ACADEMIA MILITAR
MILITARY ACADEMY

TÉCNICO LISBOA

# Integration of a Commercial Unmanned Vehicle into the Army Communications System

**Second Lieutenant (OF-1) João Pedro Araújo Silva**

Thesis to obtain the Master of Science Degree in

## Military Electrical Engineering

Supervisor(s): Prof. Dr. António Manuel Raminhos Cordeiro Grilo
Lieutenant Engr. Artur Jorge Silva Coutinho Machado

### Examination Committee

| | |
|---|---|
| Chairperson: | Prof. Dr. José Silvestre Serra Silva |
| Supervisor: | Prof. Dr. João Nuno De Oliveira e Silva |
| Member of the Committee: | Prof. Dr. António Manuel Raminhos Cordeiro Grilo |
| | Lieutenant Colonel Engr. Pedro Miguel Pena Madeira |

**October 2024**

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

# Acknowledgments

To the Portuguese Military Academy and the Instituto Superior Técnico for all the academic and personal support over the past few years.

To my supervisor, Professor António Grilo, for all his help, motivation, advice, critical spirit, and availability over the last year. Without his experience and knowledge, this thesis would not have been possible.

To my co-supervisor, Lieutenant Machado, for challenging me with the theme and for all his help, time, knowledge, and material.

To my course director, Lieutenant Colonel Pena Madeira, for all his help and advice at a military and technical level.

To the SIC-T project team, in particular Captain Gomes, Captain Félix, and Lieutenant Gomes, thank you for all your help and readiness to answer questions and support the tests.

To my comrades in the Foxtrot class, for all the camaraderie and friendship over the years. Without a doubt, our stories and adventures have moulded the person I am today. It might have been possible without you, but it wouldn't have been as good.

To my family for all their support and the safe harbour they represent. To my parents, for all the education and values they have passed on to me, they have undoubtedly had a fundamental role in building who I am today. To my sister Mariana for all our adventures and help throughout my life.

To my beloved Inês, for all her patience, encouragement, and love, everything would have been much more difficult without you.

# Resumo

Com a crescente evolução do campo de batalha, são cada vez mais evidentes as vantagens da utilização de plataformas Unmanned Vehicle (UxV) em ambientes ou tarefas perigosas para os militares. Os UxVs reduzem a exposição humana ao perigo em ambientes de alto risco. Assim sendo, é importante que a informação principal do veículo possa ser vista pelos comandantes no Common Operational Picture (COP) através da integração do veículo no Sistema de Comunicações do Exército.

Esta tese explora a integração dos UxVs comerciais em operações militares, focando-se na melhoria da comunicação entre estes veículos e os seus centros de controlo. Os conflitos globais, nomeadamente a guerra da Ucrânia, aceleraram o desenvolvimento de sistemas militares autónomos. O Exército Português está a investir nestas tecnologias como forma de baixar custos e acelerar a aquisição, desenvolvimento, e integração destas capacidades. Estas plataformas melhoraram a tomada de decisão e a eficiência operacional, podendo ser aplicadas não só no Exército Português, como também nos restantes ramos das Forças Armadas. A tese contribui com o desenvolvimento de uma biblioteca MAVLink para comunicação UxV, ferramentas híbridas para transmissão de dados usando rádios militares e Wireless Fidelity (Wi-Fi), e navegação de UxVs sem hardware de piloto automático dedicado.

A solução desenvolvida foi implementada utilizando o UGV LAFVIN, e os testes foram realizados com apoio da Equipa Projeto SIC-T. Após a obtenção dos resultados, foi possível retirar valores e conclusões importantes relativamente à utilização de rádios militares — P/PRC-525 e TR5000H — para a integração de um UGV no Battlefield Management System (BMS).

**Palavras-chave:** UGV, BMS, MAVLink, Rádios Militares, Wi-Fi.

# Abstract

With the growing evolution of the battlefield, the advantages of using UxVs platforms in dangerous environments or tasks for the military are becoming increasingly apparent. UxVs reduce human exposure to danger in high-risk environments. That being so, it is important that the main information from the vehicle can be seen by the commanders in the COP by integrating the vehicle into the Army Communications System.

This thesis explores the integration of commercial UxVs in military operations, focusing on enhancing communication between these vehicles and their control centres. Global conflicts, particularly the Ukraine war, have accelerated the development of autonomous military systems. The Portuguese Army is investing in these technologies to lower costs and speed up the acquisition, development and integration of these capabilities. These platforms have improved decision-making and operational efficiency and can be applied in the Portuguese Army and the other branches of the Armed Forces (AF). The thesis contributes by developing a MAVLink library for UxV communication, hybrid tools for data transmission using military radios and Wi-Fi, and unmanned vehicle navigation without dedicated autopilot hardware.

The solution developed was implemented using LAFVIN UGV and the tests were carried out with the support of the SIC-T project team. After obtaining the results, it was possible to draw important values and conclusions regarding using military radios — P/PRC-525 and TR5000H — for integrating a UGV into the BMS.



**Keywords:** UGV, BMS, MAVLink, Military radios, Wi-Fi.

# Contents

# List of Tables

# List of Figures

# Glossary

| | |
|---|---|
| **AF** | Armed Forces |
| **ARTEX** | ARmy Technological EXperimentation |
| **BMS** | Battlefield Management System |
| **C2** | Command and Control |
| **C4** | Command, Control, Communications, and Computers |
| **CCB** | Battalion Communications Centre (*Centro de Comunicações de Batalhão*) |
| **CCC** | Company Communications Centre (*Centro de Comunicações de Companhia*) |
| **CEMTEx** | *Centro de Experimentação e Modernização Tecnológica do Exército* |
| **CGR** | Network Management Centre (*Centro de Gestão de Rede*) |
| **CNR** | Combat-Net Radio |
| **COP** | Common Operational Picture |
| **DoD** | Department of Defense |
| **DSS** | Dismounted Soldier System |
| **FCS** | Future Combat System |
| **FFI** | Friendly Force Information |
| **FFT** | Friendly Force Tracking |
| **FPC** | Flexible Printed Circuit |
| **GCS** | Ground Control Station |
| **HF** | High Frequency |
| **ITU** | International Telecommunication Union |
| **JAUS** | Joint Architecture for Unmanned Systems |
| **LTE** | Long-Term Evolution |
| **MANET** | Mobile Ad Hoc Network |
| **MU MIMO** | Multi-User Multiple-Input Multiple-Output |
| **MU OFDMA** | Multi-User Orthogonal Frequency Division Multiple Access |
| **NA** | Access Node (*Nó de Acesso*) |
| **NATO** | North Atlantic Treaty Organization |
| **NEC** | Network Enabled Capability |
| **NT** | Transit Node (*Nó de Trânsito*) |
| **OFDM** | Orthogonal Frequency Division Multiplexing |

| | |
|---|---|
| **PAR** | Radio Access Point (*Ponto de Acesso Rádio*) |
| **QGC** | QGroundControl |
| **QoS** | Quality of Service |
| **REPMUS** | Robotic Experimentation and Prototyping augmented by Maritime Unmanned Systems |
| **RF** | Radiofrequency |
| **RL** | Rear Link |
| **RX-C** | Receiver Client |
| **SAE** | Extended Area Subsystem (*Subsistema de Área Estendida*) |
| **SAL** | Local Area Subsystem (*Subsistema de Área Local*) |
| **SCTP** | Stream Control Transmission Protocol |
| **SDR** | Software Defined Radio |
| **SGR** | Network Management Subsystem (*Subsistema de Gestão de Rede*) |
| **SIC-Op** | Operational Information and Communications System (*Sistema de Informação e Comunicações Operacional*) |
| **SIC-T** | Tactical Information and Communications System (*Sistema de Informação e Comunicações Tático*) |
| **SICCE** | Army Command and Control Information System (*Sistema de Informação para o Comando e Controlo do Exército*) |
| **SITACO** | Tactical Communications System (*Sistema Tático de Comunicações*) |
| **SSR** | Network Security Subsystem (*Subsistema de Segurança de Rede*) |
| **STANAG** | Standardization Agreement |
| **SUM** | Mobile User Subsystem (*Subsistema de Utilizadores Móveis*) |
| **TCP** | Transmission Control Protocol |
| **TO** | Theater of operations |
| **TX-C** | Transmitter Client |
| **U/E/O** | *Unidades, Estabelecimentos e Órgãos* |
| **UAV** | Unmanned Aerial Vehicle |
| **UDP** | User Datagram Protocol |
| **UGV** | Unmanned Ground Vehicle |
| **UHF** | Ultra High Frequency |
| **UMV** | Unmanned Maritime Vehicle |
| **USA** | United States of America |
| **UxV** | Unmanned Vehicle |
| **VHF** | Very High Frequency |
| **Wi-Fi** | Wireless Fidelity |
| **WLAN** | Wireless Local Area Network |

# List of Software

| | |
|---|---|
| **Visual Studio Code V1.94.0** | Code editor |
| **Arduino IDE V1.8.19** | Arduino Integrated Development Environment |
| **Putty V0.81** | SSH and telnet client |
| **MAVProxy V1.8.71** | GCS software |
| **Mission Planner V1.3.81** | GCS software |
| **GStreamer V1.18.4** | Pipeline-based multimedia framework |
| **BMS V4.4** | Battlefield Management System |
| **Wireshark V4.4.0** | Network protocol analyser |
| **iPerf3 V3.9** | Speed test tool |

# Chapter 1

# Introduction

Human life is priceless, and the loss of life can be reduced considerably with the use of robots and autonomous systems in the AF. Today, almost all military organisations use robots to perform dangerous tasks in high-risk environments [1].

A UxV is a vehicle that operates without a human onboard, either remotely controlled or autonomously. There are several types of UxVs: Unmanned Ground Vehicle (UGV), Unmanned Aerial Vehicle (UAV) and Unmanned Maritime Vehicle (UMV) [2]. Communications in a UxV are essential to enable remote control and data transmission between the vehicle and the operator [1].

## 1.1 Motivation

In recent years, in the face of increasing military tensions at a global level, especially with the start of the conflict in Ukraine on 24 February 2022, there has been a notable rush towards military technology. The purpose of this campaign was to improve defence capabilities. The North Atlantic Treaty Organization (NATO) identified Autonomous Systems and Communications as technologies that will play a highly disruptive role over the next two decades, playing a crucial role in improving human performance and promoting symbiosis between man and machine [3].

The Portuguese Army, through the 2023 Military Programming Law, is making a significant investment in Autonomous Systems through Project EXE02 — Remote and Autonomous Systems. This work is aligned with this work package, which aims to develop a communication subsystem to be applied in UGVs. This capability will help military command decisions and reduce human exposure to threats.

Research into communications and remote control of UGVs aims to improve the safety, efficiency, and ability to operate these vehicles in challenging environments. By developing advanced communication technologies, it is possible to increase the autonomy, precision and reliability of operations and, additionally, the interaction of UGVs with other systems, such as UAVs, sensors and artificial intelligence systems. This includes the implementation of efficient communication networks that enable the exchange of information and coordination between the different components of the system and the development of algorithms for cooperation and collaboration between the vehicles [4]. This UGV integration

is part of the development plans of the Tactical Information and Communications System (*Sistema de Informação e Comunicações Tático*) (SIC-T) of the Portuguese Army Information and Communications System.

The use of commercial UxV platforms by military forces is currently an important trend. Commercial UAVs are easy to procure, require minimal training, and can be quickly replaced, allowing for rapid deployment in the field. While not a replacement for military-grade UAVs, they act as valuable force multipliers for soldiers on the ground, providing crucial tactical advantages at a lower cost [5].

The integration of commercial UAV platforms, such as DJI, began with terrorist forces — the Taliban, Islamic State, Hamas, and Hezbollah. However, countries like the United States of America (USA) already have programmes to integrate this platform type. The Russia-Ukraine and Gaza wars have highlighted the growing integration of commercial UAVs into military operations. These UAVs are favoured for their affordability, accessibility, and adaptability, making them ideal for ground forces in dynamic battle environments [5].

## 1.2 Objectives

The main objective of this project is to design an architecture and develop software for the integration of commercial UGVs in the SIC-T. The primary purpose of communications in UGVs is to guarantee reliable, real-time data transmission. This allows the operator to receive information from the vehicle, such as sensor data, images and video, and send control commands to remotely guide the UGV. Communication protocols are crucial in establishing a reliable and efficient connection between an UGV and the control centre [6].

Within this main objective, the thesis has the following secondary objectives:

- To support piloting of a commercial UGV, without the need for specific autopilot hardware such as Pixhawk line or Ardupilot Mega;
- Development of a library that connects a UxV to the SIC-T;
- Testing and evaluation of the developed solution.

## 1.3 Contributions

This thesis has made numerous contributions to the subject of UxVs in the Portuguese Army, among which the following stand out:

- Development of a MAVLink library to connect a commercial UxV to the SIC-T, that can be used in future UxV exercises;
- Development of a hybrid tool that allows the UxV position to be sent via tactical radios while simultaneously sending the other driving parameters via Wi-Fi;

- Testing and evaluation results of UGV driving using tactical radios;
- The first time, as far as the author knows, that a commercial UGV has been integrated into the BMS.

## 1.4 Thesis Outline

This document is structured as follows:

- Chapter 1, the present chapter, introduces the topic, followed by the motivation, objectives, contributions, and thesis outline.
- Chapter 2 gives an overview of the evolution of UxVs and the main application areas in the civilian and military spheres. The application of UGVs in the war between Russia and Ukraine is also covered. The state-of-the-art video compression and transmission and radio communication technologies applied to UGVs are then presented. Communications Systems for Commercial UxVs and the Military Information and Communications Systems are then discussed, followed by an explanation of UGV hardware components. Finally, the communications performance evaluation tools, related work, and summary of this chapter are discussed.
- Chapter 3 provides a solution for integrating a commercial UGV into the Army Communication System. For this, two architectures are used: with autopilot hardware and without. The architecture without autopilot hardware is divided into two transmission scenarios — Wi-Fi and Military Radios — and is described in detail in the following sections. Subsequently, the software and hardware modules for the architecture without hardware autopilot are presented. Finally, the ARTEX 24 exercise is covered in the first phase, with what was developed and implemented, and, lastly, with the lessons learned and proposals for future improvement.
- Chapter 4 covers the problem description and then the baseline solution for each scenario. Finally, the solution is evaluated, with data from the communication performance evaluation tests obtained in each scenario.
- Finally, Chapter 5 addresses the achievements of this master's thesis and future work.

# Chapter 2

# Background and Related Work

The UGVs can navigate in spaces that would be dangerous or inaccessible to humans and can operate in increasingly complex environments through intelligence and efficiency. By equipping an UGV with a camera or other sensors, operators can quickly analyse objects and regions of interest with precision [7]. Space exploration, particularly on the Moon and Mars, is among the most important. In recent years, NASA has developed five UGV platforms: Sojourner (1997), Opportunity (2004), Spirit (2004), Curiosity (2012), and Perseverance (2021). In parallel, the China National Space Administration has also developed an UGV, the Zhurong (2021) [8].

As for military UGVs, they have a long lineage, descending from the Teletank of Russian origin and the Goliath of German origin, both used in the Second World War. UGVs, as we know them today, emerged in the second half of the 20th century, around 1970, with the flourishing of microprocessor technology. Many developments in robotics originated from the Department of Defense (DoD) of the USA research and development projects. Despite the technological advances, in the 1990s, military interest was flagged with the end of the Cold War. However, optimism about robotics continued to grow in the new millennium, with interest in equipping the AF with smaller, cheaper, and more resistant UGVs. Counter-insurgency operations in Iraq and Afghanistan boosted the use of UGVs, mainly to deal with improvised explosive devices [9].

The ongoing war in Ukraine has had a profound impact on the development of UGVs in Russia. Both sides are increasingly deploying UAVs and UGVs for various tasks such as combat, logistics, and medical evacuation. This conflict has underscored the importance of UGVs in reducing soldier casualties in hazardous situations [10].

Before the February 2022 invasion, Russian UGV initiatives were still nascent, with prototypes like Uran-9 and Nerekhta undergoing limited testing and facing various constraints. These early models were not yet integrated into broader military strategies [10].

