the actions of a routing server, a Virtual Private Server (VPS) running a java application are used. This application – Routing Server – is described further on **Chapter 6**.

Chapter 6
CONTROL AND MONITORING APPLICATIONS

All the applications used to control and monitor the vehicle are presented on this chapter. From GCS to routing server and raspberry scripts, each one has its importance.

6.1 Introduction

To start the chapter, the system architecture is available as well as an explanation of it all. Then all the applications that are used on the system are shown, both the developed as well as the one made available to the public by other authors. The applications can be divided in three groups: Remote GCS, Server and Raspberry Pi scripts.

6.2 System architecture

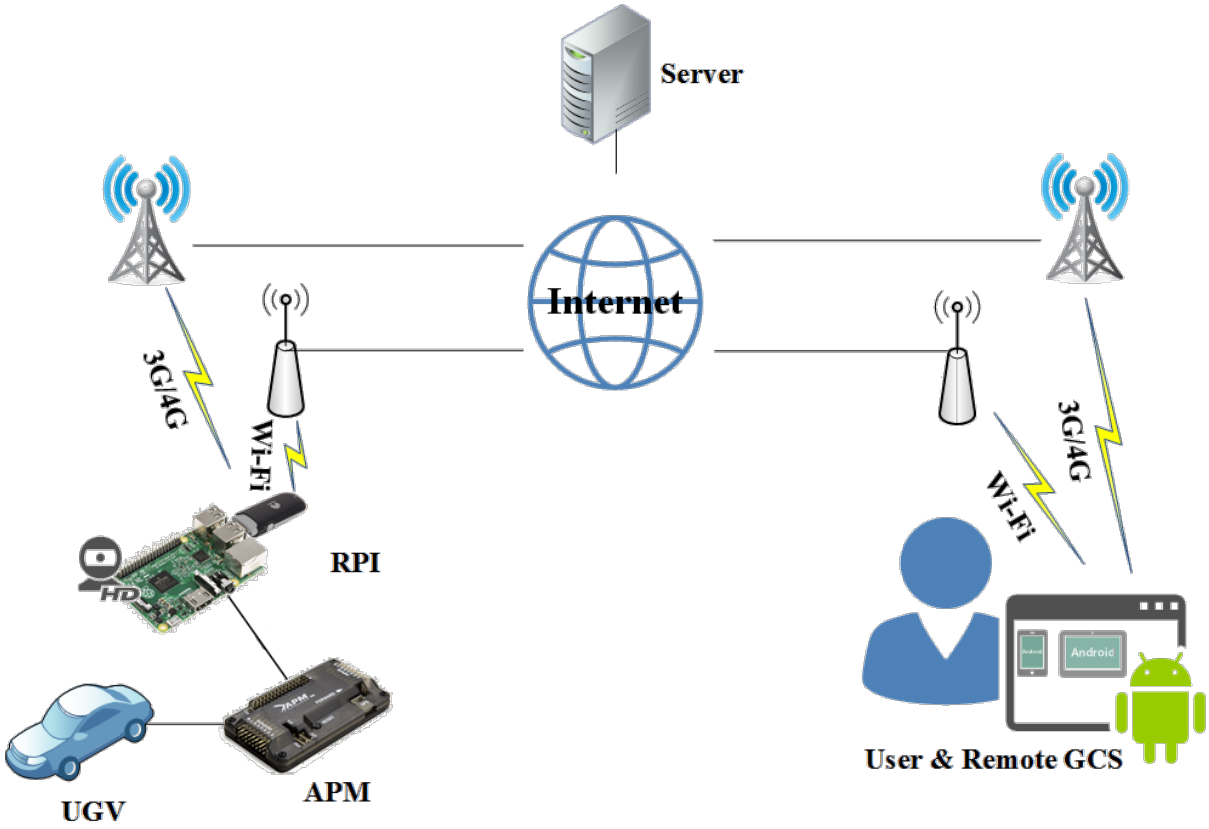


Figure 6.1 – Architecture of the whole system.

The proposed system, illustrated in **Figure 6.1**, allows a user to real-time control and monitor an UGV using heterogeneous wireless networks. Remote GCS Android application centralizes and displays all telemetry information from the vehicle with which is connected and allows the user to send command controls as well as watch the video captured onboard in real-time. The server can be divided into three different parts: routing server for data and commands, media server for video stream and server database where user credentials and run results are stored. The last part is the onboard electronic: APM and RPI. Here are used the ArduRover for APM and Sakis3G, video script, PiCamera and MAVProxy for RPI.

6.3 Remote GCS

The developed Android application was named “Remote GCS” because it was design to support all functionalities of a regular GCS plus the real-time operation, both commands and video. The functionalities list is composed by: real-time control (through joysticks or sensors), waypoint navigation, read and write APM parameters, adjust drive preferences (steering trim and steering angle), change connection settings and display final results of run in graphic form.

Remote GCS can run in any Android device, from smartphones to tablet. However, it is advised to use a tablet due to its dimensions that gives a better interface to the user as well as a quick access to all options. It is targeted to Android 4.4.2 version that corresponds to Application Programming Interface *(*API) 19, and further versions. The choice of Android is mainly because it is a Java-based programming language, ease of access to Integrated Development Environment (IDE2) and higher number of target users.

When application starts, a login screen is presented. The user can input its credentials if he already has a register. If not, he can simple click on register button and register itself. These two situations are presented on **Figure 6.2** and **Figure 6.3**.

In both cases, the database located on the server is acceded and some action is performed. To login, the database of users is simply consulted and checked if the input credentials exist and match to the right ones. For registration, the input credentials are written on users’ database and saved for a later login.

Figure 6.2 - Login screen of RemoteGCS.

Figure 6.4 – Register screen of Remote GCS.

After a successful login comes the real action. Englobed on a top view, driving mode screen appears by default; the other screen are mission planner and full parameters. It is possible to change the actual view selecting the desired one – tab navigation. **Figure 6.4** illustrates the driving mode tab without an active connection. Regardless the selected tab, there are common options to all: connect button (green button) and settings buttons (3 dots). Their names clearly indicates what they do.

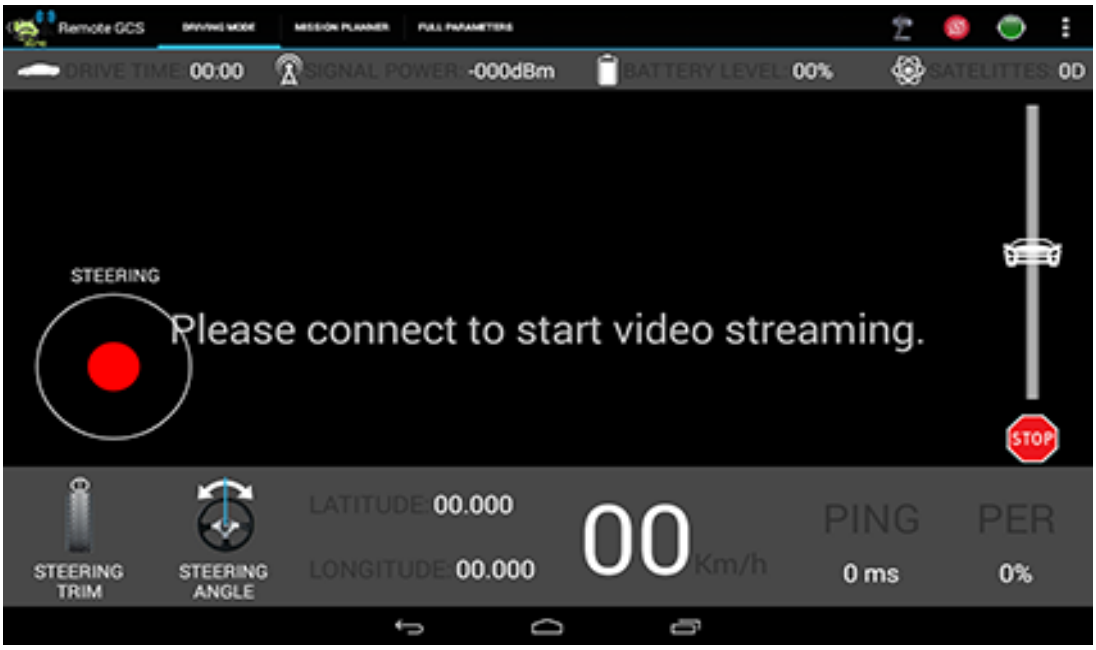


Figure 6.3 – Driving mode screen by joystick offline.

Driving mode tab allows to control the vehicle in real-time: when connected with the vehicle, the video is the background, seek bar on the right controls the speed and the UGV's direction of motion and the joystick controls the UGV's steering. However, is also possible to control the

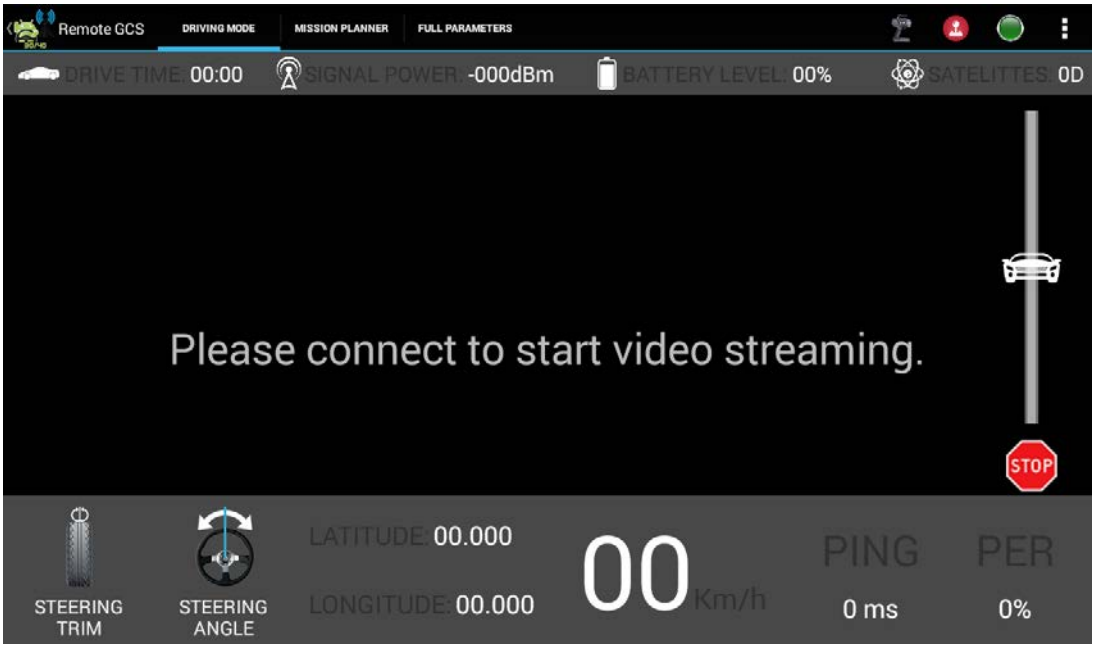


Figure 6.5 – Driving mode screen by sensors offline.

steering via sensors, in particular a gyroscope, as shown in **Figure 6.5**. To change between these two modes, just press the driving mode button (button on the left of connect button). All telemetry is displayed on this tab in order to take quick decisions about the vehicle's operation. It is also possible to adjust the vehicle's steering trim and steering angle, pressing the corresponding buttons. If the user wants to get back to the old RC command, just need to press the button with the RC command icon to allow it. The control mechanism is better explained on **Annex D**.