The war has accelerated the need for UGVs to adapt swiftly to high-risk conditions, leading to a direction toward more compact, cost-effective, and disposable UGV designs. The UGVs used in the conflict include those for surveillance and supply, and volunteer contributions have driven a surge in innovative, low-cost designs. The current focus is creating UGVs that operate efficiently amid intense

aerial and ground-based countermeasures [10].

Figure 2.1: The BRG-1 light UGV [11].

## 2.1 UxV applications

This section presents UxVs applications, with special emphasis on civilian and military applications of UGVs. The first UGV application was in the military field, with the Teletank wireless remotely controlled and used for carrying explosives and a machine gun. The Goliath UGV was controlled by cable, carrying approximately 60 kg of explosive material, serving as ammunition [12]. Only in the 1970s did civilian applications with UGVs — such as nuclear power plants or space exploration — begin [9].

### 2.1.1 Civilian UGV applications

The civilian applications of UGVs are wide. They can include power line testing, surveillance systems, mine detection, data collection, agriculture, ocean exploration, and scientific exploration in space as the main [13]. In much the same way as they are used in the military, UGVs can also be used by the police for surveillance and intervention operations in hostile scenarios [14].

UGVs offer advantages over human operators, as they can be used more often and consistently, potentially providing substantial cost savings. Autonomous systems can also be operated without safety measures or the equipment needed by humans [7].

Regarding UAVs, they have gained significant popularity due to their ability to capture high-quality multimedia content from the air. With applications such as aerial photography, cinematography, and cartography, UAVs are increasingly being used to collect various multimedia data [15].

### 2.1.2 Military

In the military context, UGVs are operated remotely, using wireless communication systems such as Radiofrequency (RF), which enable real-time video transmission. Advanced image processing techniques are used for automatic target detection.

The dismounted mission support system requirements were first outlined within the framework of the US Army's Future Combat System (FCS) program, which took place between 2003 and 2009. The support tasks should generally cover [16]:

- Combat support: involving reconnaissance, surveillance, target acquisition, and fire support;
- Engineering support: involving mine detection, clearing paths, destroying obstacles, building light bridges, and fortifying terrain;
- Logistical support: transporting ammunition, collective weapons, water, food, and medical evacuation.

Russia's first attempt at a robotic ground assault using artificial intelligence-controlled UGVs ended disastrously. The robots deployed to assault the Ukrainian 53rd Mechanised Brigade failed to achieve their objectives and were quickly neutralised. The failure highlights significant issues with Russia's robotic warfare capabilities, including technical malfunctions and vulnerabilities to electronic warfare [17].

Swarm robotics focuses on coordinating large groups of simple robots to achieve complex tasks inspired by the behaviour of social insects like ants and bees. This approach uses decentralised control, self-organisation, and local interactions to create efficient, robust, scalable systems. Advantages include resistance to individual robot failures and scalability, making swarm robotics suitable for military applications such as logistics and surveillance [18].

Local communication among robots minimises the need for a central control unit, enhancing efficiency but posing challenges in maintaining coherent global behaviour. Advanced algorithms for coordination and task allocation address these challenges [18].

Using heterogeneous vehicle swarms enhances the effectiveness of UAVs, UGVs, and UMVs in various missions by combining their strengths and mitigating their weaknesses, represented in Figure 2.2. UAVs are effective for rapid exploration and communication but are limited by flight time and payload capacity. UGVs complement UAVs by operating on diverse terrains and providing support roles, such as recharging stations. UMVs improve naval capabilities with missions like mine countermeasures and maritime security [11].

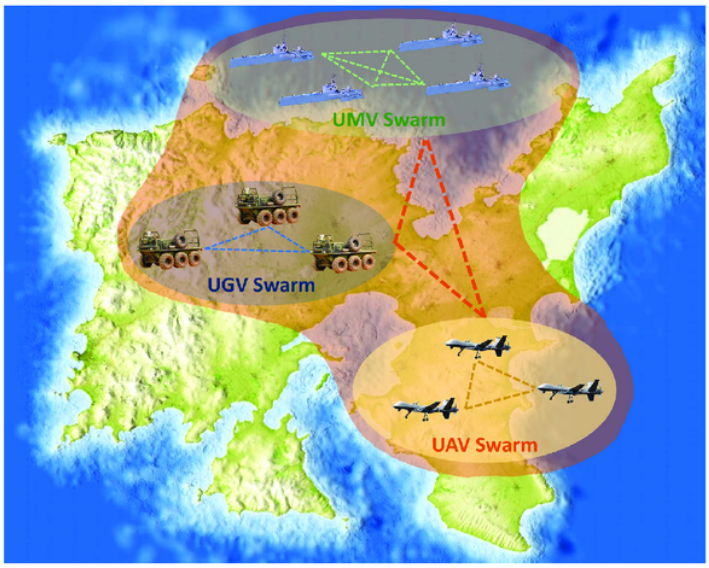


Figure 2.2: Inter-swarm collaborations [11].

## 2.2 Video compression and transmission

When piloting an UGV, video transmission becomes an aspect of utmost importance. The main problem with this transmission type is the high bandwidth demand, which makes video compression very important. This action aims to make video content available at a low bits rate while preserving satisfactory image quality, in this case, for piloting the UGV. This technique works by identifying and eliminating redundancies [19].

### 2.2.1 Video compression

With the need for video compression came video encoders, among which the most important and widely used in video transmission was x264, which encodes in the H.264 format, also known as Advanced Video Coding (AVC) or MPEG-4 Part 10, introduced by International Telecommunication Union (ITU) in May 2003. The separation of decoding, sampling, and transmission data are essential features of this standard, as they allow different configurations for decoding and transmission. Two levels are considered in the H.264 standard: the network abstraction level and the video coding level. The former converts H.264 into a network-compatible standard, while the latter encapsulates the video data and determines the header for each unit. The H.264 standard is based on the macroblock; each macroblock represents a part of the video frame, and through this technique, internal prediction can be achieved, that is, within each frame [20]. In addition, there is also prediction between different frames through motion compensation and motion estimation [21].

### 2.2.2 Video transmission

FFmpeg is a free, open-source software with libraries and programs for processing and transmitting video, audio, and other multimedia files. It is software used on the command line and can use the H.264 standard [21, 22]. FFmpeg is used on the sender and receiver side; encoding video data is divided into initialisation, calling the decoding function, calling the encoding function, calling the decoding function and releasing the resource [21].

GStreamer is a comprehensive multimedia framework for developing complex multimedia processing applications that can utilise the H.264 standard. It enables the creation of media pipelines using modular components called elements, which handle tasks such as encoding, decoding, and filtering. These elements are organised into compartments, which can be nested hierarchically, with pipelines acting as top-level compartments that manage the overall data flow and state of the media processing [23].

Elements are crucial objects in GStreamer, each with a specific function, such as reading data from a file, decoding it, or sending it to an output device. A pipeline can be created that performs tasks like media playback or capture by combining multiple elements. Blocks are elements input and output points, allowing connections between them. They have specific data handling capabilities and can constrain the data types handing through. Pipelines facilitate synchronisation among elements to ensure smooth media processing [23].

## 2.3 Communication Systems for Commercial UxVs

This section covers Radio communication technologies applied to Commercial UGVs and the MAVLink communication protocol for autonomous systems.

### 2.3.1 Radio communication technologies applied to Commercial UGVs

This subsection covers the leading radio communication technologies in UxVs: Wi-Fi, Cellular Networks, and ZigBee. It also shows proprietary technologies and hardware specifically applied to UxVs.

#### Wi-Fi

The IEEE 802.11 standard for an Wireless Local Area Network (WLAN), known commercially as Wi-Fi, has become essential daily. The evolution of this technology also resulted in the first commercial experience of high-speed optical communications, Orthogonal Frequency Division Multiplexing (OFDM), MIMO, and mmWave transmission technologies, which were later widely adopted by the mobile phone and wireless sensor network industries [24]. Wi-Fi most commonly uses the 2.4 GHz and 5.8 GHz radio bands.

Using Wi-Fi to command and control UxVs offers several advantages that enhance operational efficiency and flexibility. One primary benefit is the ability to establish a robust and high-bandwidth communication link, which supports the transmission of large amounts of data, such as high-resolution video

and complex telemetry, in real-time. This capability is advantageous in applications requiring detailed situational awareness and precise control [25].

Another significant advantage is the simplicity of deployment and cost-effectiveness of Wi-Fi infrastructure. Wi-Fi technology is widely available and relatively inexpensive compared to other communication systems, making it accessible for various applications. The existing Wi-Fi infrastructure in urban and industrial areas can be leveraged to facilitate seamless and extended-range operations without the need for dedicated communication networks [26].

The range of a Wi-Fi signal is influenced by factors such as the frequency band, radio power output, antenna gain, and antenna type, along with the modulation technique used. An access point that follows the 802.11b or 802.11g standards (2.4 GHz), using a standard antenna, typically has a range of about 100 meters. This range in UGV applications can be a limitation, representing a short-range wireless communication [25].

The most recent standard for this technology, which has already been implemented, is Wi-Fi 6E, also known as IEEE 802.11ax. The Wi-Fi 6E standard uses the 6 GHz frequency band and is committed to providing data transfer rates of up to 9.6 Gb/s, with lower energy consumption and excellent reliability. Its capabilities significantly exceed those of Wi-Fi 5 (802.11ac). The Wi-Fi 6E standard incorporates Multi-User Orthogonal Frequency Division Multiple Access (MU OFDMA), Multi-User Multiple-Input Multiple-Output (MU MIMO), new spatial reuse mechanisms, higher-order modulation, and other subtle improvements. The Wi-Fi 7 (802.11be) standard is expected to be launched by the end of 2024, and data transfer rates of up to 30 Gb/s are expected [27].

#### Cellular Network

Cellular networks consist of several base stations, each dedicated to the wireless transmission and reception of data in one or more “cells”. In this context, a cell represents a specific region of the total geographical area served by the network. In most deployments, one base station covers three cells using a strategic antenna configuration and efficient frequency planning [28].

The specific way in which the RF bands are used in mobile telecommunications is called the wireless standard — the name given to each generation of mobile telecommunications — which have replaced each other over the last 30 years: 1G, 2G, 3G, 4G, and 5G. Each generation comprises a different family of wireless protocols [29].

6G technology has already been announced and is still in the applied research and technological testing phase. 6G is seen as a union of physical space, cyberspace, and connectivity with intelligence. Artificial intelligence has the potential to be the basis of 6G so that all its components are intelligent. The prominent use cases for this environment will include telesurgery and remote control of heavy equipment, motor vehicles, aircraft, and even ships. Figure 2.3 represents an application of 6G technology in the remote control of an industrial vehicle.

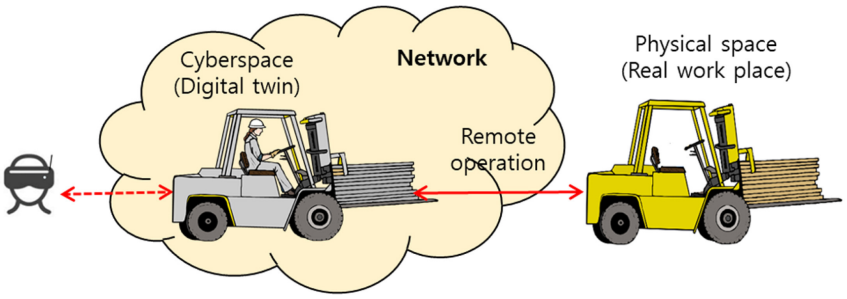


Figure 2.3: Immersive internet based on cyber-physical systems [30].

The private Long-Term Evolution (LTE) networks offer extensive customisation and control, allowing organisations to tailor configurations to meet specific requirements and maintain complete control over their network infrastructure. Enhanced security is another significant benefit, as private networks are isolated from public networks, reducing the risk of external threats and allowing for custom security measures. Regarding reliability and performance, private networks provide dedicated resources, ensuring consistent performance, low latency, and predictable and stable performance without the congestion typically associated with public networks. Flexibility and scalability are also emphasised, with private networks being easily adaptable to changing organisational needs and capable of scaling to accommodate increasing numbers of devices and data traffic. These advantages make private networks suitable for specialised and mission-critical applications [31].

#### ZigBee

Zigbee is a widely adopted communication protocol for low-power wireless communication based on the IEEE 802.15.4 standard. It is maintained by the Zigbee Alliance and is developed on the physical and medium access control layers [32]. This protocol is also used for applications with UGVs, as can be seen in [33] and [34].

Zigbee represents a low-range communication standard that operates in three bands: 2.4 GHz, 868, and 915 MHz frequency bands, with a data transmission rate between 20 and 250 Kbit/s. This protocol uses 27 different channels, distributed as follows: 16 channels in 2.4 GHz, 10 channels in 915 MHz, and 1 channel in the 868 MHz band [32].

#### Proprietary communication protocols and hardware

There are several examples of proprietary communication protocols for communicating with a UxV; this section presents some of the most commonly used hardware.

The SiK Telemetry Radio is an open-source, small, and lightweight radio platform for establishing telemetry connections between autopilots and ground stations. It connects easily to Pixhawk controllers and computers via USB; the following features stand out on this platform [35]:

- Communication range: over 300 meters extendable with better antennas;
- Frequency: 915 MHz or 433 MHz;

- Key features: MAVLink compatibility, frequency hopping, error correction, and transparent serial links.

The HM30 Full HD Transmission System by SIYI Technology offers several key features [36]:

- Communication range: Up to 30 kilometres with directional antennas and up to 15 kilometres with standard omnidirectional antennas;
- Frequency: 5100 to 5825 MHz band, centred on the 5550 MHz;
- Video: Supports dual-channel high-definition digital image transmission;
- OSD (On-screen display) Telemetry: Displays data telemetry and OSD information based on the Mavlink protocol;
- Dual operator capability: Allows two ground units to control different aspects of the same air unit;
- Remote control relay and wireless repeater: Extends control range by relaying signals through multiple ground units.

The Herelink system by CubePilot is an advanced remote control and video transmission system designed for UxVs. It integrates a remote controller with a built-in screen on a customised Android operating system. This system supports popular ground control software like QGroundControl and Solex. The main specifications are [37]:

- Communication range: Up to 20 kilometres;
- Frequency: 2.4 GHz;
- Video Latency: Ultra-low latency;
- Outputs: Multiple control and video output options.

### 2.3.2 MAVLink

MAVLink is a middleware protocol that is a universal language for autonomous systems. The main function of this protocol is to ensure that UxVs can easily talk to each other and Ground Control Station (GCS) by sharing important data like their positions. Almost everyone in the autonomous systems world uses MAVLink, even big companies that do not always use open-source [38]. MAVLink messages are formatted in binary code with a compact 8-byte header, which makes them highly efficient. Built-in error control mechanisms ensure data integrity [39].

The big deal with MAVLink is that it creates one common way to communicate with UxVs, ground stations, and accessories. This means more options for everyone involved, from autonomous systems makers to people using them, and it stops any company from dominating the market [38]. MAVLink is about improving how UxVs work together, sparking new ideas, and helping the industry grow [38].

Since UAVs navigate through three-dimensional space, they present a broad scenario for communicating position and velocity data. Consequently, it makes sense to utilise the same protocol for other

vehicles, like UGVs moving in two-dimensional space or UMVs, which operate in three dimensions but beneath the ocean's surface [38].

MAVLink has 2 versions:

- MAVLink v2.0- current version adapted by many users from early 2017;

- MAVLink v1.0- was adopted around 2013. Several legacy peripherals still use it.

MAVLink v2.0 supports C/C++ and python libraries (pymavlink), which are also backwards compatible with MAVLink v1.0. The handshaking and negotiating version tells us which version is used [40].

The main MAVLink messages used in a connection are HEARTBEAT, GPS_RAW_INT, and MANUAL_CONTROL.