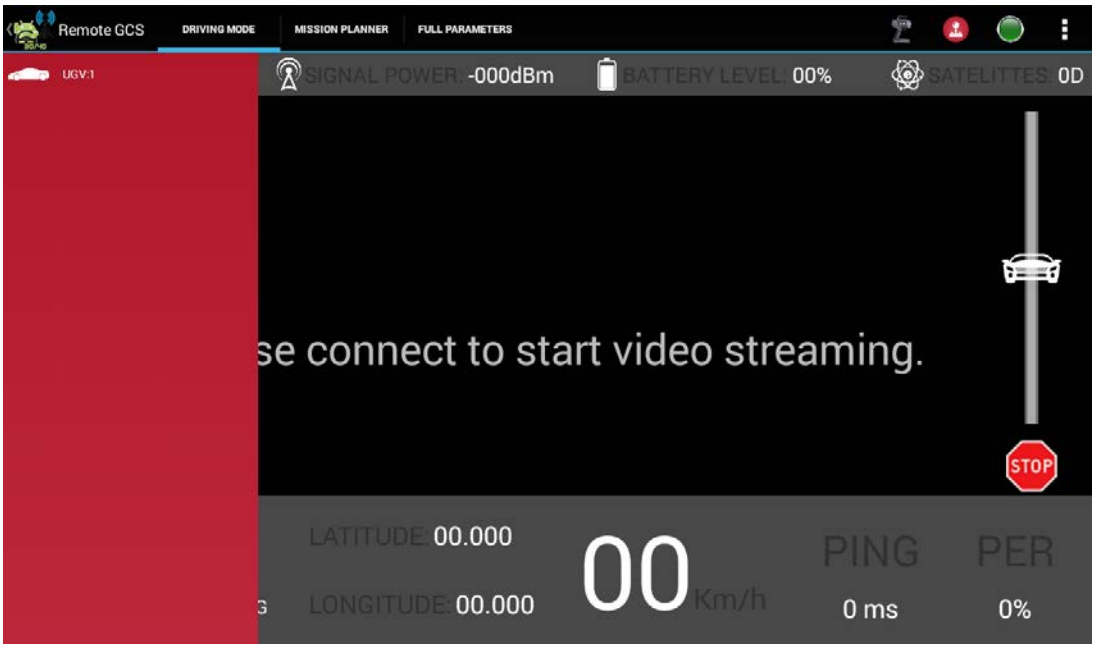


Figure 6.6 – Result list of UGVs after a connection attempt.

When the users attempts to connect to a vehicle, a list of available vehicles comes up, as shown by **Figure 6.6**. If no vehicles were found the list will be empty. When a vehicle is selected and if no errors occur, the connection between the two components is successfully done. **Annex E** contains a message diagram with an example of connect, telemetry, video and disconnect.

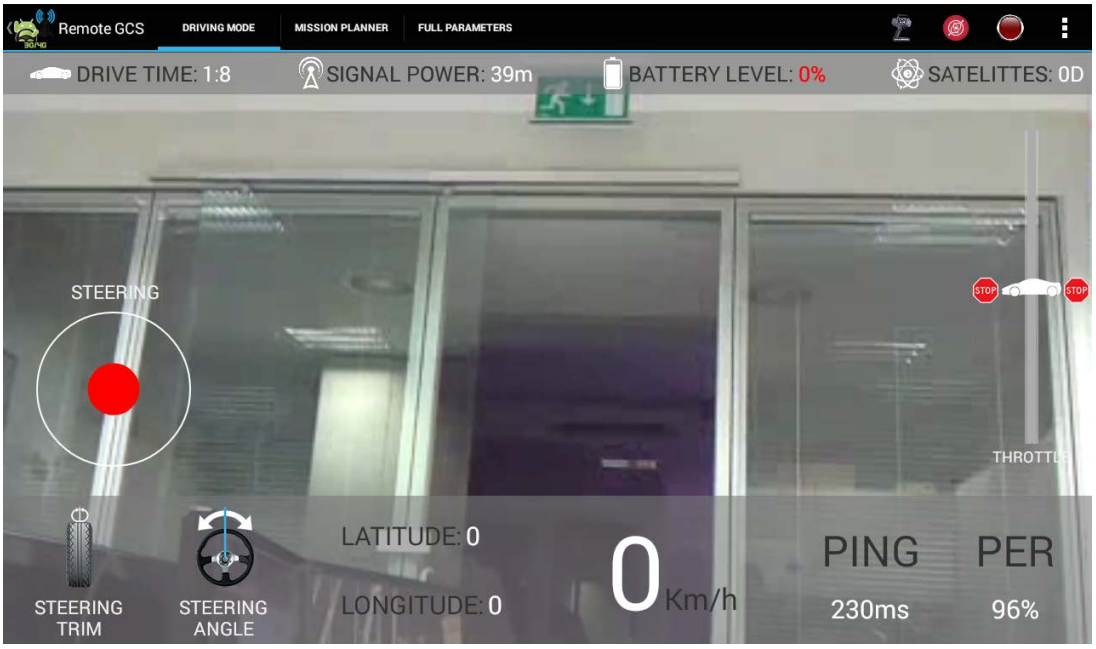


Figure 6.7 – Driving mode screen online.

Figure 6.7 illustrate the online scenario for driving mode tab. At this point is possible to send commands to the vehicle and watch live the captured video. **Figure 6.8** represent the same scenario, but this time for mission planner tab. No it is possible to mark waypoints on the map (long press on map), perform mission actions, edit waypoints and watch telemetry and live video.

Figure 6.8 – Mission planner tab with stablished connection and marked waypoints.

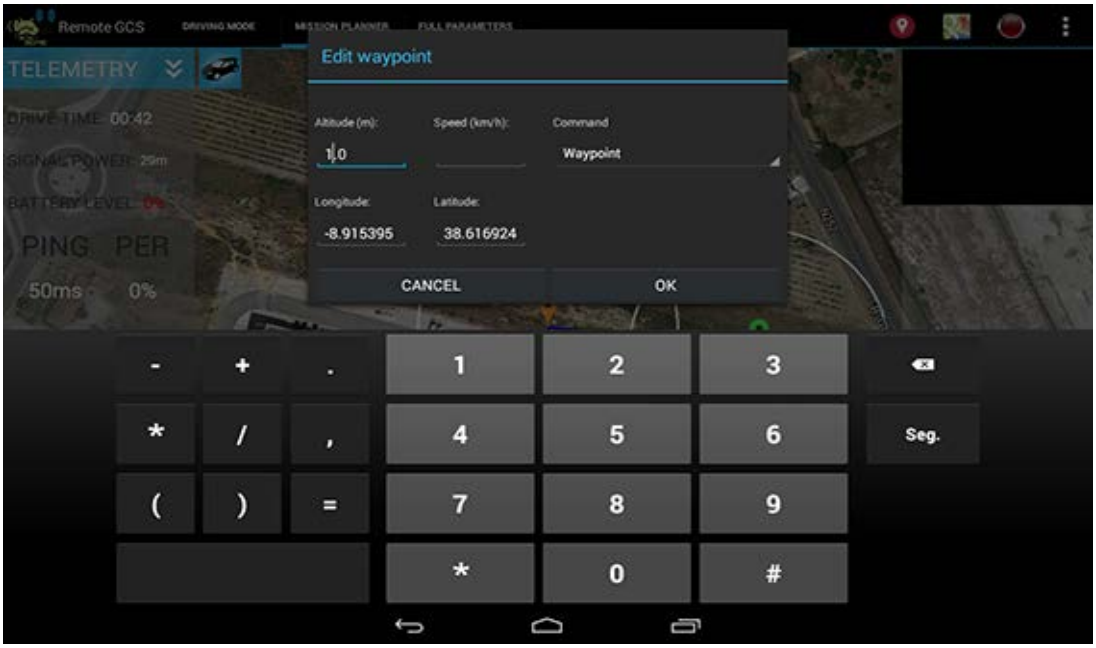


Figure 6.9 – Editing a waypoint.

When a waypoint is selected, is possible to edit its parameters: altitude, speed, command, latitude and longitude. To do that just click the middle blue button on bottom right of the screen. A pop-up window is displayed as shown by **Figure 6.9**. To delete a single waypoint, it just need to be selected and press the blue button with a cross. To delete all waypoints click on blue button with the trash bin. To move map's camera to the vehicle location press the blue button with a car icon.

Changing to Full Parameters tab, a couple of actions are possible to due, like read, edit and write parameters. After clicking the read parameters button, the result is similar to the presented by **Figure 6.10**.

Figure 6.10 – Parameters tab after read.

The settings menu allows to change some preferences, such as enable graphics at exit, specify the server IP address, server port and client port. The values that are altered are only updated when you press the save button. The settings scenario is illustrated by **Figure 6.11**.