The heartbeat message (HEARTBEAT(0)) indicates that a system or component is present and responsive, and the parameters are [39]:

| Field Name | Type | Description |
|---|---|---|
| type | `uint8_t` | Vehicle or component type. For a flight controller component, the vehicle type (e.g. quadrotor, helicopter, etc.). The component type (e.g. camera or gimbal) is used for other components. This should be used in preference to component ID to identify the component type. |
| autopilot | `uint8_t` | Autopilot type / class. Use MAV_AUTOPILOT_INVALID for components that are not flight controllers. |
| base_mode | `uint8_t` | System mode bitmap. |
| custom_mode | `uint32_t` | A bitfield for use for autopilot-specific flags. |
| system_status | `uint8_t` | System status flag. |
| mavlink_version | `uint8_t _mavlink_version` | MAVLink version, not writable by user, gets added by protocol because of magic data type: uint8_t_mavlink_version. |

Table 2.1: HEARTBEAT message [39].

The GPS_RAW_INT(24) message represents the raw sensor value of the vehicle, and the main parameters are [39]:

| Field Name | Type | Description |
|---|---|---|
| time_usec | uint64_t | Timestamp (UNIX Epoch time or time since system boot). |
| fix_type | int8_t | GPS fix type. |
| lat | int32_t | Latitude. |
| lon | int32_t | Longitude. |
| r | int16_t | R-axis, normalised to the range [-1000,1000]. |
| alt | int32_t | Mean Sea Level (MSL) altitude. Almost all GPS modules provide the MSL altitude and the WGS84 altitude. |

Table 2.2: GPS_RAW_INT message [39].

MANUAL_CONTROL(69) message allows the vehicle to be controlled manually, using the standardised terminology of the joystick axes. The main parameters are [39]:

| Field Name | Type | Description |
|---|---|---|
| target | uint8_t | The system to be controlled. |
| x | int16_t | X-axis, normalised to the range [-1000,1000]. A value of INT16_MAX indicates that this axis is invalid. |
| y | int16_t | Y-axis, normalised to the range [-1000,1000]. A value of INT16_MAX indicates that this axis is invalid. |
| z | int16_t | Z-axis, normalised to the range [-1000,1000]. A value of INT16_MAX indicates that this axis is invalid. |
| r | int16_t | R-axis, normalised to the range [-1000,1000]. A value of INT16_MAX indicates that this axis is invalid. |
| buttons | uint16_t | A bitfield corresponding to the current state of joystick buttons (0-15). 1 for pressed, 0 for release. The lowest bit corresponds to button 1. |

Table 2.3: MANUAL_CONTROL message [39].

### 2.3.3 Ground Control System

This section presents the most relevant examples of ground control systems. Each one has its own characteristics that make it more advantageous for certain situations. These systems are Mission Planner, MAVProxy, and QGC:

### Mission Planner

Mission Planner is an open-source, full-featured GCS application that supports UxV operations. Michael Oborne developed Mission Planner, which provides a comprehensive suite of tools for configuring, monitoring, and managing UxV operations, enhancing efficiency and precision [41].

Mission Planner's primary functionalities include mission planning, parameter configuration, and real-time monitoring. Its user-friendly graphical interface allows operators to set waypoints and routes on an interactive map, facilitating the creation of complex missions with specific actions at each waypoint, such as taking photographs or adjusting altitude. It also supports detailed adjustments to flight parameters, including flight modes, safety limits, and sensor settings, ensuring accurate calibration for compasses, accelerometers, and gyroscopes [41].

Moreover, Mission Planner offers robust real-time monitoring, displaying telemetry data like GPS position, speed, altitude, and battery status. Its Heads-Up Display (HUD) feature simulates cockpit instruments, providing comprehensive situational awareness. The software also allows downloading and analysing flight logs to identify issues and optimise vehicle performance [41].

### MAVProxy

MAVProxy is a robust open-source GCS software designed for UxV operations, particularly those using the MAVLink protocol. Developed initially by CanberraUAV, MAVProxy has grown into a versatile tool used widely by developers and UxV operators for its powerful command-line interface and modular design [42].

MAVProxy's core functionalities include telemetry monitoring, mission planning, and UxV control. It operates primarily through a command-line interface, making it highly flexible and extensible via numerous modules. These modules add functionalities such as map display, console command interface, and advanced scripting capabilities. MAVProxy supports real-time telemetry data display, providing critical information like GPS coordinates, altitude, speed, and battery status [42].

One of MAVProxy's advantages is its lightweight architecture, which allows it to run on various operating systems, including Windows, Linux, and MacOS, making it suitable for simple and complex UxV operations. It can be used standalone or with other GCS software like Mission Planner to provide a more graphical user interface [42].

### QGroundControl

QGC is a versatile GCS application for UxVs, offering mission planning, vehicle configuration, and real-time monitoring. Its user-friendly interface allows operators to set waypoints and routes on an interactive map, facilitating complex missions with specific actions at each waypoint, such as taking photographs or adjusting altitude. QGC supports detailed adjustments to flight parameters, including flight modes, safety limits, and sensor settings, ensuring accurate calibration of compasses, accelerometers, and gyroscopes [43].

Additionally, QGC provides robust real-time monitoring, displaying telemetry data such as GPS position, speed, altitude, and battery status. The software includes safety features like geofencing and return-to-home protocols, enhancing operational reliability. The open-source nature of QGC allows for extensive customisation and integration with other systems; it is designed to run on multiple operating systems, including Windows, macOS, Linux, iOS, and Android, making it a flexible tool for various UxVs applications [43].

## 2.4 Military Information and Communications Systems

This section discusses the most relevant international standards regarding information and communications systems and, in detail, the Portuguese information and communications system.

### 2.4.1 International standards

The Joint Architecture for Unmanned Systems (JAUS) is an international standard that establishes a common set of message formats and communication protocols to support interoperability within and between UGVs and control stations. It was created by DoD to provide an open architecture for the UGV domain. It was later converted into an international industry standard and currently belongs to the Society of Automotive Engineers. All the normative documents that define the JAUS can be purchased from the Society of Automotives Engineers, which maintains the main objective of defining open communication standards to support the interoperability of robotic systems in the military sector [44].

Standardization Agreement (STANAG) 4677, officially known as “Dismounted Soldier Systems Standards and Protocols for Command, Control, Communications, and Computers (C4) Interoperability,” is a NATO Standardisation Agreement developed to ensure interoperability through standardised information exchange protocols among C4 systems used by dismounted soldiers within NATO and partnership for peace nations [45].

NATO STANAG 5527 aims to respond to the following interoperability requirements. To provide friendly ground force information throughout the NATO Command Structure, within the respective chain of command, and to increase situational awareness of the land battle space. The specific benefits of an Friendly Force Tracking (FFT) capability in NATO are improved overall situational awareness, Command and Control (C2), force multiplication, improved synchronisation of forces, and reduced risk of fratricide [46].

NATO started FFT in 2004 with the FFT working group aiming for a NATO-wide capability by 2010. Due to operations in Afghanistan, a temporary standard, the NATO Friendly Force Information Standard, was adopted. The FFT working group evolved into the FFT capability team, which developed standards for using and exchanging information between different FFT systems, detailed in the NATO Message Catalogue, APP-11(D)(1) [46].

In any operation, whether national, multinational, coalition, or NATO, all command authorities need constant and highly accurate situational knowledge of the exact positions of all friendly forces. FFT is

the ability to deliver positional data from the lowest possible level. This data is integrated into tactical and operational pictures using available communication channels through a coordinated flow of information, an Friendly Force Information (FFI). As a basic principle, an FFI report shall contain the following minimum information [46]:

- Transponder Identification;
- System Name;
- Track Security Marking;
- Time;
- Location (Latitude, Longitude).

There are several editions of the NATO ATANAG 5527 for FFT: edition A, versions 1 and 2, and the most recent, which is still in draft, edition B [46].

### 2.4.2 Portuguese Information and Communications System

The Portuguese Army Information and Communications System can be divided into SIC-T and Operational Information and Communications System (*Sistema de Informação e Comunicações Operacional*) (SIC-Op). The SIC-T can be considered a temporary extension of the SIC-Op. The SIC-Op is permanent and fixed in nature and is installed in every *Unidades, Estabelecimentos e Órgãos* (U/E/O) of the Army [47].

#### SIC-T

The SIC-T can be divided into two strands: Tactical Communications System (*Sistema Tático de Comunicações*) (SITACO) and Army Command and Control Information System (*Sistema de Informação para o Comando e Controlo do Exército*) (SICCE). The SIC-T equips the Army's tactical component with the communications means and information system needed to conduct network-centred operations [47] and must provide the command with the correct information in the right place and at the right time [48]. This system was developed and based on full-IP technology, with the ability to adapt to the changing operational environment, ensuring compliance with the guiding principles Network Enabled Capability (NEC) of the NATO [49].

The structure of the SIC-T, specifically its communications component (SITACO), is organised into different subsystems, namely an Extended Area Subsystem (*Subsistema de Área Estendida*) (SAE), an Local Area Subsystem (*Subsistema de Área Local*) (SAL), an Mobile User Subsystem (*Subsistema de Utilizadores Móveis*) (SUM), an Network Management Subsystem (*Subsistema de Gestão de Rede*) (SGR) and an Network Security Subsystem (*Subsistema de Segurança de Rede*) (SSR). The SGR and SSR play a comprehensive role in the system. In turn, SSR is responsible for guaranteeing the security of the information transmitted on the network, while SGR is considered the "brain of the network". To

implement this functional architecture of the SIC-T, seven systemic modules were designed, each representing subsystems: Transit Node (*Nó de Trânsito*) (NT), Access Node (*Nó de Acesso*) (NA), Radio Access Point (*Ponto de Acesso Rádio*) (PAR), Rear Link (RL), Battalion Communications Centre (*Centro de Comunicações de Batalhão*) (CCB), Company Communications Centre (*Centro de Comunicações de Companhia*) (CCC) and Network Management Centre (*Centro de Gestão de Rede*) (CGR) [48]. Figure 2.4 shows the SIC-T integrated with all the subsystems mentioned above.

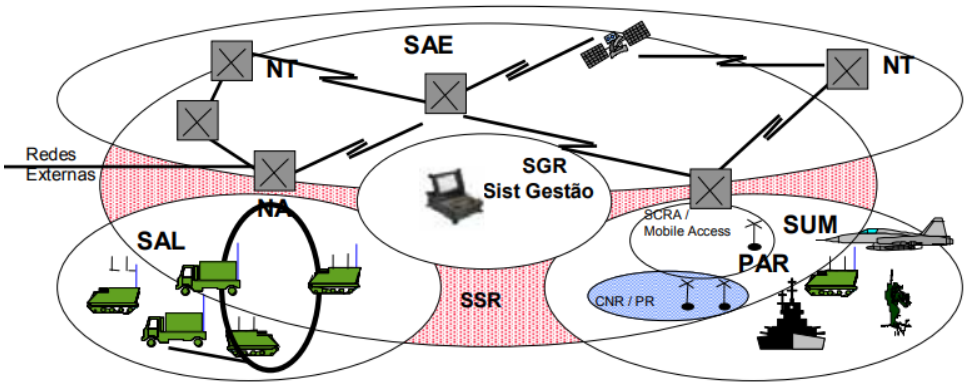


Figure 2.4: The SIC-T and its subsystems [48].

**BMS**

The BMS integrates the SICCE of the SIC-T, which in turn supports the C2. The BMS works via the radio network in the Very High Frequency (VHF) band in secure mode (SECOM-V). This C2 system is designed for low-ranking units (Battalion and below) in vehicular use, with an application that facilitates the sharing of geographical position in the field via portable terminals connected to a secure radio network [50].

In the context of a complex Theater of operations (TO), the BMS represents a crucial C2 tool, equipping low echelons, or echelons at manoeuvre level, collecting reliable data quickly and securely. It also enables near-real-time transmission to all chain of command levels, characterising it as a process that provides comprehensive situational awareness at both the frontline and command levels. The user can receive and share information on the COP with other units connected to the network, making it possible to [50]:

- Perform operations on the map, graphics, and layers;
- Perform terrain analysis and information filtering operations;
- Send and receive messages, requests, reports, and events.

The BMS fulfils the requirements of the NATO, guaranteeing interoperability with the means of communication of the alliance's member countries, through the STANAG 5527 standard - Friendly Force Tracking [51]. The general description of the BMS operation and its main features are presented below [52]:

**BMS operation:**

- A set of networks is established for the echelons;
- Each unit makes up its COP through the information received by the network;
- Each unit contributes its information to the network;
- Information is distributed horizontally and vertically.

**BMS features:**

- Displays own position and georeferenced friend positions;
- Object design (generic and APP6-D);
- Transparent design;
- Sending and receiving messages (APP11);
- Exchange quick text messages and files (chat and XMPP chat);
- Terrain analysis: elevation profile, line of sight, and calculation of areas and distances;
- Requests from NATO Vector Graphics;
- Start/Sop Gateways;
- Log viewer.

When setting up a mission in the BMS, it is mandatory to configure a GPS service. This can come from a radio device, a mobile device GPS service or using user defined position data (using a configured .KML file or manually introduced during execution). Figure 2.5 shows the list of predefined GPS [53].

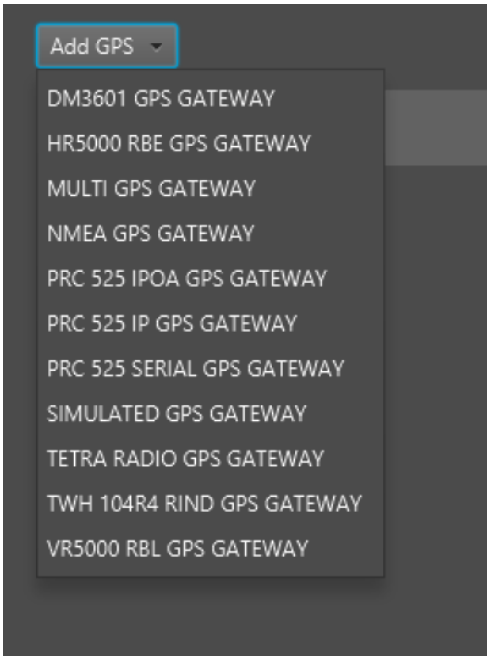


Figure 2.5: GPS list.

#### DSS

The Dismounted Soldier System (DSS) is the Portuguese Army's C2 system for mounted soldiers. This system is an extension of the BMS for platoon commanders and section commanders when they are outside a vehicle. The DSS allows access to the COP without physically being in the tactical vehicles, thus conferring a huge advantage in terms of C2 and situational awareness. DSS is interoperable with BMS and can be accessed via a data terminal using the Android operating system [54].

It should be noted that the Portuguese Army is developing a project using augmented reality to increase the situational awareness and the C2 of soldiers who use the DSS [54]. The DSS and this augmented reality project, in particular, will be one of the integration points to be considered with the SIC-T.

Its fully digitalized and modular architecture provides operational versatility. It can be customised to meet various scenarios and mission demands across all infantry levels, enabling the transmission of voice, data, and video. Furthermore, it will be completely compatible with existing communications equipment currently deployed in the Portuguese Army [55]. Figure 2.6 represents the DSS operating in the ARTEX 24.

Figure 2.6: DSS in the Exercise "ARTEX 24".

### 2.4.3 Army Tactical Radios

This subsection presents two tactical radios used by the Portuguese Army, the P/PRC-525 and the TR5000.

#### P/PRC-525

The P/PRC-525 Tactical Radio results from a partnership between EID and Rohde & Schwarz. It offers excellent flexibility in terms of frequency bands, as it operates on High Frequency (HF), VHF, and Ultra High Frequency (UHF) and follows the Software Defined Radio (SDR) philosophy. The radio is highly configurable, allowing new functions and waveforms to be added. One of the P/PRC-525 radio's

vulnerabilities is its low bandwidth (72 Kbit/s) to support data and voice simultaneously. This becomes more of a constraint when transmitting video, as it requires more bandwidth [50].

Some of the data communication modes available in the P/PRC-525 are as follows [56]:

- Digital Fixed Frequency Modulation (FSK): In FSK mode, the data interface provides transmission rates of 150, 300, 600, 1200, Depending on the selected band, 2400, 4800, 9600, 14400, 16000 bit/s. Transmission is in transparent mode;
- SECOM -V (16 Kbit/s): Offers transmission rates of 600, 1200, 2400, 4800, 9600 and 16000 bit/s. FEC (Forward Error Correction) is added to the data at all transmission rates below the maximum;
- SECOM -H (2.4 Kbit/s): Offers 600, 1200 and 2400 bit/s transmission rates with FEC;
- Internal 64Kbit/s data modem - OFDM: The OFDM data modem with transmission rates between 16 Kbit/s and 72 Kbit/s and variable bandwidth is available for the VHF/UHF bands. This modem operates in a transparent mode.

#### TR5000

In August 2019, to meet the Army's needs, TR5000 radios were purchased from Rohde & Schwarz's SOVERON family, in the vehicular and manpack versions, TR5000V and TR5000H, respectively [57]. The main features of the vehicle version are [58]:

- Frequency band: VHF/UHF;
- Frequency range: 30 MHz - 512 MHz;
- Simultaneous voice and data transmission in IPv4;
- Waveforms: standard waveforms in line with STANAGs, SOVERON and R&S SECOS;
- *Data rates* with SOVERON waveform: 110 Kbit/s, 630 Kbit/s and 2100 Kbit/s, depending on the type of SOVERON waveform.

The manpack version is distinguished by [59]:

- Distance covered: 7km using Mobile Ad Hoc Network (MANET);
- Frequency range: 30 MHz - 512 MHz;
- GPS: embedded;
- Waveforms: fixed frequency and SOVERON WAVE TNW.

## 2.5 UGV hardware components

This section analyses the hardware used in terms of UGV platforms in this thesis and the study in terms of onboard control units to be used in the UGV.

### 2.5.1 UGVs platforms

The leading development platform for this thesis was the LAFVIN Multi-Functional Smart Car, but communication and connection tests were also carried out on other platforms, such as Army & IST Vigilant.

#### LAFVIN Multi-Functional Smart Car

The vehicle belongs to a kit with sensors, line tracking and ultrasonic, and receivers for controlling the vehicle, such as Infrared and Bluetooth. The other components in the kit are Arduino UNO, V5 Expansion Board, 4 DC Motors, and the L298N Motor Board. The car already came with a manufacturer code for the Arduino [60].


Figure 2.7: LAFVIN Multi-Functional Smart Car [60].

#### Army & IST Vigilant

Some specific features of the Army & IST Vigilant, in terms of the component model, will not be explained as it is still a project under development. All this information has been obtained from the team developing Vigilant, as no academic work describing it has yet been published.

The Vigilant UGV is based on the Dr Robot Jaguar 4x4 wheeled UGV. The chassis and wheels were used, and the rest of the modules were replaced.

The main features are:

- Platform weight: 20 kg;
- Dimensions: 52x37x16 cm;
- Maximum payload: 20 kg carrying and 40 kg dragging;
- Top speed: 4.5 m/s (16 km/h).

The main components are:

- Autopilot;
- Onboard computer;
- Mobile network router;
- Switch;
- Telemetry radio;
- Image transmission system (at 2.4 GHz);
- Camera RGB;
- Thermal camera;
- LiDAR;
- Batteries;
- Motor board;
- GPS;
- Control command (radio at 2.4 GHz or cellular network).


Figure 2.8: Army & IST Vigilant UGV.

### 2.5.2 Development Boards for the Onboard Control Unit

This section covers the main development boards for embedded systems used in UGVs projects. When working with small vehicles, the onboard unit must be relatively small and light, with the minimum processing capacity for the tasks intrinsic in unmanned driving (processing and sending video, voice, and control commands). Table 2.4 compares the Arduino UNO R4 Wi-Fi and the Raspberry Pi 3 Model B+, which are two boards with similar computing power and price [61] [62]. Table 2.5 compares the NVIDIA Jetson Orin NX and Raspberry Pi 5, which have similar computing power and price. These boards were chosen because they are three recent models from different manufacturers (Arduino UNO R4 Wi-Fi, NVIDIA Jetson Orin NX, and Raspberry Pi 5) [63] [64]. The Raspberry Pi 3 Model B+ was chosen for comparison because it is an existing resource.

| Features | Arduino UNO R4 Wi-Fi | Raspberry Pi 3 Model B+ |
|---|---|---|
| Processor | Arm Cortex-M4, ESP32-S3 | Cortex-A53 |
| Processor Frequency | 48 MHz + up to 240 MHz | 1.4GHz |
| Architecture | 32-bit | 64-bit |
| RAM | 32 kB + 512 kB SRAM | 1 GB |
| Storage | 256 kB Flash + 384 kB | microSD card |
| Wireless communication | Wi-Fi 802.11 b/g/n, Bluetooth 5.0 | Wi-Fi 802.11ac, Bluetooth 4,2 |
| Ports | 6 Analogue input pins, 1 DAC, 6 PWM pins | 4x USB 2.0, HDMI, RJ45 |
| Operating System | Can be programmed with Arduino IDE | Raspbian (based on Debian) |

Table 2.4: Comparison of Arduino UNO R4 Wi-Fi and Raspberry Pi 3 Model B+ [61] [62].

| Features | NVIDIA Jetson Orin NX | Raspberry Pi 5 |
|---|---|---|
| Processor | 6-core Arm Cortex-A78AE v8.2 | Quad-core Arm Cortex-A76 |
| Processor Frequency | 2 GHz | 2.4GHz |
| Architecture | 64-bit | 64-bit |
| RAM | Up to 16 GB | Up to 8 GB |
| Storage | microSD card | microSD card |
| Wireless communication | Wi-Fi a/b/g/n/ac, Bluetooth 5.0 | Wi-Fi 802.11ac, Bluetooth 5.0 |
| Ports | 3x USB 3.2 Gen2, 3x USB 2.0, HDMI, RJ45 | 2x USB 3.0, 2x USB 2.0, 2x micro-HDMI, RJ45 |
| Operating System | Linux for Tegra | Raspbian (based on Debian) |

Table 2.5: Comparison of NVIDIA Jetson Orin NX and Raspberry Pi 5 [63] [64].

After analysing the boards in Table 2.4, it is clear that the Arduino UNO R4 Wi-Fi is unfavourable because it needs to be programmed and cannot work autonomously.

When comparing the Raspberry Pi 5 and the NVIDIA Jetson Orin NX, the choice favours the Raspberry model, as it is more widely used and can consequently find applications in similar jobs [65] [33].

## 2.6 Communication performance evaluation tools

iPerf is a widely used network testing tool designed to measure the maximum achievable bandwidth on IP networks using TCP, UDP, and Stream Control Transmission Protocol (SCTP) protocols. It operates on a simple client-server model and allows users to customise parameters, such as timing and buffers, to optimise network performance; typically, the command is only executed on the client when the server is already started. iPerf can evaluate bandwidth, delay jitter, and packet loss, supporting unidirectional and

bidirectional data transfers. The latest version, iPerf3, developed by ESnet/Lawrence Berkeley National Laboratory, is a complete rewrite, not backwards compatible with the original [66].

The Netperf is a parameter that can be used to measure various aspects of network performance. The main objectives are to evaluate the transfer of bulk data and the performance of requests using TCP or UDP. Netperf is based on an elementary client-server model with two executables: netperf and netserver. Generally, only netperf is executed, with netserver already started. The connection is used to send test configuration information and results to and from the remote system. Once the control connection has been established and the configuration information has been passed on, a data connection will be opened, which is used for the measurement. When the test is complete, the data connection will be disconnected, and the results from the netserver will be sent back via the control connection and combined with the result from the netperf for a display to the user [66].

In an application where the MAVLink communication protocol is used, the GCS software can also be a tool for evaluating network performance. MAVProxy, in particular, allows users to see the packet loss of MAVLink packets and the delay associated with sending them [42].

## 2.7 Related Work

The article “Unmanned Systems Interoperability in Military Maritime Operations: MAVLink to STANAG 4586 Bridge” presents a link between the MAVLink communication protocol and STANAG 4586 — which defines the architectures, protocols, data elements, and interfaces between the various control systems of an unmanned aerial system. To realise this link, converter software was developed on the UAV. This article aimed to enable UAVs that communicate with MAVLink, which is cheaper, to communicate with a GCS that uses STANAG 4586. The proposed system was developed using the Python programming language and a Raspberry Pi [67].

The experiments by the NATO IST-149-RTG group demonstrate interoperability between UGV and C2 systems. The goal was to test interoperability standards in real-world scenarios, specifically the JAUS and its interoperability profile. The experiments involved multiple NATO countries using different UGVs and GCS, demonstrating that systems could be easily extended to comply with these standards. No experiments were carried out with UAVs in this experiment due to technical problems. However, the integration of UAVs that used the MAVLink communication protocol was studied [68].

The report by the Estonian Defence Investment Centre [69] analyses eleven platforms from different companies/consortia from various countries. The report concludes that, in general, all the UGVs were able to drive in an open field with waypoints based on satellite navigation systems; the same was not true in rougher scenarios or with vegetation. Table 2.6 shows the companies/consortium, platform and country of origin.

| Company/Consortium | Platform | Country of Origin |
|---|---|---|
| ARX Landsysteme | GEREON RCS | Germany |
| charismaTec OG | ARTUS PTAone | Austria |
| Czech University of Defence/ VOP CZ, s.p. | TAROS V4 | Czech Republic |
| Diehl Defence | Ziesel | Germany |
| Hentschel System | MoSeS UGV | Germany |
| iMUGS | THeMIS 4.5 + iMUGS autonomy kit | Multinational consortium |
| LEM S.R.L | RANGER-POLARIS 570 | Italy |
| Milrem Robotics | THeMIS 4.5 + MIFIK | Estonia |
| Nexter-KNDS | ULTRO 600 | France |
| Rheinmetall Canada | Mission Master SP | Canada |
| Star Defence Logistics & Engineering | SR-0001 UGV 4x4 | Spain |

Table 2.6: List of companies and the platforms that participated in the Estonian trials [69].

The THeMIS UGV uses an IP radio at 2.4 GHz. It employs MIMO technology in a mesh network with 4W of power. This configuration allows a control range of up to 1.5 kilometres in line of sight [70].


Figure 2.9: THeMIS UGV [70].

The Gereon RCS UGV of ARX Landsysteme in terms of remote control and communication [71]:

- Mobile communication (5G backwards compatible);
- Wi-Fi;
- Radio frequencies of 698 — 960 MHz, 1710 — 2700 MHz, 3200 — 3800 MHz, 2.4 — 2.5 GHz and 5 — 6 GHz.

Figure 2.10: ARX Robotics GEREON RCS — Medium Autonomous Transportation Platform [71].

The Robotic Experimentation and Prototyping augmented by Maritime Unmanned Systems (REPMUS) is a Portuguese-led exercise focusing on UMV experimentation, capability development, and interoperability. Since 2014, the NATO Centre for Maritime Research and Experimentation has joined REPMUS. The live experimentation and test include [72]:

- Maritime interoperability experimentation, testing, and capability development based around a common UxV C2 network[1];
- Mine countermeasures;
- Dedicated anti-submarine warfare;

## 2.8 Summary

The most widely used video compression standard is H.264, which can be used with FFmpeg and GStreamer software. As GStreamer is compatible with GCS software, it is advantageous, allowing a pipeline to be established with Mission Planner.

As for Section 2.3, it is clear that Wi-Fi and Cellular network communication technologies are advantageous as they are widely used and allow a lot of flexibility. These are also suitable for transmissions that require a lot of bandwidth, such as sending video; however, with the increase in available bandwidth, there is a trade-off with the decrease in the range of these technologies.

As far as communication protocols are concerned, MAVLink is the most widely used at the civilian level and, as can be seen in Section 2.7, MAVLink converters have already been developed for a STANAG standard because the MAVLink communication protocol is widespread and allows the use of civilian UAV platforms.

[1] Experiments at REPMUS with private mobile cells concluded that the use of this technology is suitable for limited areas — for example, military bases — with a relatively small size, demonstrating a trade-off between capacity and range. These observations were made by the IST team present at the exercise.

The connection to the BMS must be made using the STANAG 5527 communication standard and the sending of the FFI XML can be done via IP-based communication technologies, such as Wi-Fi or the military radios presented in Section 2.4.3.

As can be seen, there are gaps in the literature, and no converter allows MAVLink to be translated into STANAG 5527 and consequently connected to the BMS/DSS. Another limitation is the existence of a communication library that allows communication with a commercial UGV without the specific use of autopilot hardware.

# Chapter 3

# Solution for integrating a commercial UGV into the Army Communications System

The reason for not using autopilot hardware on the LAFVIN UGV is that it is already being used on the VIGILANT, to whose experimentation on ARTEX24 this thesis has contributed. Therefore, also due to cost constraints, it was decided to only support real-time piloting with LAFVIN UGV, which was sufficient for the purpose of providing integration into the BMS.

Integrating a commercial UGV into the Army Communications System can be done in different ways. In the LAFVIN UGV, the MAVLink communication protocol was used for the UGV C2. The easiest way to use MAVLink is to employ autopilot hardware, such as the Pixhawk line, which does not require the use of onboard units like Arduino or Raspberry Pi and the development of communication code with the unmanned vehicle. A specific transmission system is used to send video to the GCS. This integration is represented in Figure 3.1, using Vigilant UGV.

Section 3.1 presents the solution developed in this thesis using a commercial UGV (LAFVIN), which originally did not communicate with MAVLink. To do this, the capacity to communicate with MAVLink had to be added.

Section 3.2 shows the system constitution in terms of software modules for the different types of architecture.

Section 3.3 describes the changes in the Arduino script and the Python scripts for connecting the LAFVIN UGV to the BMS. It also shows the steps in the GCS software, Mission Planner, to receive video and send control commands.

Finally, Section 3.4 presents the ARTEX 24 exercise, with the general description and objectives, as well as lessons learned and proposals for future improvement.

## 3.1 Solution Architecture

The solution architecture, with hardware autopilot, is presented in a configuration with Wi-Fi communication, represented in Figure 3.1. This architecture is represented using Vigilant, with all the components that make it up, which were described in Section 2.5.1. In general, this architecture was the ultimate goal of the ARmy Technological EXperimentation (ARTEX) 24 application.

The autopilot hardware connects to the UGV motor board, image transmission system, telemetry radio, GPS, and cameras. These components are connected via Universal Asynchronous Receiver-Transmitter (UART) ports. The connection to the LiDAR sensor is made via a Controller Area Network (CAN) port. The Autopilot ↔ GCS connection is designed to send the control commands to the autopilot and GPS and image to the GCS.

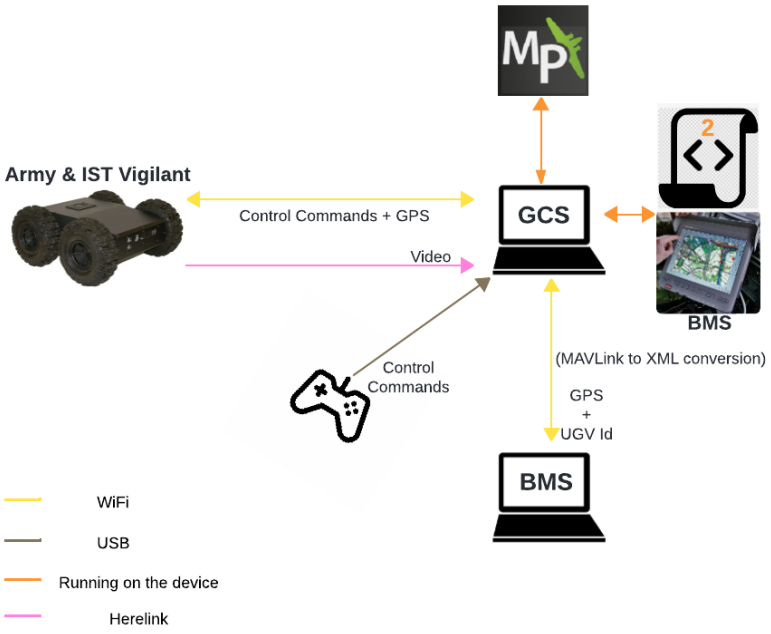


Figure 3.1: General architecture — with autopilot hardware.