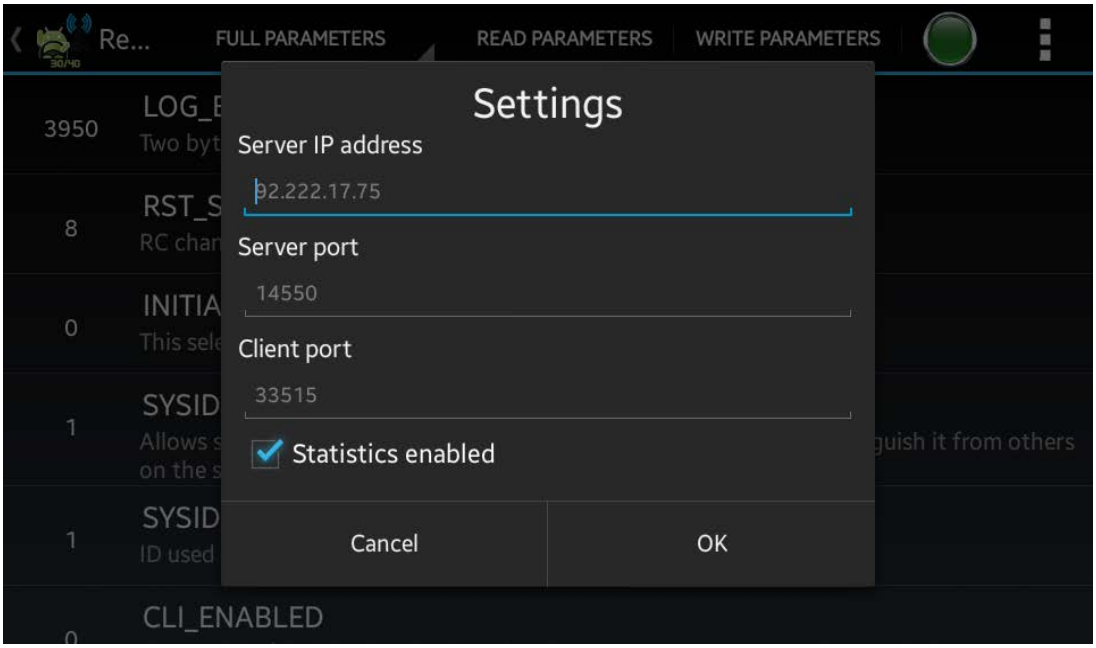


Figure 6.11 – Settings pop-up window.

When exiting the application after a successful connection and the preference for draw graphics with the run results is enable, comes a new window with the multiple results: delay vs speed, delay vs altitude, PER vs time, PER vs speed and battery vs time (see **Figure 6.12**). This window have a tab-navigation mode to change between the possible results.

Figure 6.12 – Graphical representation of collected data.

6.4 Server

Routing Server

As referred many times across the document, the routing server will allow the connection between Remote GCS and the vehicle. It was design for multiple-clients multiple-vehicles. This means that is possible to have multiple connections at same time. The routing server is a Java-based program which structure is represented by **Figure 6.13**.

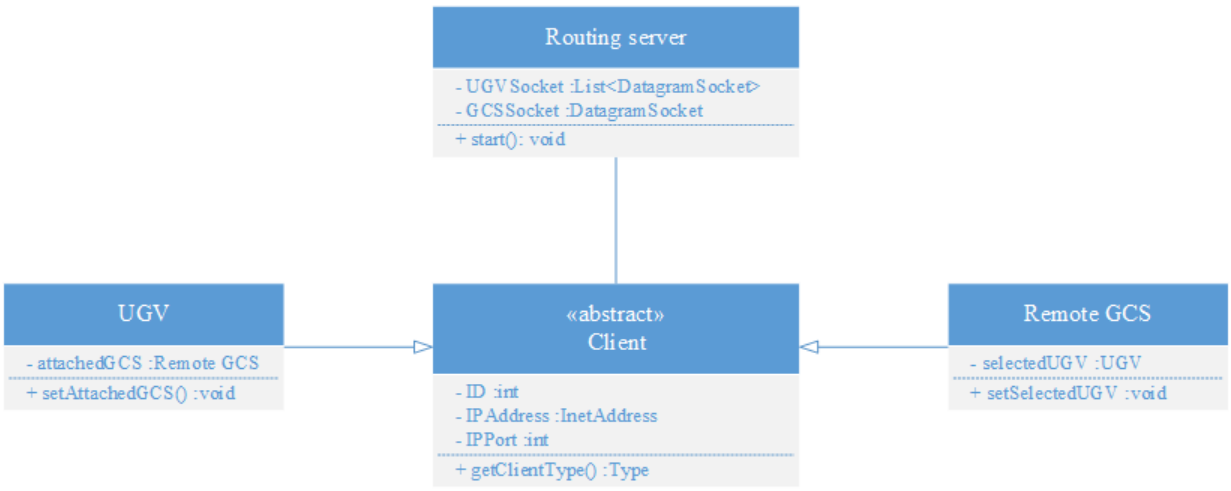


Figure 6.13 – Routing server class diagram.

The MAVLink is an application level protocol. Thus, there is need to implement a transport protocol. The most used is Transmission Control Protocol (TCP), which is connection-oriented protocol that implements message confirmation (or ACK) [8][159]. Other well-known is User Datagram Protocol (UDP), a connectionless protocol used for simpler message transmission [8][159]. These two protocols are very popular and the application type that will decide the best protocol to be used for transmission. In this case, TCP will cause some problems due to the fact that it only is possible to send other message when the previous is confirmed, which would lead to delays. Bearing in mind that wireless communication are used (quite susceptible to errors), many retransmissions would be done, aggravating the fact cited above. Thus, the solution is to use UDP, where the message flow without restrictions and there are not confirmation and so no additional delays. However, some messages need a special attention and really need to have confirmation: throttle, steering, connect, disconnect and select UGV. *E.g.* if the vehicle is running and a stop message is sent and is not received an accident can occur. So, the ideal scenario would be to have a hybrid solution where only those messages are confirmed. That is the implemented solution. To confirm those messages, the Remote GCS initializes a thread when a command is sent. It sends the command to the vehicle and periodically compares the sent value with the actual value: if they are different, it sends the message again to ensure that

the desired value is reached; if they are equal, the thread stops and confirms the message. These mechanism is represented on **Figure 6.14** where “command” refers to an action which will lead to the sending of said messages. On **Annex E** is shown a message flow diagram of a connection, select UGV, telemetry, command and disconnect.

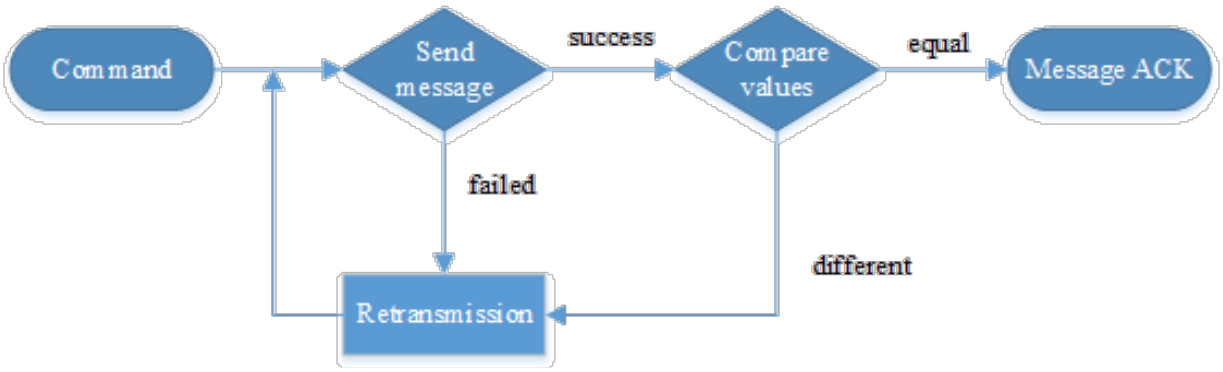


Figure 6.14 – Diagram of UDP messages with confirmation.

Media Server

To route the video stream from RPI to Remote GCS, WSE is used and is running on the same physical server where routing server is implemented. WSE is a robust and customizable media server software that provides a high-quality and reliable video and audio streaming to any device, anywhere [160]. Versatility is other of its features because WSE offers the aforementioned service regardless where the software is developed. This software avoids all complexity of video streaming (*e.g.* multiple protocols, video formats and other technical challenges), offering a simple stream [161].WSE transcodes any video and delivers it in multiple formats with the best possible quality. WSE also offers multiple levels of content security to ensure that streams, video, network and audio assets are protected [161].

6.5 Raspberry Pi scripts

Video stream

This script named “video stream” was developed to start the video capture and stream as soon as the RPI is powered up. It also allows to control some video parameters, such as Frames Per Second (FPS), color (black and white or full colors), orientation, rotation, *etc*. Thus the video quality can be adapted to the communication conditions. By default the video is captured in full colors with 20 FPS and further encoded in Fast Forward Moving Pictures Expert Group (FFMPEG). To transmit the video from the RPI to server and from server to Remote GCS, the

Real Time Messaging Protocol (RTMP) and Real Time Streaming Protocol (RTSP) protocols, respectively (see **Figure 6.15)**. The specification of these protocols can be found at [162] and [163], respectively. As referred before, PiCamera Python library is used to provide the camera capabilities to the RPI. This library was developed by RPI Foundation and is the official library for RPI cameras.

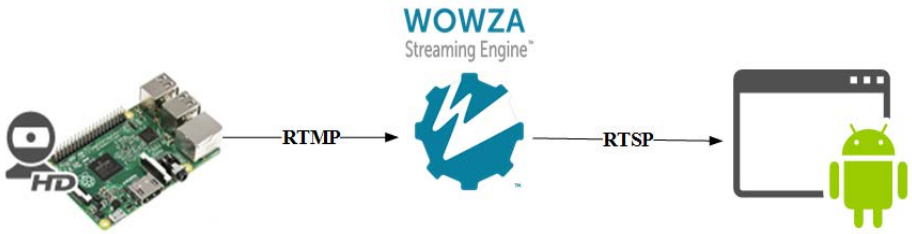


Figure 6.15 – Video transmission protocols across the system.

MAVProxy

MAVProxy is intended to be a complete, portable and fully-functional GCS for UAVs [164]. However, it can also be used for UGVs and other vehicles that support MAVLink protocol (like APM). This GCS can run on Linux, OS X, Windows and others – portability. Its light-weight design allows it to run on small computers with ease, such as RPI. MAVProxy is written in Python and is open source [165]. Although the MAVProxy be a GCS in this context is only used to make the route of MAVLink messages between the Remote GCS and APM.

Sakis3G

Sakis3G is a script developed to provide Internet connection through 3G modems on Linux and Linux-based OS [166]. These modems can be attached via Bluetooth or USB. The two aforementioned facts makes possible that RPI to have Internet access through a mobile communication modem. Sakis3G offers 2 possibilities to stablish a connection: Through a command line or via a graphical user interface. All procedures to configure the RPI and the modem can be found at **http://www.sakis3g.com** [166] or **http://raspberry-at-home.com/installing-3g-modem/#more-138** [167].

This page intentionally left in blank

Chapter 7
RESULTS AND EVALUATION

It is time to analyze the system reliability and viability. Key parameters were collected during specific test-fields and scenarios and they are now processed and analyzed.

7.1 Introduction

In order to evaluate the system performance and viability, multiple fields test were performed in various scenarios. **Table 7.1** resumes the field tests, scenarios and other relevant factors.

Table 7.1 – Field tests and respective scenarios.

Scenario		Test type	Weather conditions	Wind
INDOOR		10min free-run;	23º C	—
OUTDOOR	Rural	10min free-run	23º C; sunny intervals.	Weak (10km/h max)
		40m at speed1		
		40m at speed2		
		40m at speed3		
	Suburban	10min free-run	27ºC; sunny	Weak (10km/h max)
		40m at speed1		
		40m at speed2		
		40m at speed3		
	Urban	10min free-run	26ºC; sunny	Weak (10km/h max)
		40m at speed1		
		40m at speed2		
		40m at speed3		

One indoor test was performed, Wi-Fi connections were used for Remote GCS and UGV. For outdoor only 4G communications were used. In each race data were obtained for assessing the performance of the system: speed, GPS location, Round Trip Time (RTT) of a ping message, PER, $Bitrate_{VIDEO}$, $Bitrate_{IN}$ and $Bitrate_{OUT}$ from GCS and vehicle's received wireless strength (P_{RX}), both cellular and Wi-Fi. When the vehicle is connected to a mobile networks is also possible to collect information about technology in use, Cell Identifier (ID), Mobile Network Code (MNC) and Location Area Code (LAC) (see **Annex F** for more details about it). The parameters wireless strength, Cell ID, MNC and LAC were collected using a developed Android application. The ping is a message that travels to the whole system, from the Remote GCS to UGV and comes back. The time that the message take to do that trip (RTT) is measured, processed and its Cumulative Distribution Function (CDF) analyzed, as well as its Probability Distribution Function (PDF). The $Bitrate_{IN}$ and $Bitrate_{OUT}$ parameters refer to the transmission rates into and exit from Remote GCS, respectively.

On outdoor scenarios, the device that is running Remote GCS application is connected to MEO mobile operator; by its turn, the vehicle is connected to the internet through Vodafone network. For indoor scenario, the router was transmitting with a power of 40mW ≈ 16.02dBm, using IEEE 802.11b standard.

7.2 Field-test 1: 10 minutes free-run

The results of field-test 1 are presented from figure 7.1 to figure 7.8: speed, RTT, PER, P_{RX}, cell number, $Bitrate_{OUT}$, $Bitrate_{IN}$ and $Bitrate_{VIDEO,}$ respectively. Note that all these results are analyzed as a function of running time. After the analysis and comments of those results, come figure 7.9 that presents the CDF of RTT and their comments.

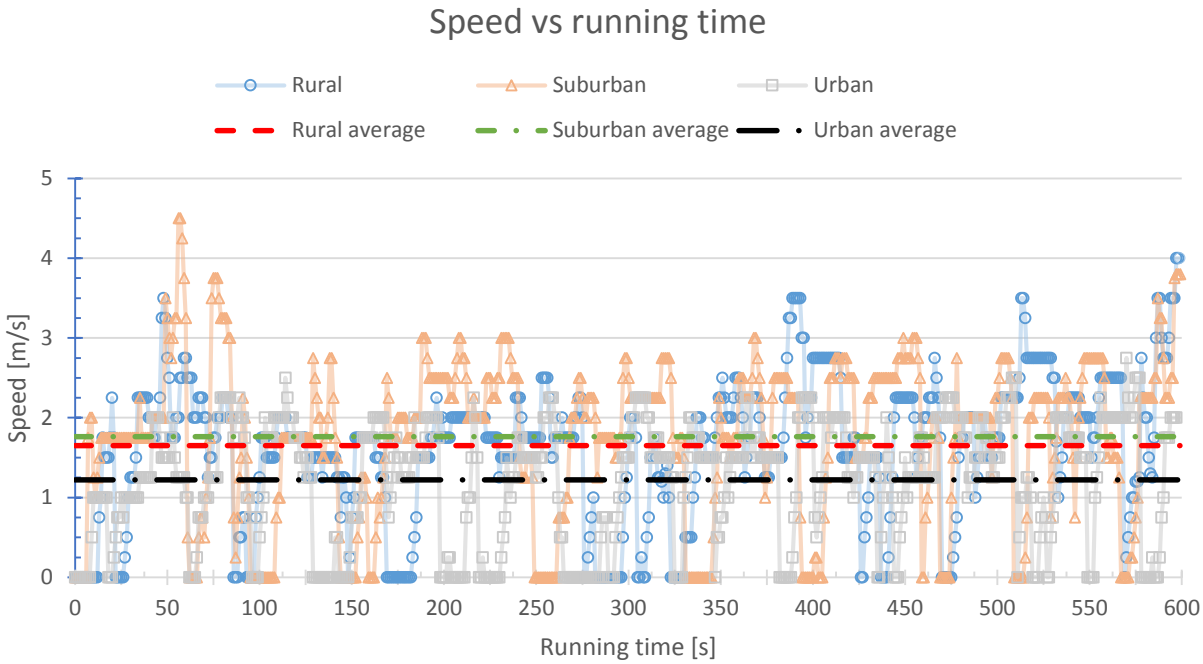


Figure 7.1 – Speed vs running time graph.

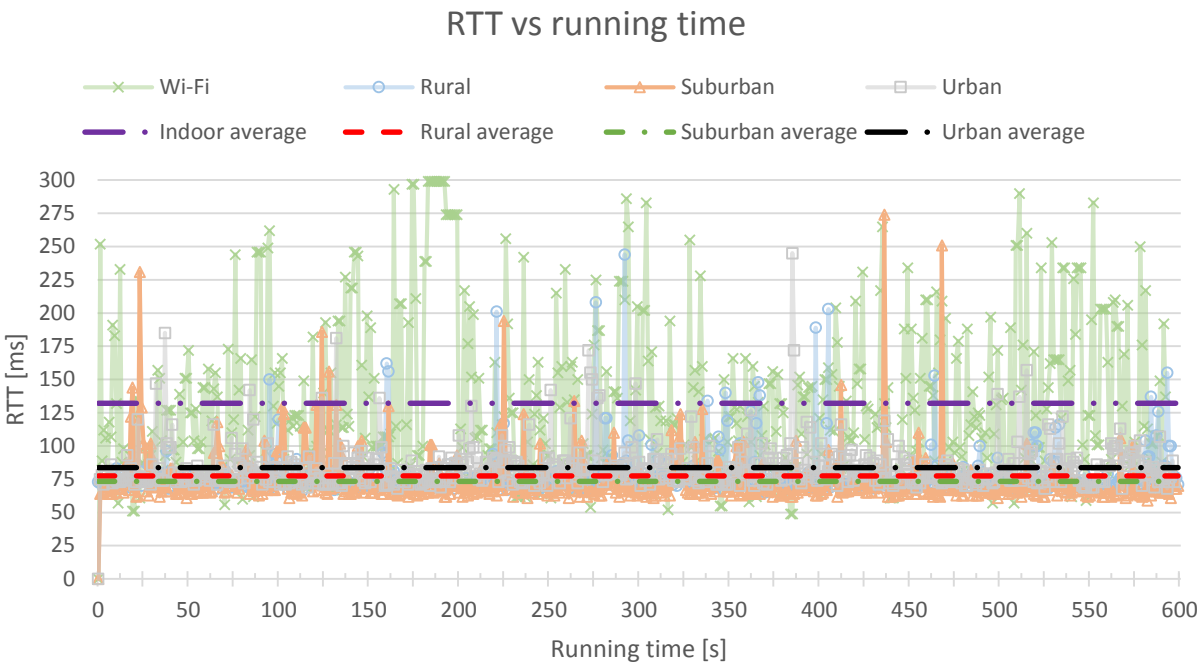


Figure 7.2 – RTT vs running time graph.

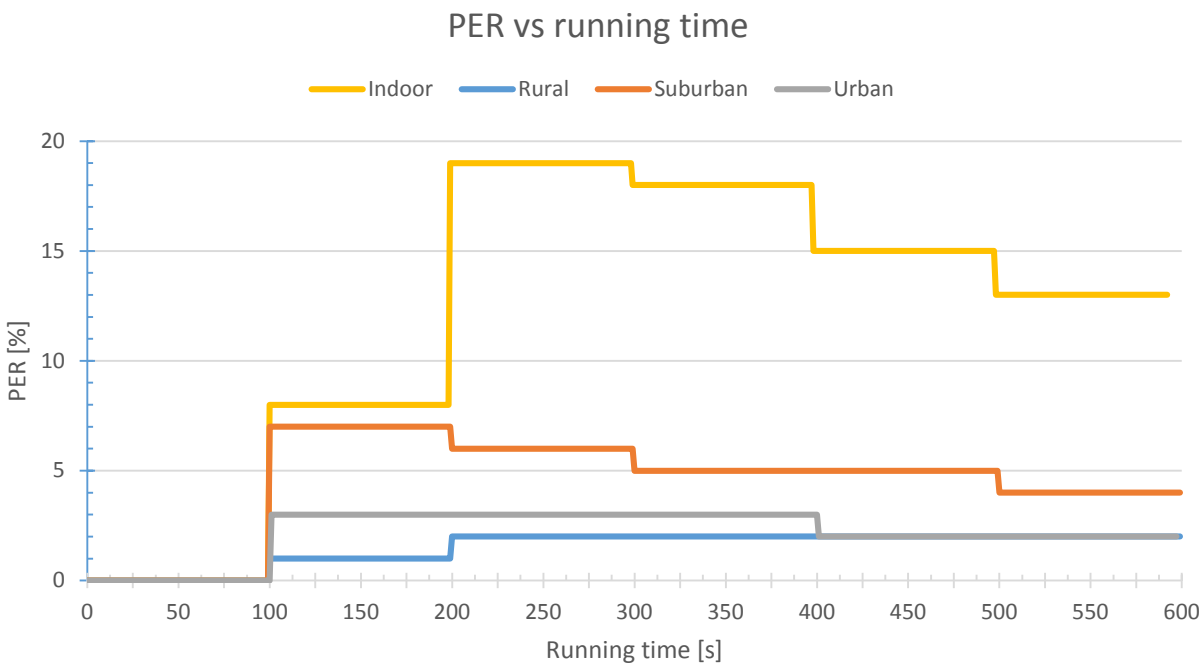


Figure 7.4 – PER vs running time graph.