For the communication without autopilot hardware, two architectures were used: one with Wi-Fi communication, presented in Figure 3.2 and the other using Combat-Net Radio (CNR), represented in Figure 3.3. The LAFVIN Multi-Functional Smart Car Kit was used for this implementation, representing the UGV and Arduino.

The onboard Raspberry Pi device connects to different components of the architecture. The onboard Raspberry Pi ↔ Arduino connection is designed to send the control commands received by the Raspberry Pi to the Arduino.

The Raspberry Pi ↔ GCS connection is designed to send the control commands to the Raspberry Pi and GPS and image to the GCS.

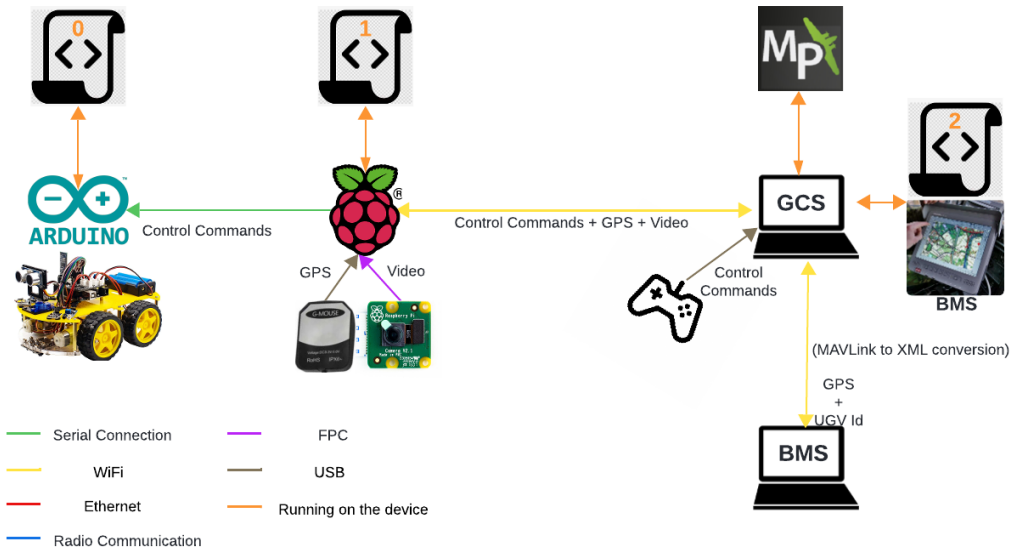


Figure 3.2: General architecture — Wi-Fi.

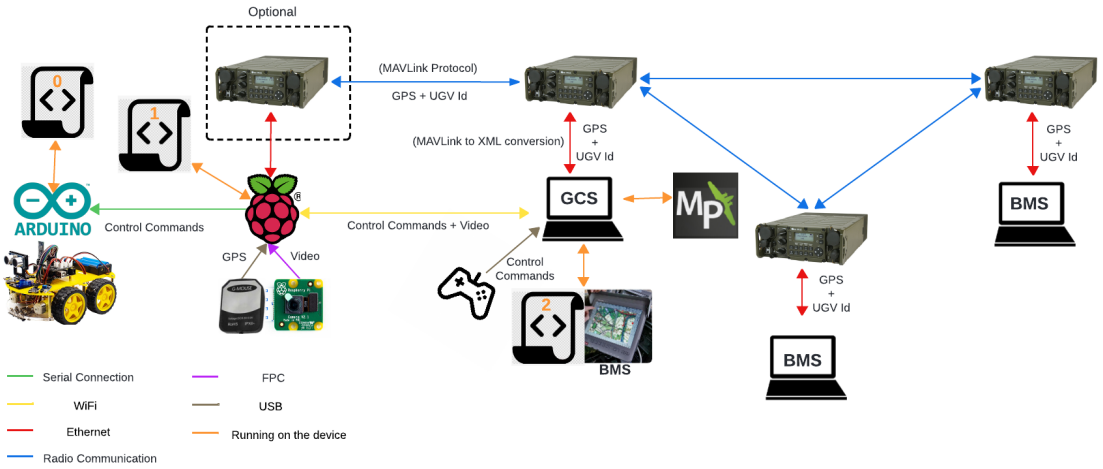


Figure 3.3: General architecture — Wi-Fi + Military Radios.

Figure 3.3 depicts the scenario with a radio connected to the UGV. This scenario makes sense when the UGV is large enough to carry a radio. In this specific application, given its size, the LAFVIN UGV is not used with a vehicular radio in moving tests. It can only be used in static tests or with small radios. In situations without a radio, all communications with the UGV are via Wi-Fi.

It would be much simpler if this architecture had autopilot hardware and a specific transmission system, like in Figure 3.1. In this case, Scripts 0 and 1 would not be necessary, as the hardware used is already designed for such applications. Connection to Mission Planner or other ground stations is automatic.

When communicating with CNR, the computer on the GCS and the Raspberry Pi connect to the radio via a network cable. The radios on both sides need to be connected to the antenna, which is made using a coaxial cable with an N connector. The used antenna will depend on the type of radio, manpack,

or vehicle version used in each scenario. On the GCS side, the gamepad connects to the computer via USB; on the UGV side, the GPS connects to the Raspberry Pi via USB. In distinction, the Raspberry Pi and the camera are connected using a Flexible Printed Circuit (FPC). The Arduino connects to the Raspberry Pi via a serial cable.

In Wi-Fi communication, all connections are used except the ones to the tactical radios since Wi-Fi is used for communication, and this is done via the network boards integrated into each device (GCS computer and Raspberry Pi).

### 3.1.1 GCS Architecture

The GCS MAVProxy software establishes a connection with the Raspberry Pi. The system GCS is a proxy server between the Raspberry Pi, Mission Planner, and BMS. The architecture of the GCS has to be like this so that data can be sent/received simultaneously to the BMS and the Mission Planner. This architecture, represented in Figure 3.4, also makes it possible to filter the messages sent to each of the software programs.

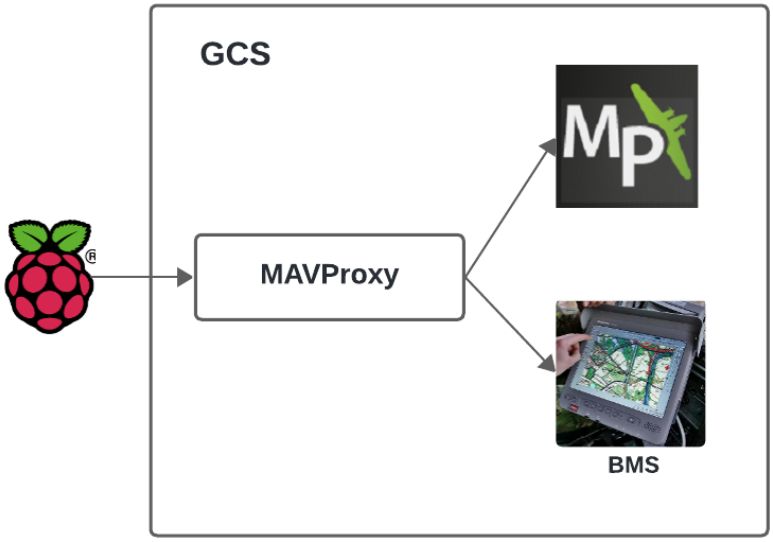


Figure 3.4: Architecture GCS.

The line of code used for Wi-Fi communication is: `mavproxy.py --master=udpout:[RPI_IP]:14550 --out=udp:127.0.0.1:14551 --out=udp:127.0.0.1:14552`.

In the case of communication with CNR, since there is a hybrid communication situation in which some data is sent via Wi-Fi and some via military radios, two connections were established between the Raspberry Pi and the MAVProxy running on the GCS. In this case, the line of code used is: `mavproxy.py --master=udpout:[RPI_IP_Wi-Fi]:14550 --master=udpout:[RPI_IP_eth0]:14550 --out=udp:127.0.0.1:14551 --out=udp:127.0.0.1:14552`.

### 3.1.2 Protocol Stack

The protocol stack for communication over Wi-Fi consists of a conventional TCP/IP stack. The transport layer will be TCP or UDP, and the Network layer will be IP; the Physical + Data Link layers correspond to Wi-Fi protocols.

Regarding communication over the P/PRC-525 and TR5000 tactical radios, represented by Figure 3.5, it should be noted that it has a routing module MANET module at the network layer. Like Wi-Fi, the transport layer is TCP/UDP, and the Network layer is IP. The Physical + Data Link layers correspond to protocols specific to radio.

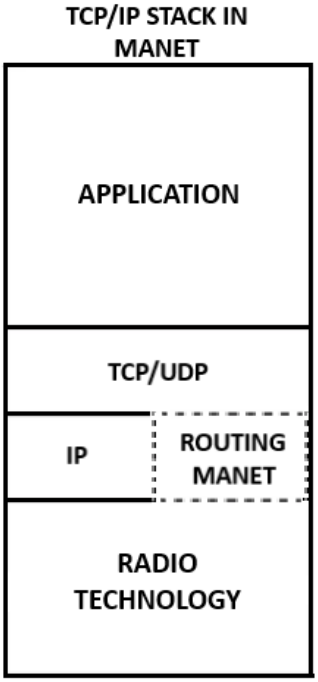


Figure 3.5: Protocol stack of a MANET network.

The protocol stack may change slightly depending on the radio used. Specific protocols may be implemented inside or outside the radio technology, depending on the radio model.

## 3.2 Hardware components

Since the Raspberry Pi 3 Model B+ is already a resource, its price is low, and it is already used in many identical implementations. Hence, the choice went to it, saving resources.

Since the aim was to control the vehicle via Mission Planner, the sensors of the LAFVIN kit were not used; only the Arduino UNO, V5 Expansion Board, 4 DC Motors, and the L298N Motor Board were used.

The UGV consists of the following components:

- Raspberry Pi 3 Model B+;

- LAFVIN kit;
- Battery;
- G-Mouse GPS VK-162;
- Raspberry Pi Camera Module 2;
- Radio.

The use of a military radio in the UGV is only for communication with CNR and depends on the weight and volume of the radio, taking into account the payload capacity of the UGV. The G-Mouse GPS VK-162 with a USB connection to the Raspberry Pi was adopted. The hardware used in the LAFVIN UGV for Wi-Fi communication is represented by Figure 3.6.

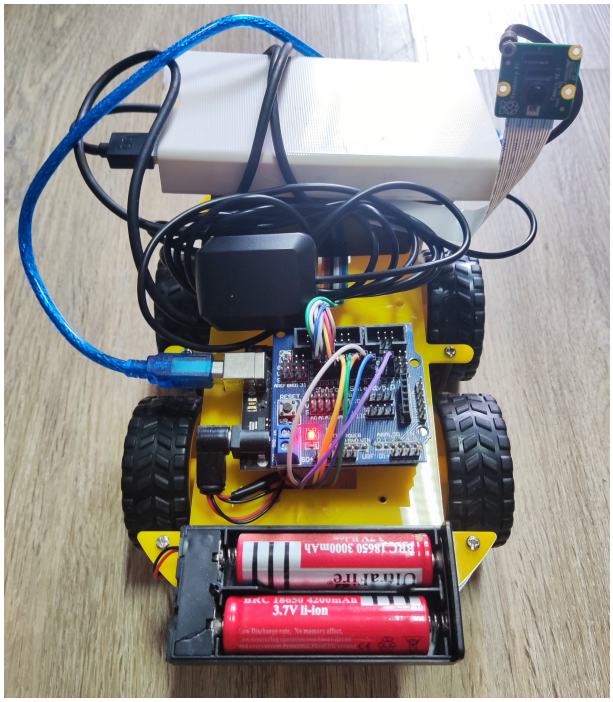


Figure 3.6: Hardware in LAFVIN UGV for Wi-Fi communication.

The GCS consists of the following components:

- Computer;
- Gembird Gamepad;
- Radio.

As with the UGV, military radios are used in the GCS only for communication with CNR. It is important to note that in this thesis, the camera used for transmission is from the same manufacturer as the onboard unit. This makes it easier to integrate the camera with the onboard unit; it is only necessary to configure the camera in the settings without the need for specific software.

## 3.3 Software Modules

The LAFVIN UGV manufacturer code, represented by Script 0 in the general architecture, had to be adapted as communication is now via a serial cable, and the manufacturer did not plan this mode of communication; it runs on the Arduino.

Script 1, shown in Figure 3.8, represents a link between the UGV and the ground station used to send the UGV position to the GCS. It also is used to send command controls to the UGV. This script runs on the Raspberry Pi.

Script 2, shown in Figure 3.9, runs on the GCS and represents a link to the BMS. It creates an XML, and it is sent to the BMS.

The delay block in sending heartbeat logic allows the control of the frequency of sending HEARTBEAT messages. The delay block, in send position logic, present in both scripts allows adjusting the frequency of position messages, GPS_RAW _INT in the case of Figure 3.8, and XML messages to the BMS with the identification of the UGV and its position, in the case of Figure 3.9. This frequency adjustment makes it possible to communicate in scenarios with less bandwidth available.

### 3.3.1 Script 0 — Arduino

In Arduino, the function `processRaspberryPiCommand(String command)` allows commands to be received via a serial connection to the Raspberry Pi. Figure 3.7 represents the commands received for the UGV to move in the corresponding direction.

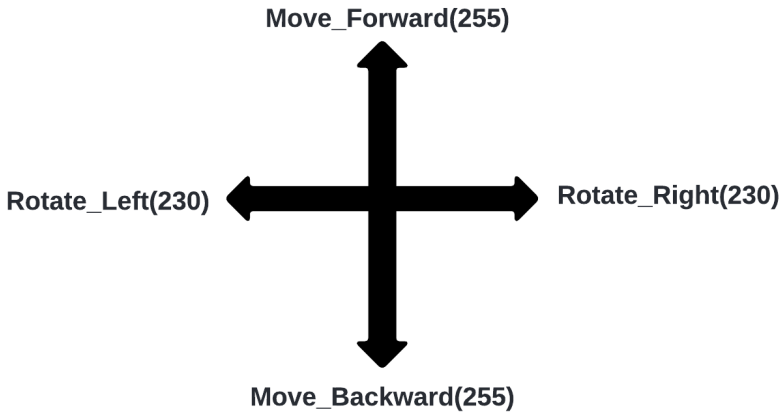


Figure 3.7: UGV move commands.

The manufacturer code did not include this communication mode. To avoid problems, all the other communication modes — Infrared and Bluetooth — were deactivated.

### 3.3.2 Script 1 — Management of the link between Raspberry Pi, Arduino, and GCS

In the case of communication using military radios, it is necessary to initiate two connections to MAVProxy, one for the radios and the other for Wi-Fi, since both are used. The Raspberry Pi $\leftrightarrow$ Arduino connection is used to send the control commands to the Arduino. The main functions are:

- Start of connection with Arduino: `arduino_controller.connect()` — it is represented by the Arduino Connection block in Figure 3.8;

- Sending control commands to Arduino: `arduino_controller.send_command(command)` — it is represented by the Send Command to Arduino block in Figure 3.8.

The main functions in the Raspberry Pi $\leftrightarrow$ GCS connection are:

- Establish connection with MAVProxy: `mavlink_connection()` — it is represented by the MAVProxy Connection block in Figure 3.8;

- Maintain connection with Mission Planner: `send_heartbeat()` — it is represented by the Send Heartbeat block in Figure 3.8;

- Handle control commands from GCS: `send_command_to_arduino(command)` — it is represented by the Handle Commands block in Figure 3.8;

- Get position data from GPS: `get_gps_data()` — it is represented by the Get GPS data block in Figure 3.8;

- Send UGV position: `send_position(latitude,longitude, altitude)` — it is represented by the Send Position block in Figure 3.8.

All commands between the GCS and the Raspberry Pi are sent via Wi-Fi in the configuration using Wi-Fi. In the case of the configuration using tactical radios, the heartbeats are sent by both communication channels so that the availability of both can be seen. The UGV position, through the `send position(latitude, longitude, altitude)` function, is only sent using tactical radios. This is the most critical communication component since it displays the UGV position. By using military radios, it is possible to exploit the encryption they offer and thus make communication more secure.