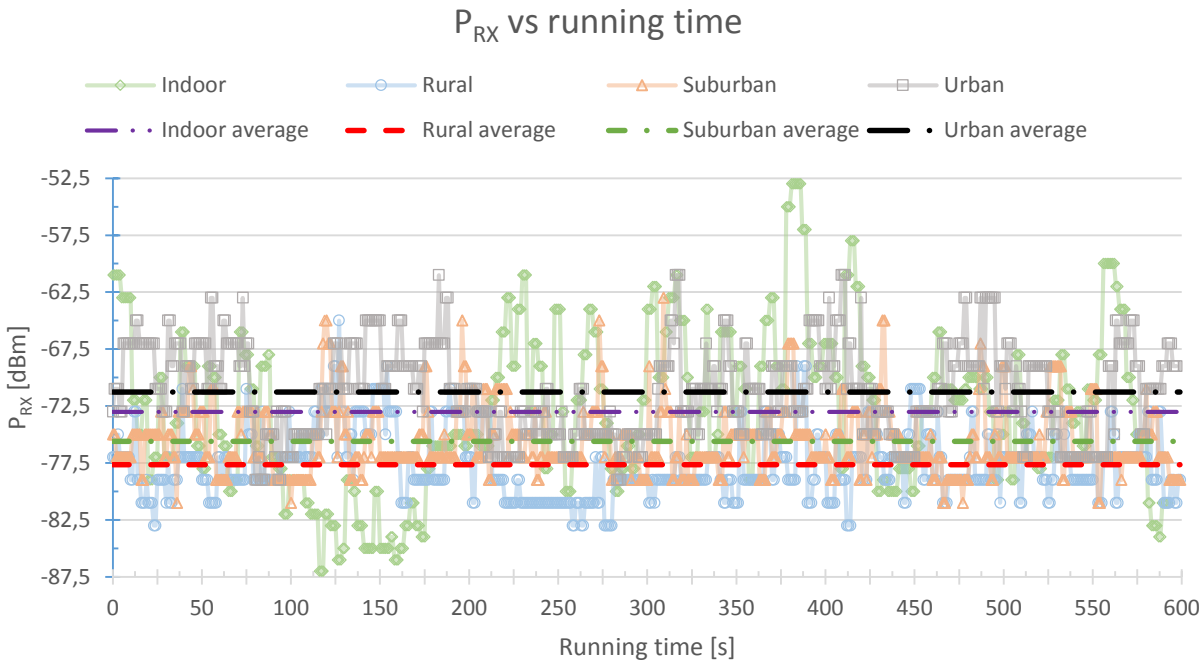


Figure 7.3 – P_{RX} vs running time graph.

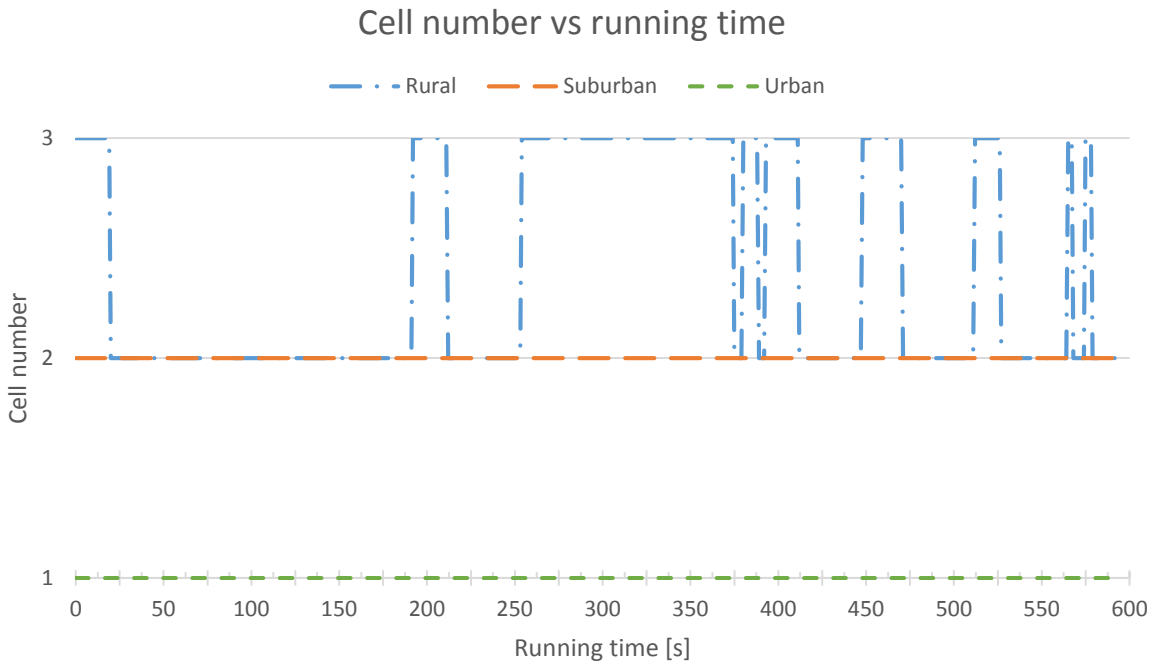


Figure 7.6 – Cell number vs running time graph.

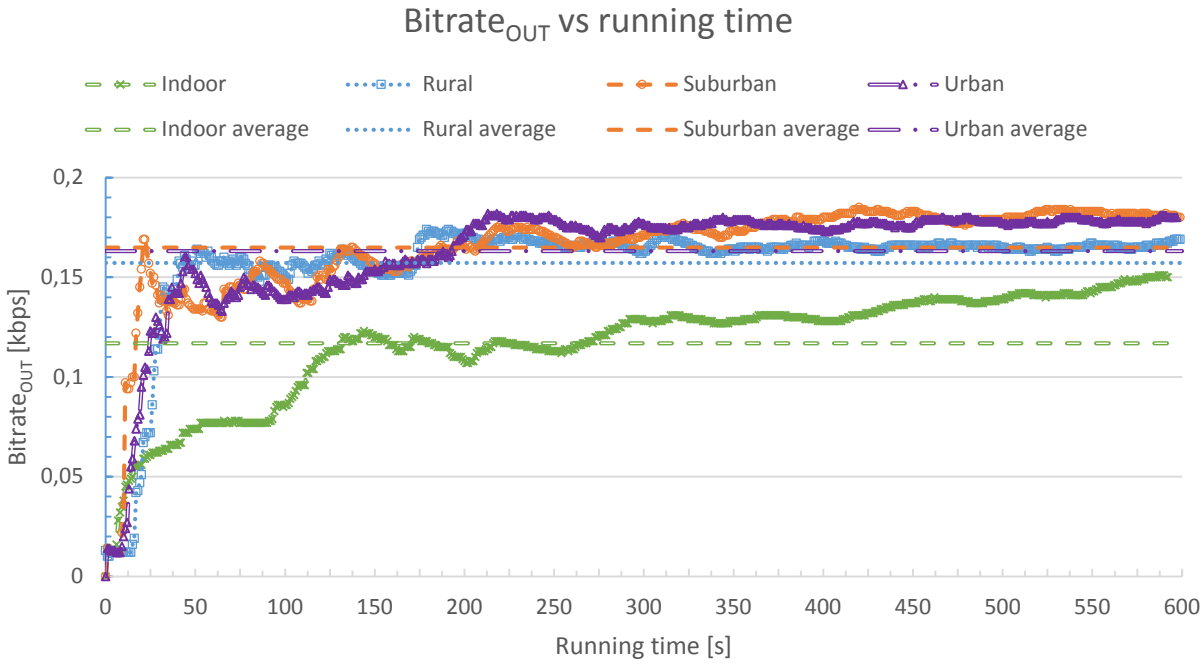


Figure 7.5 – Bitrate$_{OUT}$ vs running time graph.

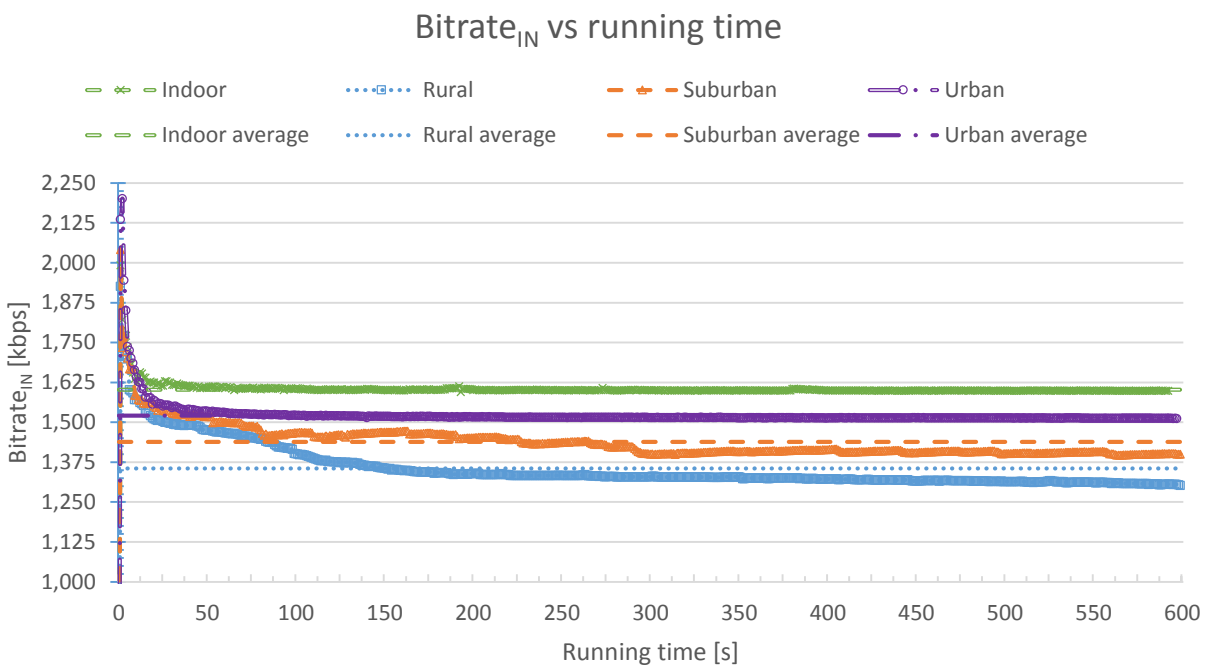


Figure 7.8 – Bitrate$_{IN}$ variation over running time.

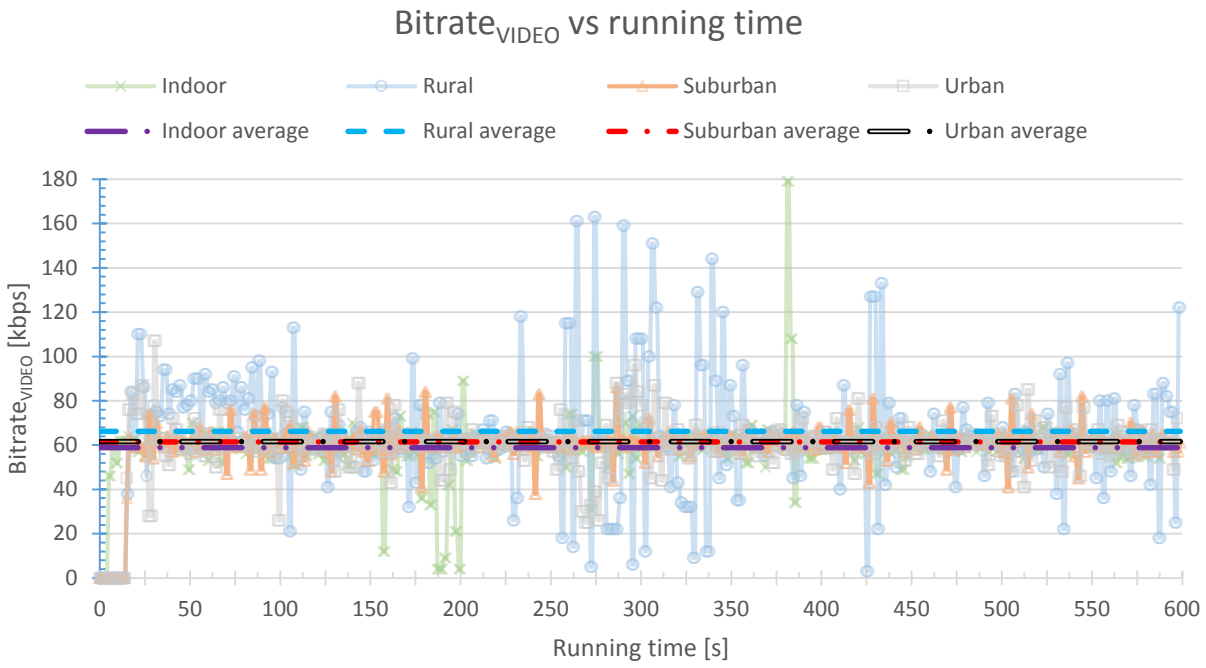


Figure 7.7 – Bitrate$_{VIDEO}$ evolution over running time.

For indoor scenario, the results of RTT are consistent with the P_{RX} values (inversely proportional). From all scenarios, this is the worst one on the results of RTT. As referred before, a 16dBm signal is emitted from AP. By observation of **Figure 7.3**, is possible to see that the received power of indoor scenario reaches minimum values of -87.5dBm (attenuation of 103.5dBm). Those facts can be explained with the positioning of the Access Point (AP) (under a table) and the presence of large obstacles that hinder the signal propagation. Also the channel

quality and the number of active users are contributing factors. The PER values are consistent with RTT results, *i.e.* higher delay zones is where are more error packets (first half time – 300 seconds) and it decreases with time. However, the PER values are worrying – values between 19 and 8%. For the bitrate of the video it has an average value of 58.79kbps. The variations are consistent with the received power values: the higher $P_{RX,}$ the higher the bitrate. *E.g.* in the time interval between 150 and 200 seconds the P_{RX} decreases as well as the $Bitrate_{VIDEO}$; the maximum peak of $Bitrate_{VIDEO}$ occurs on the maximum peak of P_{RX}, around 390 seconds.

The rural scenario is the most interesting of the four. That fact is due to the transition between two different cells – handover (see **Figure 7.6**). Observing figures 7.4 and 7.5, when a handover occurs, the P_{RX} values increase; when the UE is near to the periphery of the cell, the P_{RX} decreases. This scenario is one where there are less stops, the average speed is 1.65m/s (see **Figure 7.1**). By observation of **Figure 7.4**, the PER is quite low which supports the idea that the handovers occurred without errors (there were no communication failures). These values also occur due to composition rural setting: little or no obstacles large constraints that cause a communication line of sight, by the UE. Crossing the information of figures 7.1, 7.2 and 7.3, the results are consistent with each other: higher speed, lower the P_{RX}, the greater the P_{RX}, the smaller the RTT and the inverse of both is also true. Observing **Figure 7.7**, is possible to see many variations of $Bitrate_{VIDEO}$. That can be explained with the handovers and interference from neighboring cells that harm the numerical results.

On suburban scenario is quite noticeable the influence of speed in the received power values. By observation of figures 7.1 and 7.4, the minimum peaks of P_{RX} are consistent with the maximum peaks of speed. Consequently, it is there that are the maximum peaks of RTT. AS there are no handovers in this case, the obstacles and distance to the Base Transceiver Station (BTS) can be influent factors in these results. Once more, the $Bitrate_{VIDEO}$ values are dependent on P_{RX} values: observing **Figure 7.7**, the $Bitrate_{VIDEO}$ has an average of 61.45kbps. Over time, many variations occur around this value. Those variations are consistent with P_{RX} values (see **Figure 7.3**). The PER decreases over the time due to the improvement of the channel conditions: the P_{RX} have more ad higher peaks over average value and less under. However, a PER of 7% has to be taken into account because it is already a significant amount in the operation of a real-time system.

The urban scenario presents the lower speed values, as expected and a speed average of 1.22m/s. It was also the scenario that register more stops, to avoid collisions and due to the limited space

compared to the other outdoor scenarios. Observing **Figure 7.2**, is possible to see that the RTT average is around 83.7ms and that there is a maximum peak of 245ms at time 385 seconds. Crossing this information with the **Figure 7.3**, that phenomenon occurs on a minimum peak of P_{RX}. The P_{RX} average is around -71.26dBm, registering the best value of outdoor scenarios. The values of $Bitrate_{VIDEO}$ are consistent with P_{RX} values, once more. PER results are quite low and therefore this is not a worrying factor. At last, the $Bitrate_{VIDEO}$ is consistent with P_{RX} – directly proportional (see figures 7.4 and 7.8).

At last but not less important, the $Bitrate_{IN}$ and $Bitrate_{OUT}$. These 2 parameters were left for last because there are no major conclusions to be drawn from them. The average values for outdoor scenarios are quite similar (see figures 7.6 and 7.7). Suburban scenario register the higher average value of $Bitrate_{OUT}$ (0.1632 kbps), followed by urban and rural. The indoor scenario is the worst, registering an average of 0.1169kbps. For $Bitrate_{IN}$, indoor scenario registers the best of the results with an average of 1.6kbps, followed by urban, suburban and rural.

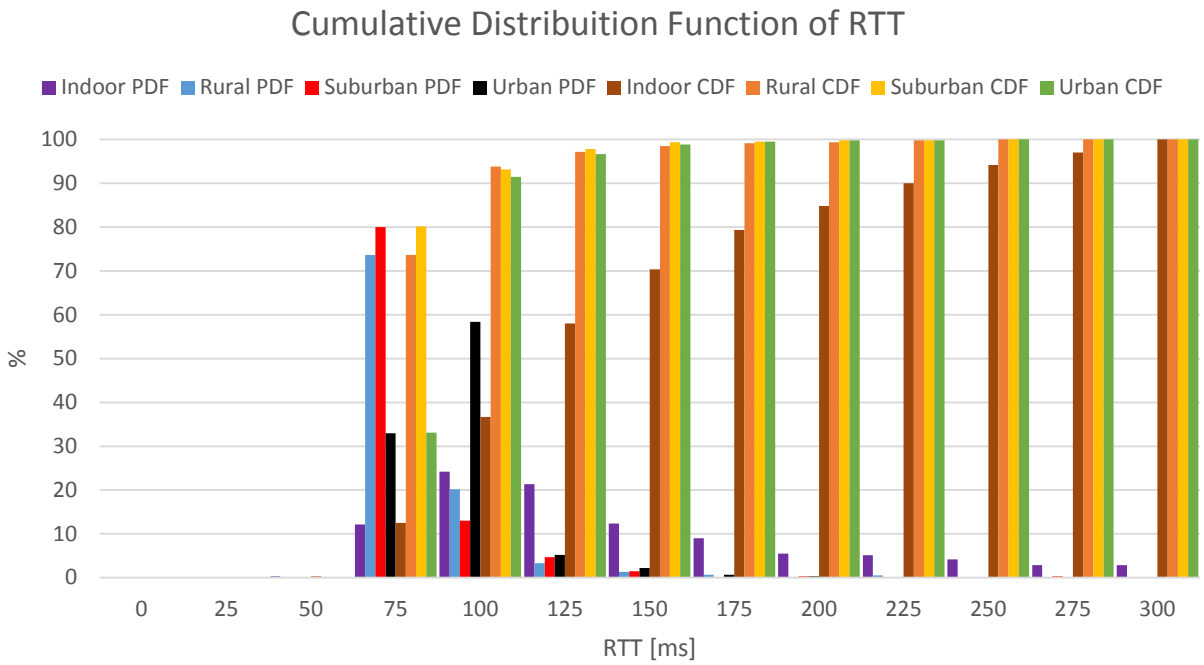


Figure 7.9 – Cumulative Distribution Function of RTT.

Observing **Figure 7.9** is possible to see that for outdoor scenarios 95% of occurrences happens for RTT ≤ 125ms: for rural and suburban scenarios the majority of occurrences happens for RTT ≤ 75ms; for urban scenario, the majority happens for RTT ≤ 100ms. Studying the indoor case, it register occurrences for every case, being the majority for RTT ≤ 100ms. However the results are not satisfactory, since there are significant percentages of occurrences for high RTT values, which can compromise the system.

7.3 Field-test 2: 40 meters

This section follows the same philosophy of the previous. The results are present from figures 7.10 to 7.16. Speed1 corresponds to 1.75m/s, speed2 to 2.25m/s and speed3 to 3.5m/s.