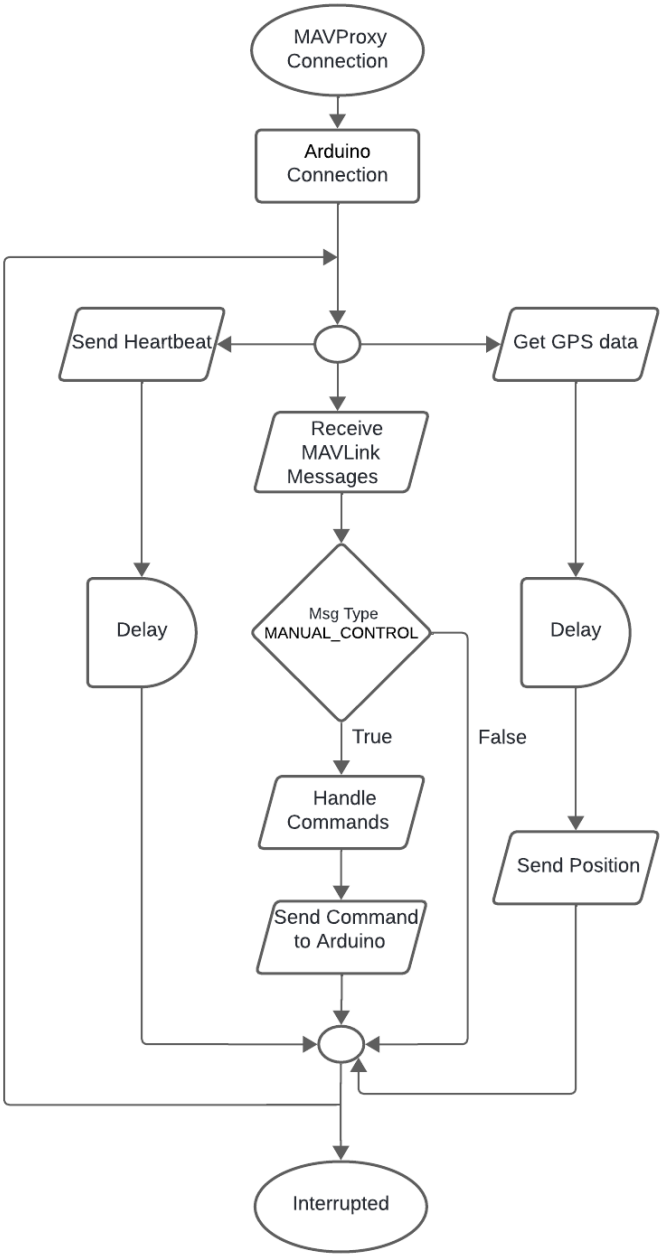


Figure 3.8: Script 1 — Management of the link between Raspberry Pi, Arduino, and GCS.

### GPS

To get the UGV position, the “gpsd”, “gpsd-clients”, and “python-gps” libraries were used to obtain the GPS data using the `get_gps_data()` function. These libraries had to be installed on Raspberry Pi. The initial step was to start the session, and then the latitude, longitude, and altitude variables were updated in real-time.

Once it had the variables to update, it created the `send_position(latitude, longitude, altitude)` function. This function sends MAVLink GPS_RAW_INT messages with latitude, longitude, and altitude data to the Mission Planner.

#### Control commands

The MAVLink MANUAL_CONTROL messages are received by the Raspberry Pi using the `handle_manual_control` function. Depending on the parameters received, the corresponding message is sent to the Arduino to operate the UGV:

- Front: `send_command_to_arduino(%F#);`
- Back: `send_command_to_arduino(%B#);`
- Right: `send_command_to_arduino(%R#);`
- Left: `send_command_to_arduino(%L#).`

### 3.3.3 Script 2 — Management of the link between GCS and BMS

This script aims to extract the UGV position from MAVLink messages and generate an XML according to STANAG 5527. The principal functions of the script running on the GCS are:

- Establish connection with MAVProxy: `mavlink_connection()` — it is represented by the MAVProxy Connection block in Figure 3.9;
- Extracts position from the GPS_RAW_INT MAVLink message: `message.get_position_data(msg)` — it is represented by the Get Position block in Figure 3.9;
- Generate XML with UGV position: `create_bms_xml(latitude, longitude)` — it is represented by the Create XML block in Figure 3.9;
- Send XML to BMS: `send_xml_file(filename, server_ip, server_port)` — it is represented by the Send XML block in Figure 3.9.

The connection is established with port 14552, where MAVLink messages are forwarded via MAVProxy. This script can run on GCS if MAVProxy is forwarding messages to port 14552 on localhost or on another machine.

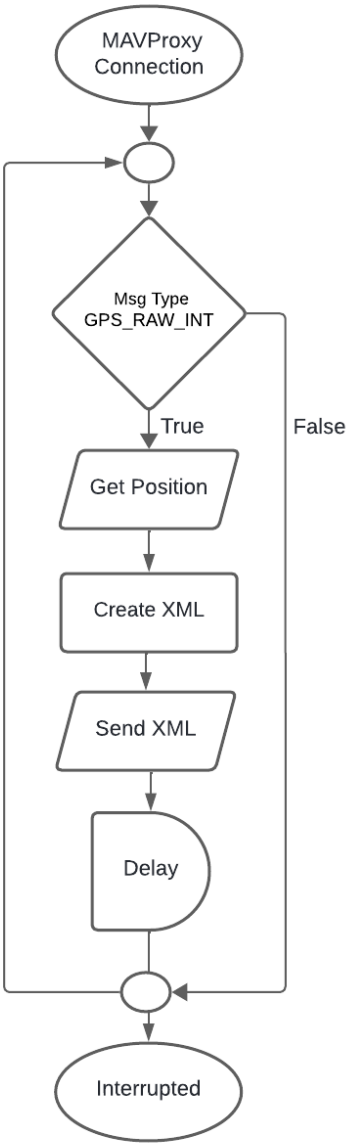


Figure 3.9: Script 2 — Management of the link between GCS and BMS.

The used XML model is the A2 of the STANAG 5527. Depending on where the BMS is running, the XML must be sent there. In a configuration where the BMS is running on the GCS, it is enough to send it to localhost; otherwise, it requires the IP of a machine on the network that is running the BMS. In the tactical radio scenario, this sending is done via military radios. The IP to which the XML is sent must be configured in the function `send_xml_file(filename, server_ip, server_port)` used to send the file.

In the BMS, in addition to the standard configuration, which includes creating a mission with some mandatory parameters, such as the configuration of the stack and GPS, a gateway must be configured in the BMS with the port where the report will be received, in this case, port 48882, and the message template which will be used, in this case, FFT Systems—STANAG 5527 A1/A2. Figure 3.10 represents the general structure of the FFI XML message.


Figure 3.10: LAFVIN UGV FFI XML message.

### 3.3.4 Mission Planner Configuration

This subsection presents the main aspects to be configured in the Mission Planner, namely, receiving the video and sending the control commands. The Mission Planner is configured on port 14551. Once the software is open, after running Script 1 and then starting GCS with MAVProxy, a UDP connection is automatically started on port 14551.

First, the configuration of video reception on the Mission Planner and its capture and sending on the Raspberry Pi is presented using the Raspivid and GStreamer libraries, respectively. Then, in this section, the configuration of the joystick in Mission Planner and the configurations to send control commands are presented.

#### Video streaming

Image capture was first initiated using the Raspivid application on the Raspberry Pi. Then, the GStreamer application was used to send the video stream to the Mission Planner, running on the GCS. These were the commands used on each of the devices:

- Mission Planner: `gst-launch-1.0 udpsrc port=5600 caps="application/x-rtp, media=video, clock-rate=90000, encoding-name=H264, payload=96" ! rtph264depay ! avdec_h264 ! videoconvert ! autovideosink;`
- Raspberry Pi: `raspivid -n -w 460 -h 460 -b 1000000 -fps 30 -t 0 -o - | gst-launch-1.0 -v fdsrc ! h264parse ! rtph264pay config-interval=10 pt=96 ! udpsink host=[GCS_IP] port=5600.`

The quality of image capture has an influence on the bandwidth required for communication. The quality of the image is adjusted according to the characteristics of the transmission channel, which will be analysed in the results.

### Control commands

To send control commands to the UGV, the Gembird Gamepad was first connected to the GCS and then paired with the Mission Planner. When connected to the Mission Planner, two axes of movement were used, allowing the UGV to move in four different directions. The axes were chosen automatically by the Mission Planner, considering the movement of the controller's analog, selected by the user. Figure 3.11 represents the configuration of the Mission Planner software.

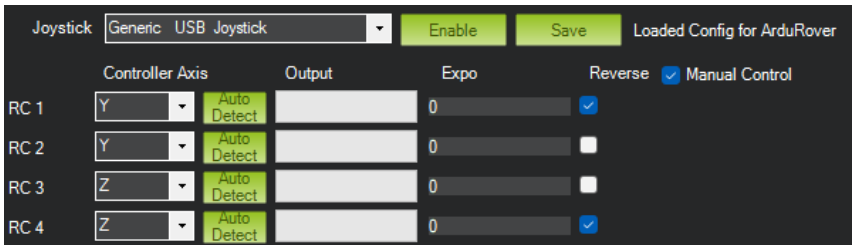


Figure 3.11: Joystick connection in the Mission Planner.

For movements in different directions on the same axis, the "Reverse" option was used. The mode chosen was "Manual Control", which will be important in the type of MAVLink messages generated at the GCS. Figure 3.12 represents the axis of the movement on the gamepad.


Figure 3.12: Controller Axis.

The MAVLink message generated for each direction is:

- Front: MANUAL_CONTROL(1000, -1000, 0, 0);
- Back: MANUAL_CONTROL(-1000, 1000, 0, 0);
- Right: MANUAL_CONTROL(0, 0, 1000, -1000);
- Left: MANUAL_CONTROL(0, 0, -1000, 1000).

In both configurations, the control commands are sent from the GCS to the Raspberry Pi via Wi-Fi, as it is necessary to have as little delay as possible. This mode of communication, established by the Mission Planner, is demanding in terms of bandwidth.

## 3.4 ARTEX 24

The ARTEX is an exercise organised by *Centro de Experimentação e Modernização Tecnológica do Exército* (CEMTEx), which contributes to research and development activities and, simultaneously, to boosting the defence economy at the national level.

The ARTEX 24 went from 17th June to 5th July 2024 and focused on technological experimentation. As far as the UxV exercise is concerned, it involves two forces: Red and Blue. The final ambition of CEMTEx was to represent the different UxV in the BMS by developing a script that would subscribe to topics on the MQTT server and translate them from GeoJSON into XML, which would then be represented in the BMS. The exercise organisers could not develop this integration in time, so it could not be described in the BMS. Google Earth Pro was then used to represent the COP. Figure 3.13 shows a simplified representation of the general architecture of a Blue force.

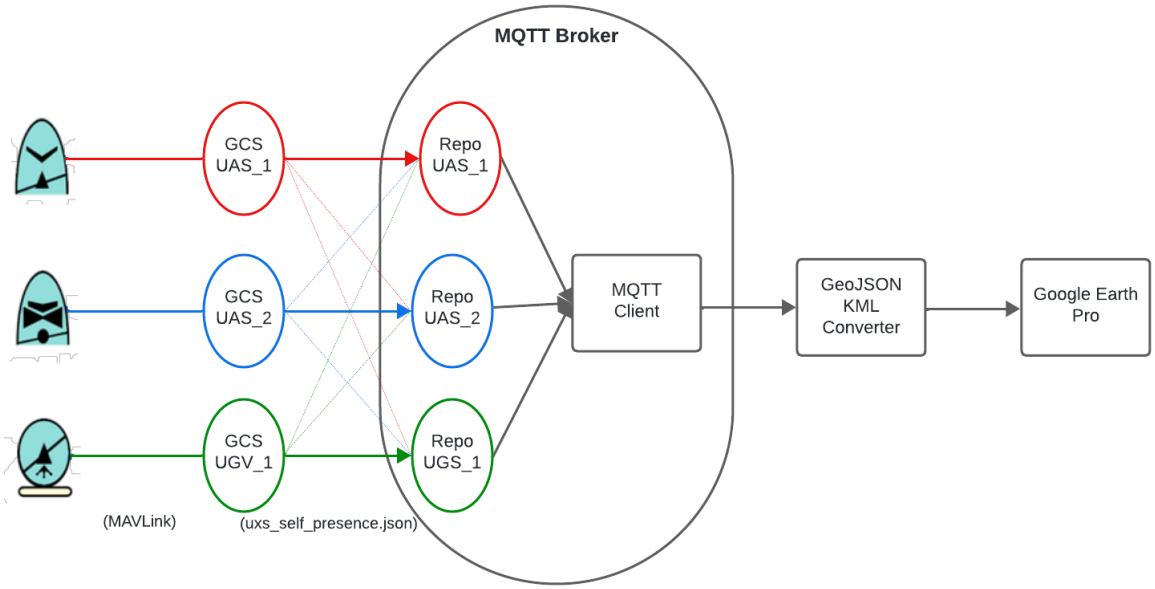


Figure 3.13: Artex 24 simplified architecture.

### 3.4.1 Integration of IST UxVs

The IST team was present with two UxVs: UAV Eye In The Sky and UGV Vigilant. As part of the IST team for ARTEX 24, two scripts were developed:

- Representing their own position by sending `uxs_self_presence.json` messages in Figure 3.14.

- Represent other vehicles in the Mission Planner, belonging to the same force by receiving `uxs_self_presence.json` messages represented by Figure 3.15;

To ensure that the messages arrive at the MQTT server in GeoJSON format, the script shown in Figure 3.14 was developed to convert MAVLink messages into GeoJSON. This allows other UxVs to know the position of other UxVs or sensors belonging to the same force.

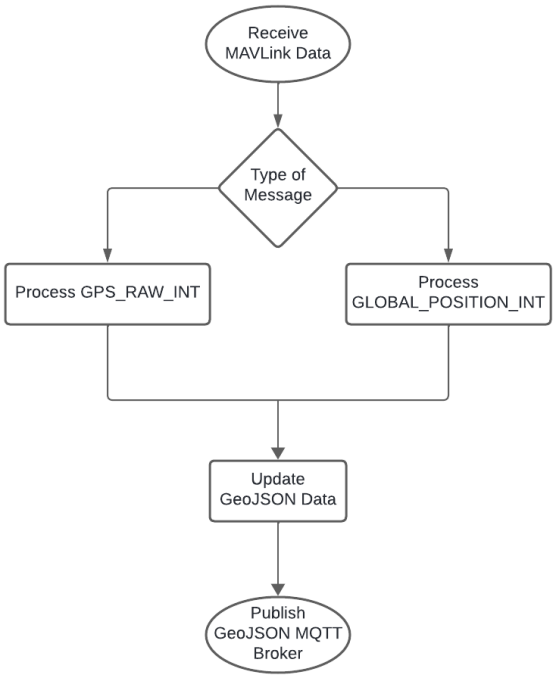


Figure 3.14: Message sending logic.

The second script shown in Figure 3.15 allows centralising the other UxVs positions in Mission Planner. This was used in both UxVs of the IST team — UGV Vigilant and UAV Eye In the Sky. The messages with the position were subscribed to considering the force to which each vehicle belonged.

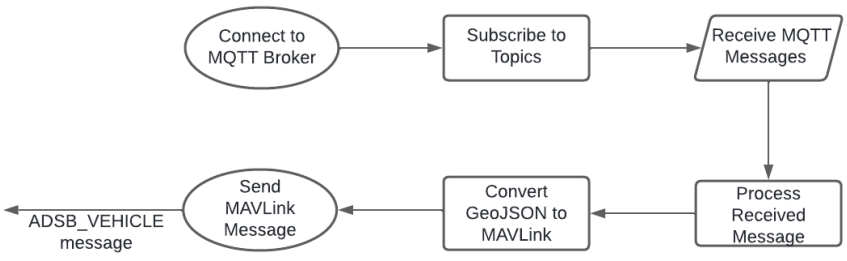


Figure 3.15: Mission Planner reception logic.