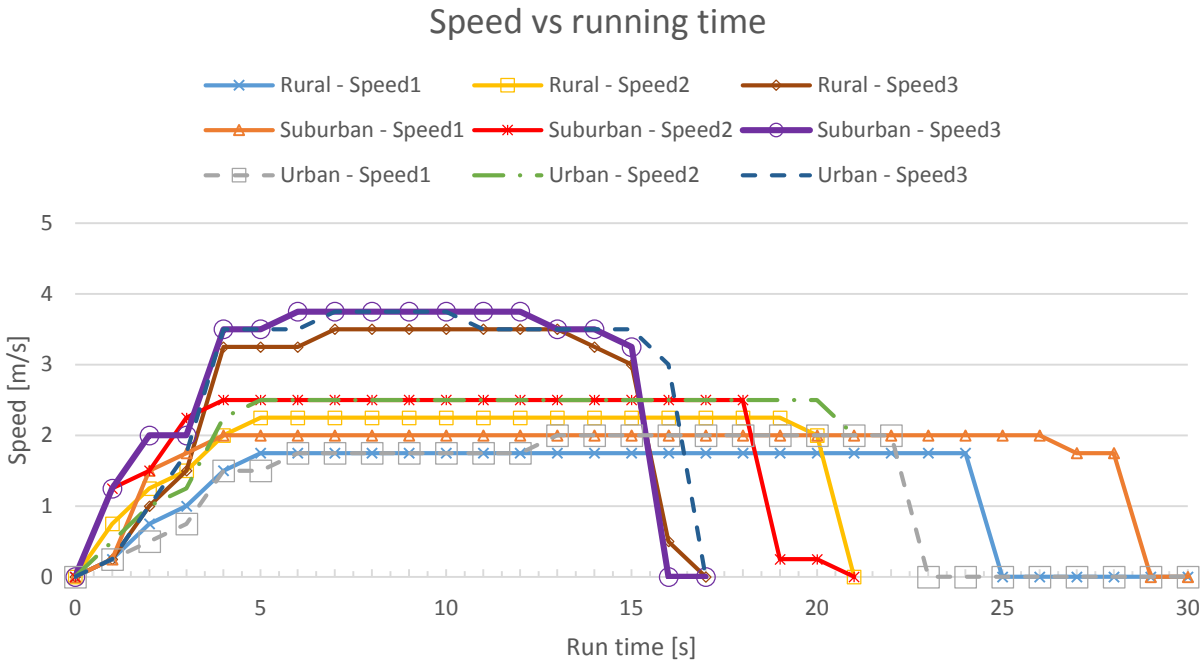


Figure 7.10 – Speed evolution over running time.

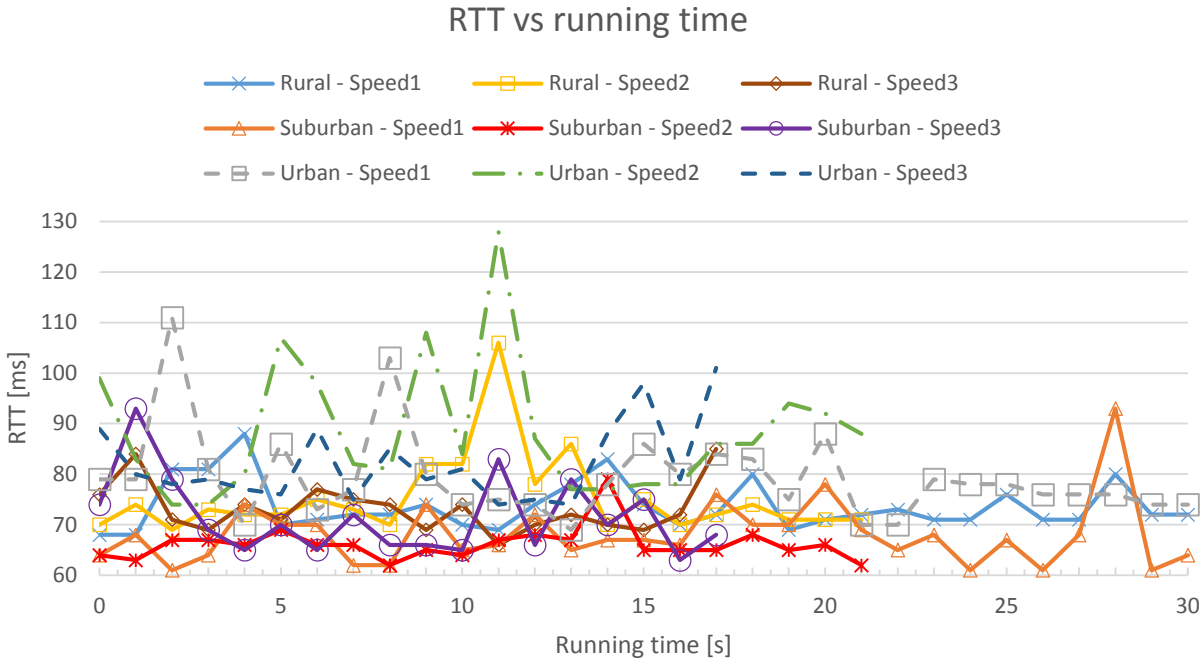


Figure 7.11 – RTT evolution over running time.

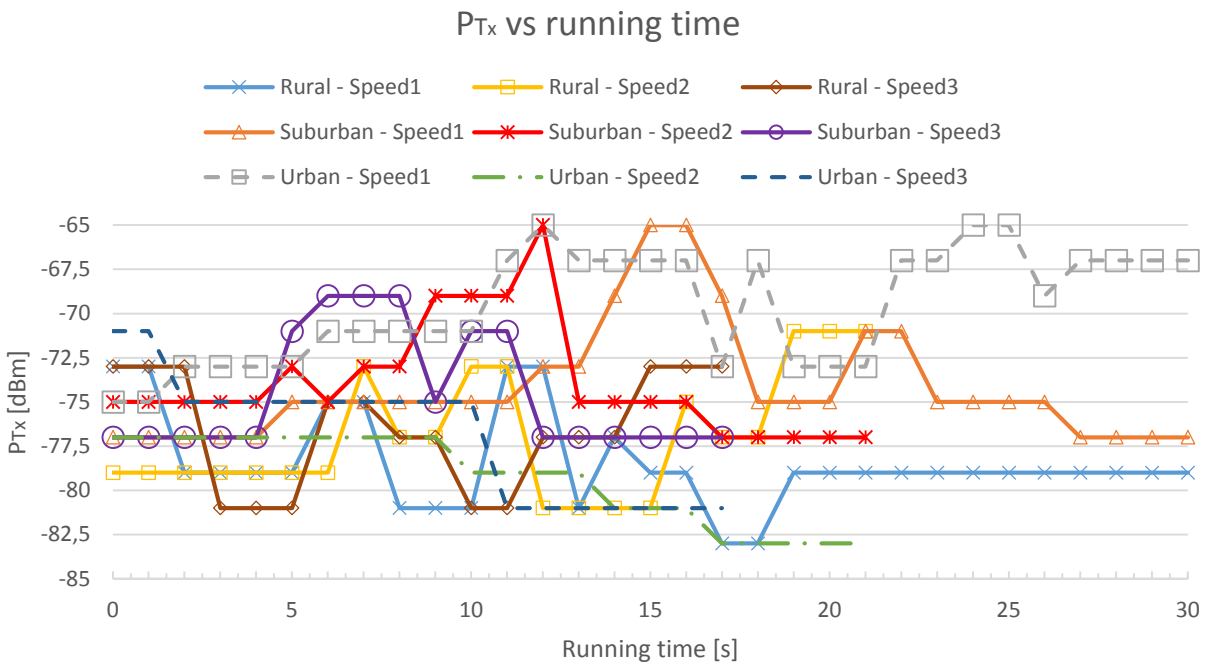


Figure 7.13 – P_{RX} variation over running time.

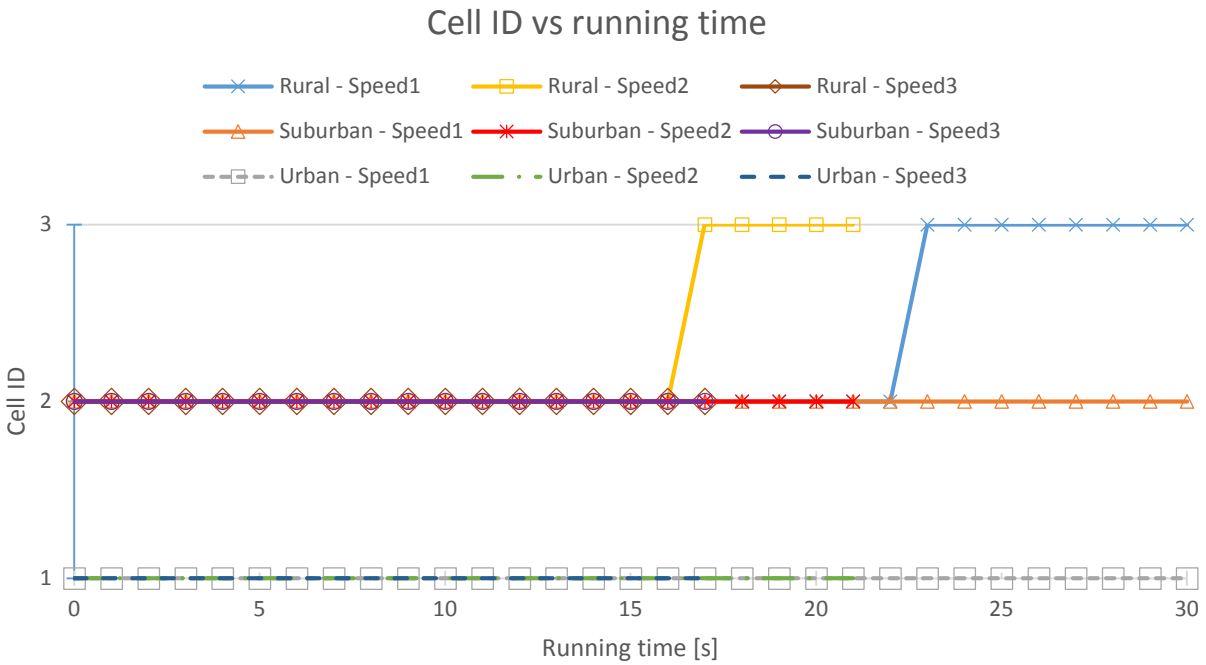


Figure 7.12 – Cell number variations over running time.