These scripts aim to provide a link to Mission Planner, allow vehicles to share positions, and connect to the MQTT server. A GeoJSON format has been defined by CEMTEx to make the publication of messages on the MQTT server uniform.

Message reception and subsequent representation in Mission Planner was done by converting the `uxs_self_presence.json` messages into MAVLink ADSB_VEHICLE messages.

Sending `uxs_self_presence.json` messages consists of translating MAVLink messages: GPS_RAW_INT or GLOBAL_POSITION_INT, into GeoJSON messages.

### 3.4.2 Lessons learned and suggestions for future improvement

In response to some ARTEX 24 constraints regarding the connection to the BMS, changes were made to the scripts used in the exercise to fill the gaps.

In the message-sending logic, after publishing on the GeoJSON MQTT broker, these two functions were added:

- Generate XML with vehicles position: `create_bms_xml(latitude, longitude, transp_id);`
- Send XML to BMS: `send_xml_file(filename, server_ip, server_port).`

The value of transp_id was fixed since this was the uxs_name of our vehicle, in the case of the UGV, Vigilant. The latitude and longitude were retrieved from the GPS_RAW_INT or GLOBAL_ POSITION_INT message.

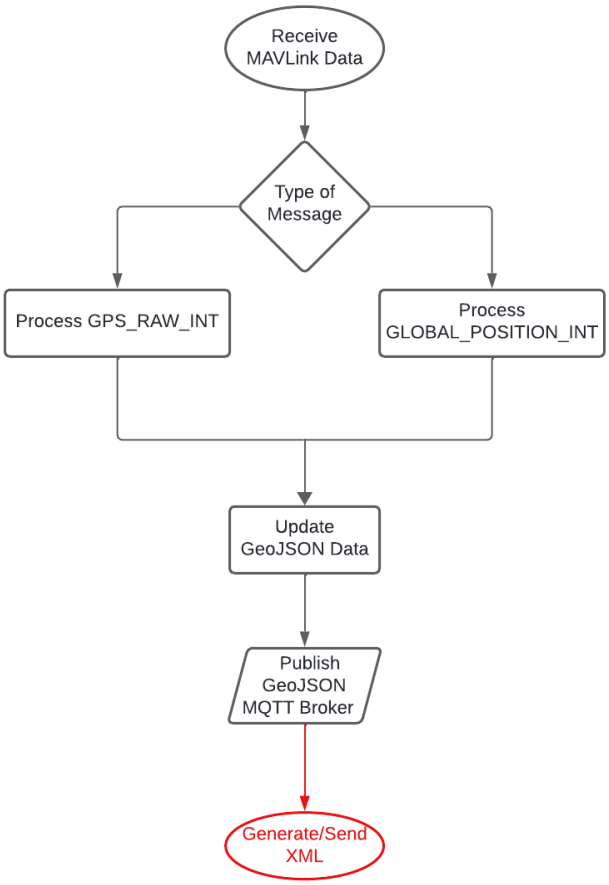


Figure 3.16: Logic for sending messages with sending to the BMS.

In the reception logic, after sending the ADSB_VEHICLE message, the latitude, longitude, and the transp_id, which represent the uxs_name, were retrieved from the GeoJSON. Then, the XML was filled with the updated values and sent to the BMS. The added functions are the same as the sending logic.

The Figures 3.16 and 3.17 represent the scripts used in the ARTEX 24, with the update to send to the BMS. In red, there is the additional part that provides this functionality.

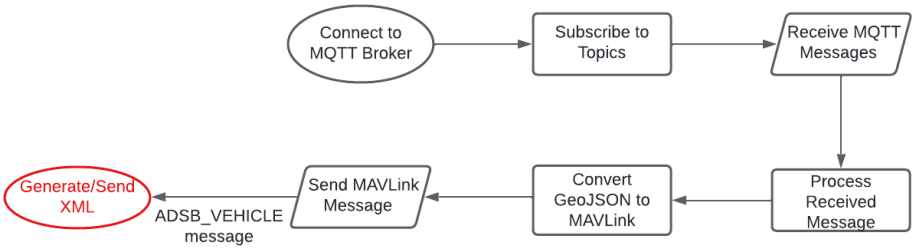


Figure 3.17: Mission Planner reception logic with sending to the BMS.

Since these developments were made after ARTEX 24 and the MQTT broker had already been closed down, Eclipse Mosquitto was used instead. This open-source message broker made it possible to test the operation of this BMS connection configuration.

The logic of sending messages was tested using LAFVIN UGV. To do this, the UGV was connected to GCS software, in this instance, Mission Planner. It was confirmed that position messages were being produced — in this case, GPS_RAW_INT — and then confirmed in the BMS that the UGV position had been received.

The reception logic was also tested using the LAFVIN UGV and the Mosquitto broker. In this case, the messages produced by LAFVIN UGV were considered to be messages from Vigilant UGV. Then, assuming the constitution of the force to which the UGV belongs, vehicle messages were forged in the corresponding topics so that they could be subscribed to the MQTT broker and then sent to the BMS. This made it possible to confirm the functionality of the reception logic developed for exercises such as ARTEX.

## 3.5 Summary

Two architectures were studied: with autopilot hardware and without autopilot hardware. It became evident that the use of autopilot hardware allows for easier integration without the need to use Scripts 0 and 1.

The architecture without autopilot hardware describes all the changes to the existing LAFVIN UGV platform, the software modules developed, and the integration steps with the GCS. The GCS demonstrated the connection to the Mission Planner using Script 1 and the connection to the BMS using Script 2 and the XML model A2 of the STANAG 5527.

In ARTEX 24, some knowledge was learned about UxVs that work with MAVLink and code was developed that allowed the Vigilant UGV and Eye In The Sky UAV to be connected to Google Earth Pro.

They were then connected to the BMS, filling a gap in the exercise and developing a code library for future exercises.

# Chapter 4

# Results

This Chapter presents the results of the experiments carried out in *Paço de Arcos*, where the SIC-T project team is located, using the configuration presented with the LAFVIN UGV. Firstly, some issues regarding the interoperability between UxVs and BMS are presented. Subsequently, network architectures are described that allow some test configurations to be carried out using Wi-Fi and CNR, and finally, the solution found is evaluated using network evaluation metrics.

## 4.1 Interoperability Challenges

The problem was that there was no documentation for connecting a commercial UGV to the BMS by sending the position in an XML file. Until now, this method of sending had not been used since tactical vehicles used predefined GPS models, which were already associated with the BMS; on the one hand, this configuration makes the connection easier since the pairing work was done by the BMS developer (Critical Software), but on the other hand, it ends up being more restrictive, since it reduces the options of GPS models for pairing with the BMS.

Another problem was the lack of information and literature on sending data in real-time via tactical radios. This made it difficult to assess and choose what messages could be sent via tactical radios in an unmanned driving configuration. Different configurations were also evaluated and studied, recognising communication limitations via military radios.

## 4.2 Baseline Solution

The solutions for the tests in the SIC-T project team will be presented. To achieve this, tests were carried out using Wi-Fi technology alone and a hybrid configuration using tactical radios and Wi-Fi. The tactical radios fill an existing gap in Wi-Fi communication — security — since the UGV position is considered critical data.

Generally speaking, the solution is to generate and send the XML files containing the UGV position to the BMS, this was executed using Script 2 and can be achieved using Wi-Fi, Military Radios or other

IP-based radio communication technologies.

The delay blocks in the Figures 3.8 and 3.9 were fixed for all the tests as follows:

- HEARTBEAT message (63 Bytes): 1 second;
- GPS_RAW_INT message (101 Bytes): 10 seconds;
- Send XML (3.3 KBytes): 10 seconds;

All tests were carried out with the following parameters in GStreamer:

- Video image width: 460 pixels;
- Video image height: 460 pixels;
- Total size of each video image: 460 x 460 = 211600 pixels;
- Bitrate: 1 Mbit/s;
- Frames per second: 30 fps.

The Wi-Fi tests were divided into two scenarios: one using existing network infrastructure and the other using the RAD-MAVLink-Bridge architecture, which transforms the Raspberry Pi into a wireless access point. The military radio tests were divided into two scenarios: one exclusively using P/PRC-525 Tactical Radios and the other using TR5000H radios.

The computer representing the GCS has Windows 11, a Ryzen 7 3700U processor, and 16 GBytes of RAM.

The computer representing the BMS is the Getac used by the Portuguese Army, which has Windows 10 Pro, an Intel Core i5-6300U processor, and 8 GBytes of RAM.

### 4.2.1 Scenario 1: Wi-Fi

All the data was sent via Wi-Fi in the Wi-Fi tests shown in Figure 4.1. The only requirement is that the GCS and the Raspberry Pi be on the same network. The Wi-Fi protocol used in the Wi-Fi tests was Wi-Fi 4 (802.11n) at a frequency of 2.4 GHz.

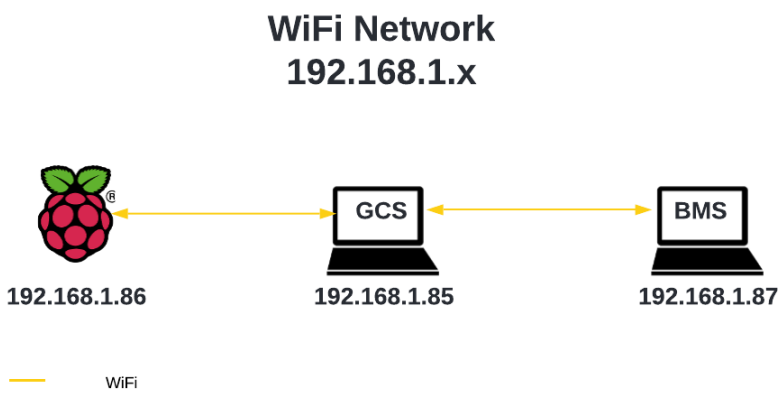


Figure 4.1: Network Architecture — Wi-Fi.

The BMS could be running on GCS by sending the XML, by TCP, to localhost. In either case, Script 2 is required.

### 4.2.2 Scenario 2: Wi-Fi-RAD-MAVLink-Bridge

The Figure 4.2 shows the Wi-Fi-RAD-MAVLink-Bridge architecture. The RAD-MAVLink-Bridge script turns the Raspberry Pi into a wireless access point. A Wi-Fi network (rad-bridge) connects the GCS to this network and sends the necessary commands. The Wi-Fi protocol used in the Wi-Fi-RAD-MAVLink-Bridge tests was Wi-Fi 4 (802.11n) at a frequency of 2.4 GHz.

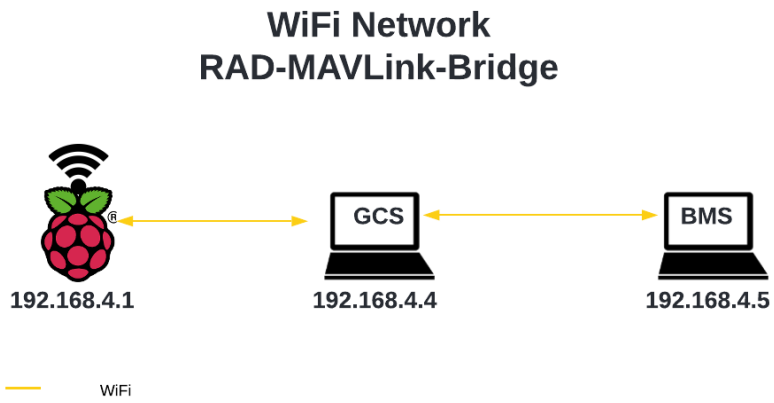


Figure 4.2: Network Architecture — Wi-Fi-RAD-MAVLink-Bridge.

The BMS could be running on GCS by sending the XML, by TCP, to localhost. In either case, Script 2 is required.

### 4.2.3 Scenario 3: Military Radios — P/PRC-525

In the configuration with P/PRC-525 Tactical Radio, VHF (SECOM-V 102-104 MHz) radio communication in IPoA was used to send the HEARTBEAT and GPS_RAW_INT messages from the Raspberry Pi to the GCS and the XML with the UGV position to the BMS; it is represented in Figure 4.3. All the other data (control commands and video) was sent via Wi-Fi, as in Figure 4.1.

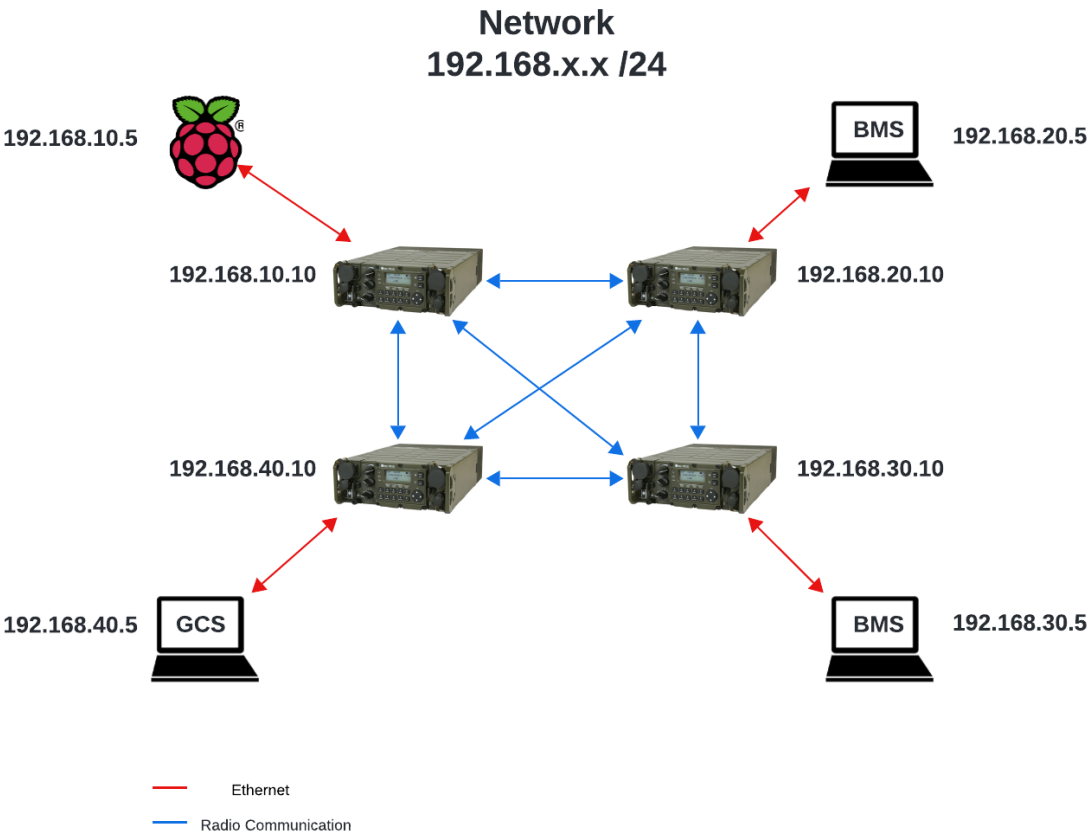


Figure 4.3: Network Architecture — P/PRC-525.

The choice of this scenario with the P/PRC-525 is because these are the radios of most significant use in the Portuguese Army. This scenario can be transported to a situation where there are two manned tactical vehicles, a UGV and a vehicle that symbolises the command vehicle (GCS).

Given the size of LAFVIN UGV, this configuration only makes sense in a static situation. In a dynamic situation, data from the UGV to the GCS would only be sent via Wi-Fi.

### 4.2.4 Scenario 4: Military Radios — TR5000H

In the configuration with TR5000H Tactical Radio, VHF (225-228 MHz) radio communication in IPoA with continuous phase modulation in balanced mode. It was used to send the HEARTBEAT and GPS_RAW_INT messages from the Raspberry Pi to the GCS and the XML with the UGV position to the BMS; it is represented in Figure 4.4. All the other data (control commands and video) was sent via Wi-Fi, as in Figure 4.1.