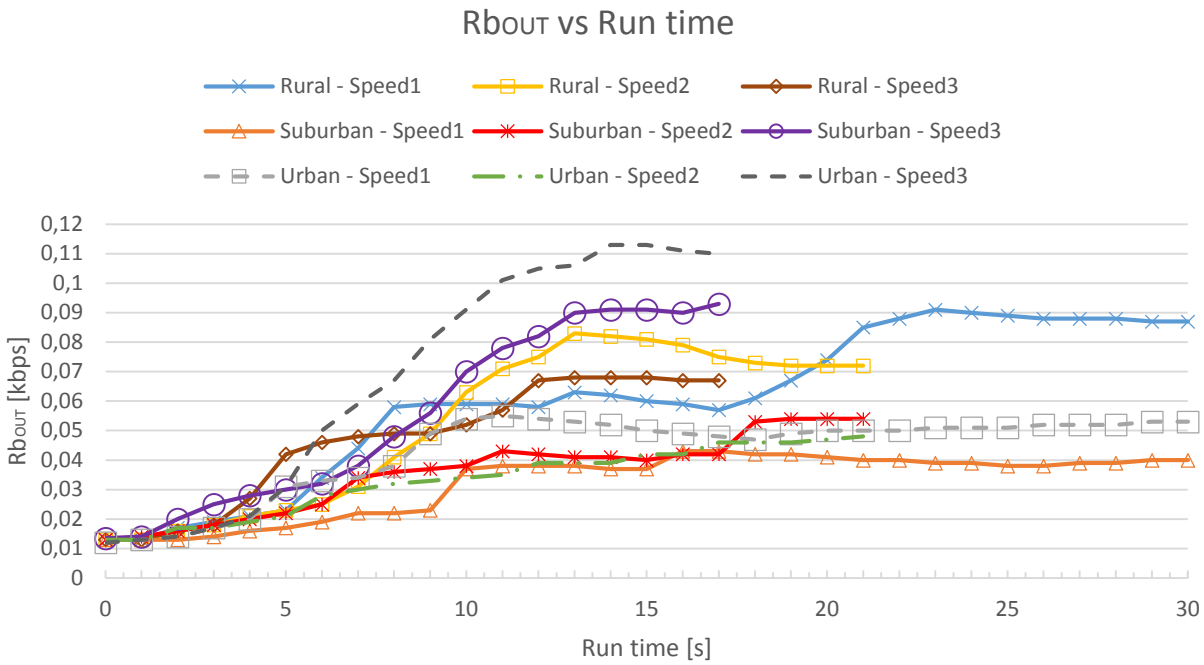


Figure 7.14 – Bitrate$_{OUT}$ vs running time graph.

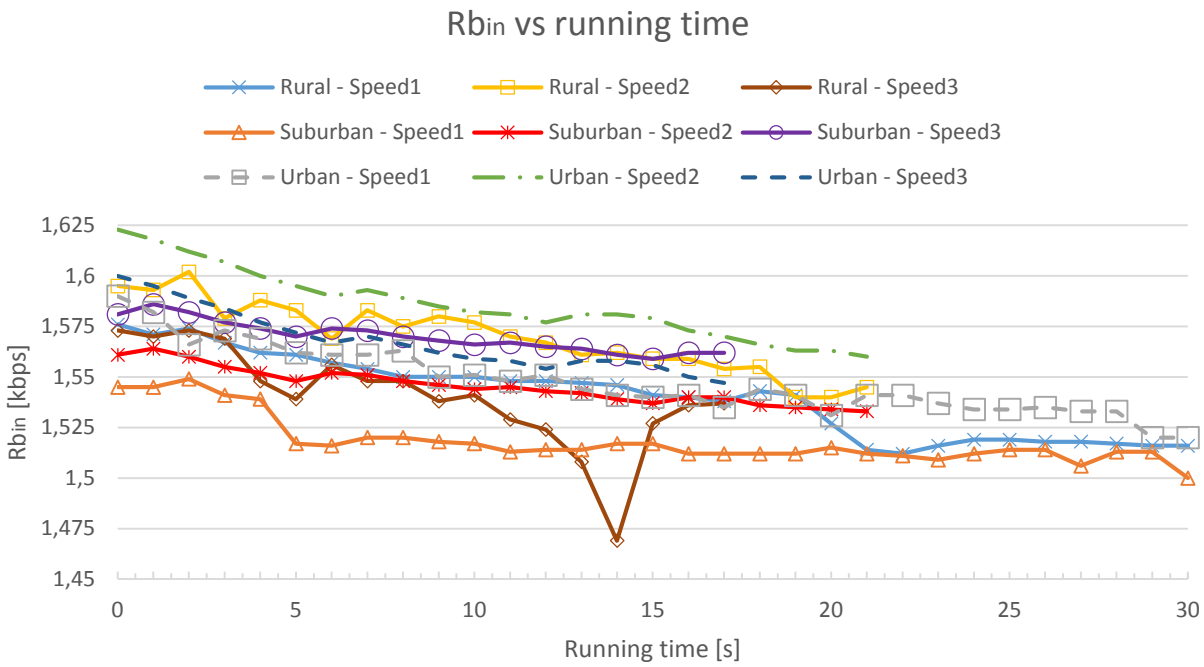


Figure 7.15 – Bitrate$_{IN}$ evolution over running time.

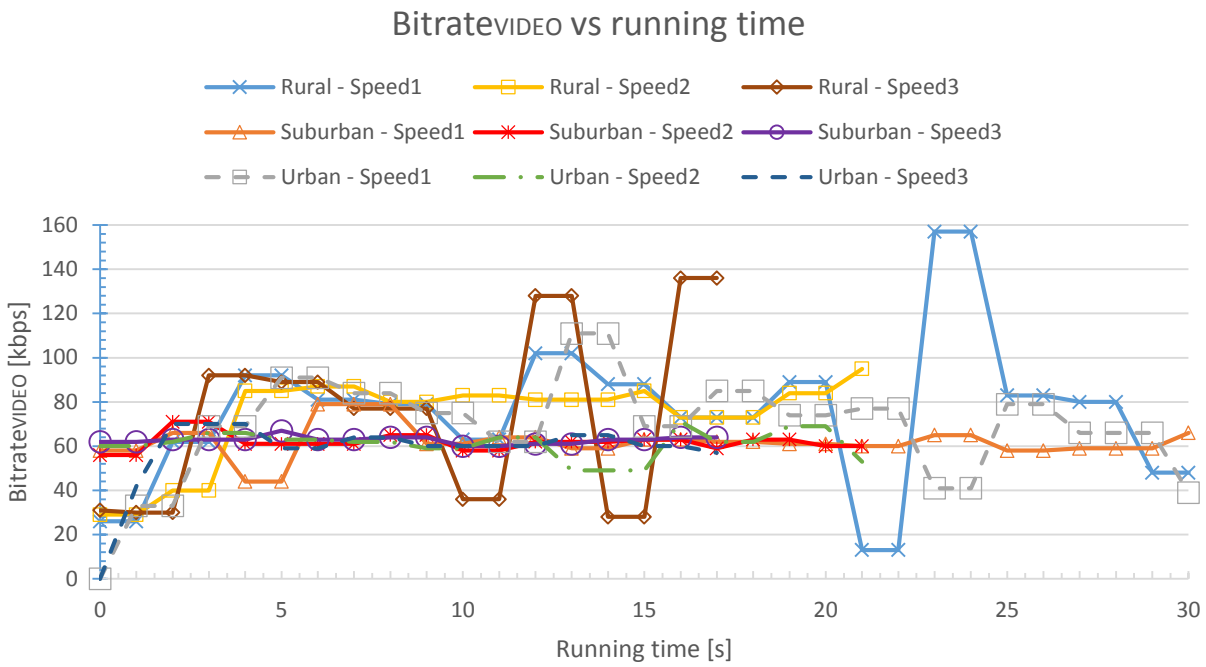


Figure 7.16 – $Bitrate_{VIDEO}$ variations over running time.

The main goal of these tests is to evaluate the influence of speed on all the other parameters. First, comparing the speed results with the data on **Figure 7.11**, is possible to observe in most cases that when the vehicle accelerates the RTT increases and vice-versa. On urban scenario situations it is also observable that the RTT results are very instable. An explanation could be the scenario itself, where the presence of high buildings and constructions affect the signal propagation. When the speed is constant the P_{RX} has small variations (**Figure 7.12**). These variations are greater the higher the speed.

Crossing the information of **Figure 7.10** with **Figure 7.14** and **Figure 7.15**, is possible to conclude that speed has no influence on $Bitrate_{IN}$ and $Bitrate_{OUT}$. In terms of cell switch, the results are more visible on $Bitrate_{VIDEO}$: the approach to the limit of the cell is associated with lower $Bitrate_{VIDEO}$; when the cell transition, this increases again (**Figure 7.16**).

Chapter 8
CONCLUSIONS AND ROAD AHEAD

What can be concluded with this project? There is anything that can be done to improve the system’s functioning?

8.1 Introduction

The scope of the developed work was to develop and implement a UGS, fully capable of control and monitor a UGV in real-time using heterogeneous wireless networks.

Within the aforementioned context, a mobile application was developed – Remote GCS – as well as all the electronic components assemble on the vehicle that provide its operation. Besides that, a Server application was also developed which acts like a routing server, both for data and video. A hybrid implementation of UDP was implemented on Remote GCS in order to provide the ACK of a set of special messages: throttle, steering, connect, disconnect and select UGV.

After all the developed work and respective results, many conclusions can be taken:

- RemoteGCS is a very functional, complete and user-friendly application. Its use is very intuitive and the vehicle instantaneously reacts to the commands (better results for 4G than for Wi-Fi);
- The RPI version 2 was a big improvement. On a first stage a RPI version B+ was used, but the results were not satisfactory: video buffering and delay, and commands delay. So, the RPI 2 makes possible the real-time operation, mostly due to its huge processing capacity.
- From the tested scenarios, the indoor was the worst in terms of delay (RTT): the video has no delay but the vehicle take some time to react to the commands when far from the AP. These comment is based on the made tests and respective Wi-Fi configurations;
- However, on the other cases the system afford the real time operation, both in data and video transmission. The obtained delays (RTT) are within the requirements for a real-time application – both command and video low delay tolerance, high bandwidth for video and low/medium delay jitter tolerance for command/video;

8.2 Road ahead

As future work some improvements can be made on the system:

- Failsafe mechanisms can be implemented in order to preserve the vehicle’s integrity;
- Original aim of these project: implementation on a real car;
- Security: prevent the system from being attacked from the outside;
- Promote the interoperability between multiple systems.