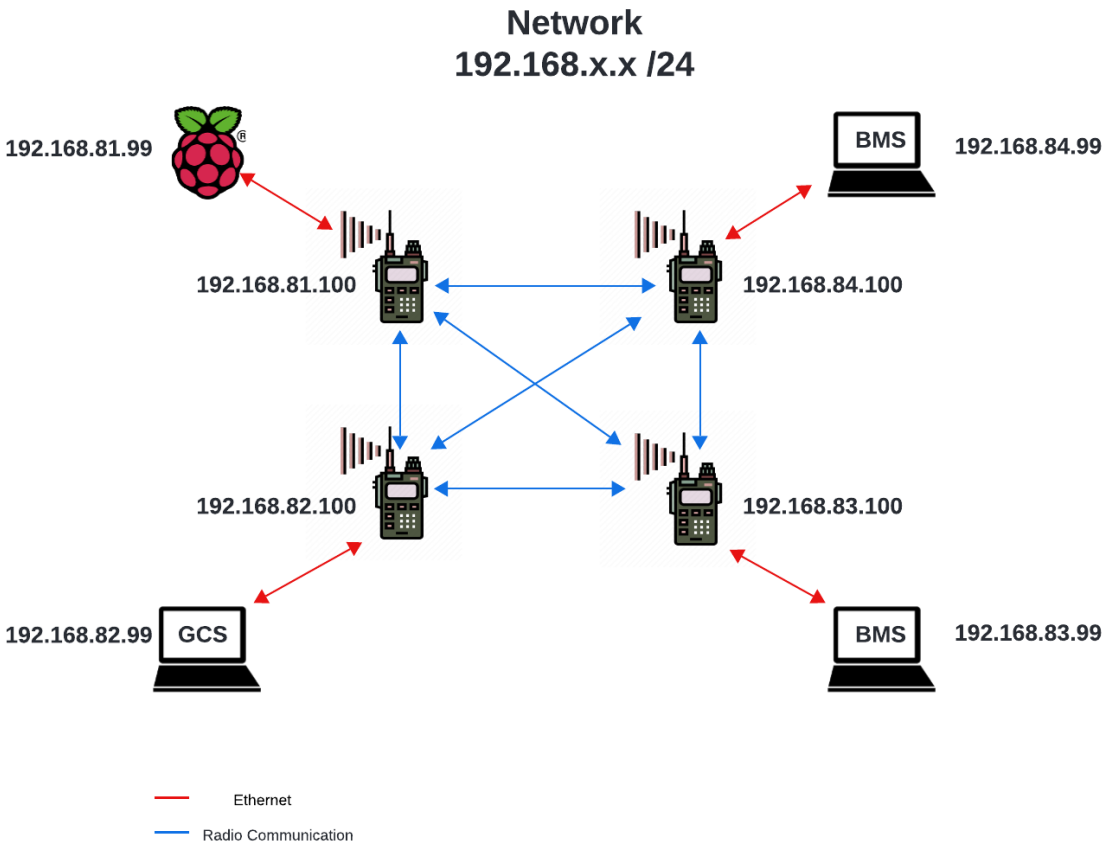


Figure 4.4: Network Architecture — TR5000H.

This architecture was chosen because the TR5000 is the radio most recently acquired by the Portuguese Army. Because there is a manpack version, it can be integrated into smaller UGVs, such as the Vigilant UGV.

## 4.3 Solution Evaluation

The network performance results were determined using the iPerf3 software in a configuration where the Raspberry Pi was the server (sender), and the GCS was the client (receiver). In the case of Wi-Fi tests, the iPerf3 commands were executed in a scenario where communication was fully established, representing an additional throughput consumption already existing by the UGV C2.

The tests with military radios were carried out in two scenarios: one with no data to be transmitted on the communication channel and the other with HEARTBEAT, GPS_RAW_INT, and XML messages to be sent. Given that the data being transmitted was more easily calculable — since no video or MAVLink messages were being transmitted, which by default are sent from the GCS to the UGV via this channel — the size of the iPerf3 command packets was adapted to simulate the transmission that takes place with them. Another adaptation in this scenario was to carry out UDP and TCP tests simultaneously.

Each iPerf test was conducted for approximately 60 seconds, covering the following scenarios:

1. Measurement of throughput using TCP;

2. Bidirectional TCP bandwidth testing;
3. UDP testing for jitter and lost data analysis[1].

In the context of network testing:

- GCS → RPI (TX): Indicates the GCS sending data, focusing on TX performance. It is equivalent to Transmitter Client (TX-C) Sender mode on iPerf3;
- GCS → RPI (RX): Refers to the GCS sending data in bidirectional communication, focussing on RX performance. It is equivalent to TX-C Receiver mode on iPerf3;
- RPI → GCS (TX): Indicates the Raspberry Pi sending data in this configuration, evaluating TX performance. It is equivalent to Receiver Client (RX-C) Sender mode on iPerf3;
- RPI → GCS (RX): Refers to the Raspberry Pi sending data, measuring RX efficiency. It is equivalent to RX-C Receiver mode on iPerf3.

The main difference lies in how each function focuses on transmitting or receiving data, whether in TX-C or RX-C configuration.

The GCS software MAVProxy and Wireshark were used to analyse the number of MAVLink messages sent in a 60-second communication and the size of each message.

### 4.3.1 Scenario 1: Wi-Fi

The Wi-Fi network used was, at the moment, running at speeds of 57.59 Mbit/s of download and 10.54 Mbit/s of upload, measured using the tool (www.speedtest.net). In this situation, range and speed depend on the Wi-Fi network infrastructure. The static scenario represents a configuration where the UGV, GCS, and Wi-Fi router are next to each other, roughly 1 meter apart. In the case of the movement tests, a scenario was tested in which the GCS was stationary, and the UGV was in motion.

The result obtained in the BMS with the configuration via Wi-Fi is shown in Figure 4.5. As there is no radio associated with the GCS, only the UGV is displayed.

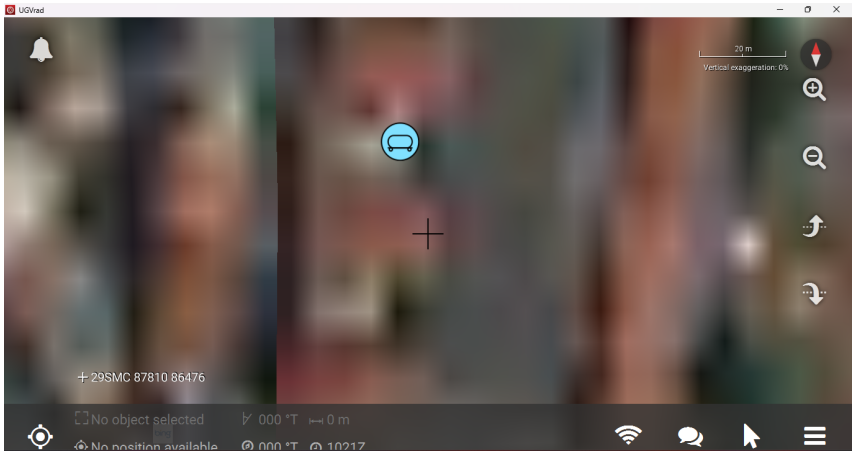


Figure 4.5: BMS — Wi-Fi results.

[1]Appendix B shows loss datagram data that can be related to the loss data. In most cases, the values are very similar, but in some situations, there are some differences that may come from the iPerf3 tool.

Table 4.1 shows throughput measures using TCP. The results show an equal transfer size for sending and receiving data, indicating a symmetrical and stable throughput. In static tests, both devices (Raspberry Pi and GCS) transferred 139 MBytes of data; the bitrate for GCS was 19.1 Mbit/s. In the movement scenario, the amount of data transferred was reduced to 74.4 MBytes, with a corresponding decrease in bitrate to 10.3 Mbit/s for GCS. The movement significantly impacted the throughput, resulting in about half the transfer rate compared to the static scenario.

| Role | Interval | Transfer | Bitrate |
|---|---|---|---|
| **With communication (Static)** | | | |
| **TX** | 0.00-60.01 sec | 139 MBytes | |
| **RX** | 0.00-60.85 sec | 139 MBytes | 19.1 Mbit/s |
| **With communication (Movement)** | | | |
| **TX** | 0.00-60.01 sec | 74.4 MBytes | |
| **RX** | 0.00-60.86 sec | 74.4 MBytes | 10.3 Mbit/s |

Table 4.1: Measuring throughput with TCP — Wi-Fi.

As with TCP unidirectional tests, almost all the data sent is received in the static situation. Table 4.2 shows that the GCS has a greater capacity for sending TCP packets than the Raspberry Pi. On the movement tests, it is again clear that the bitrate drops by about half, requiring 3 retransmissions.

| Role | Direction | Interval | Transfer | Bitrate | Retr |
|---|---|---|---|---|---|
| **With communication (Static)** | | | | | |
| **TX** | GCS → RPI | 0.00-60.00 sec | 93.2 MBytes | | 0 |
| **RX** | GCS → RPI | 0.00-60.74 sec | 90.8 MBytes | 12.5 Mbit/s | 0 |
| **TX** | RPI → GCS | 0.00-60.00 sec | 23.0 MBytes | | 1 |
| **RX** | RPI → GCS | 0.00-60.74 sec | 22.2 MBytes | 3.07 Mbit/s | 0 |
| **With communication (Movement)** | | | | | |
| **TX** | GCS → RPI | 0.00-60.01 sec | 51.0 MBytes | | 0 |
| **RX** | GCS → RPI | 0.00-62.23 sec | 50.4 MBytes | 6.90 Mbit/s | 0 |
| **TX** | RPI → GCS | 0.00-60.01 sec | 12.9 MBytes | | 3 |
| **RX** | RPI → GCS | 0.00-62.23 sec | 12.1 MBytes | 1.63 Mbit/s | 0 |

Table 4.2: Bidirectional TCP mode for measuring throughput — Wi-Fi.

The tests on Table 4.3 show, in static tests, the Raspberry Pi and GCS are very similar in terms of data transfer, with Raspberry Pi transferring 71.5 MBytes and GCS receiving 68.8 MBytes at 9.61 Mbit/s. The jitter in GCS is 0.737 ms, and the data loss for GCS is (4%).

In the movement scenario, GCS suffered a significant performance drop, receiving only 40.9 MBytes at 5.14 Mbit/s. The jitter increased to 3.183 ms, and data loss reached 42%, highlighting the severe impact of motion on UDP performance in Wi-Fi communication.

| Role | Interval | Transfer | Bitrate | Jitter | Lost Data |
|---|---|---|---|---|---|
| **With communication (Static)** | | | | | |
| **TX** | 0.00-60.01 sec | 71.5 MBytes | | | |
| **RX** | 0.00-60.06 sec | 68.8 MBytes | 9.61 Mbit/s | 0.737 ms | 4% |
| **With communication (Movement)** | | | | | |
| **TX** | 0.00-60.00 sec | 71.5 MBytes | | | |
| **RX** | 0.00-60.74 sec | 40.9 MBytes | 5.14 Mbit/s | 3.183 ms | 42% |

Table 4.3: Testing with UDP for jitter and data loss — Wi-Fi.

Overall, these tables illustrate the degradation in performance when moving from static to dynamic environments, especially in terms of bitrate and data loss. UDP is particularly sensitive to motion, as seen in Table 4.3.

The results determined using the MAVProxy GCS software, and Wireshark were 903 MAVLink packets with 58956 Bytes of information. The messages transmitted were HEARTBEAT, GPS_RAW_INT, MANUAL_CONTROL, and other default messages from the GCS to the UGV — REQUEST_DATA_STREAM, PARAM_REQUEST_LIST and COMMAND_LONG. The large majority of the messages transmitted were MANUAL_CONTROL, which are 65 Bytes in size. Were also transmited 6 XML report messages to the BMS.

### 4.3.2 Scenario 2: Wi-Fi-RAD-MAVLink-Bridge

The results obtained on the BMS were the same as for the Wi-Fi configuration, as shown in Figure 4.5; only the UGV is exhibited. In these tests, a scenario is added compared to Section 4.3.1 since, in this configuration, it is essential to measure the range of the network generated by the Raspberry Pi. To do this, the distance limit (47 metres) was empirically tested in an open field, where there was still data transmission, although the image already had some cuts and delays. In the case of the movement tests, the movement of the UGV was also tested in an open field, up to a maximum of 30 metres.

Table 4.4 measures TCP throughput, the performance at 1 metre (static) shows excellent results, with TX and RX achieving 414 MBytes of data transfer. This demonstrates the robustness of the system's performance at short distances. However, at 47 meters (static), the performance drops significantly, with the data transfer reduced to 18.2 MBytes and the bitrate falling to around 2.53 Mbit/s, indicating a substantial decrease in performance with increasing distance. In the movement scenario, the system transfers 268 MBytes, showing better performance than at 47 meters but still lower than at 1 meter. Movement introduces additional challenges, but the system copes better with movement than with distance.

| Role | Interval | Transfer | Bitrate |
|---|---|---|---|
| **With communication (Static – 1m)** | | | |
| **TX** | 0.00-60.01 sec | 414 MBytes | |
| **RX** | 0.00-60.22 sec | 414 MBytes | 57.6 Mbit/s |
| **With communication (Static – 47m)** | | | |
| **TX** | 0.00-60.01 sec | 18.2 MBytes | |
| **RX** | 0.00-60.40 sec | 18.2 MBytes | 2.53 Mbit/s |
| **Movement** | | | |
| **TX** | 0.00-60.01 sec | 268 MBytes | |
| **RX** | 0.00-60.83 sec | 268 MBytes | 36.6 Mbit/s |

Table 4.4: Measuring throughput with TCP — Wi-Fi-RAD-MAVLink-Bridge.

Table 4.5 tests evaluate bidirectional bandwidth. At 1 meter (static), the system performs well, transferring 397 MBytes and receiving the same amount of data, and there are no retransmissions. At 47 meters (static), the performance degrades significantly, with GCS to RPI transfer data rates dropping to 23.8 MBytes. At the same time, RPI to GCS suffers even more, transferring only 8.35 MBytes and experiencing 53 retransmissions, indicating data loss. In the movement scenario, the transfer rates are moderate, with GCS to RPI achieving 186 MBytes. However, RPI to GCS experiences some retransmissions (21), suggesting that movement impacts performance but not as severely as distance.

| Role | Direction | Interval | Transfer | Bitrate | Retr |
|---|---|---|---|---|---|
| **With communication (Static – 1m)** | | | | | |
| **TX** | GCS → RPI | 0.00-60.01 sec | 397 MBytes | | 0 |
| **RX** | GCS → RPI | 0.00-60.17 sec | 397 MBytes | 55.3 Mbit/s | 0 |
| **TX** | RPI → GCS | 0.00-60.01 sec | 22.5 MBytes | | 0 |
| **RX** | RPI → GCS | 0.00-60.17 sec | 22.2 MBytes | 3.08 Mbit/s | 0 |
| **With communication (Static – 47m)** | | | | | |
| **TX** | GCS → RPI | 0.00-60.01 sec | 23.8 MBytes | 3.32 Mbit/s | 0 |
| **RX** | GCS → RPI | 0.00-60.20 sec | 21.9 MBytes | 3.06 Mbit/s | 0 |
| **TX** | RPI → GCS | 0.00-60.01 sec | 8.35 MBytes | | 53 |
| **RX** | RPI → GCS | 0.00-60.20 sec | 8.00 MBytes | 1.11 Mbit/s | 0 |
| **With communication (Movement)** | | | | | |
| **TX** | GCS → RPI | 0.00-60.01 sec | 186 MBytes | | 0 |
| **RX** | GCS → RPI | 0.00-60.60 sec | 185 MBytes | 25.7 Mbit/s | 0 |
| **TX** | RPI → GCS | 0.00-60.01 sec | 18.8 MBytes | | 21 |
| **RX** | RPI → GCS | 0.00-60.60 sec | 18.5 MBytes | 2.58 Mbit/s | 0 |

Table 4.5: Bidirectional TCP mode for measuring throughput — Wi-Fi-RAD-MAVLink-Bridge.

Table 4.6 focuses on UDP performance, measuring jitter and data loss. At 1 meter (static), TX and RX had the same amount of data (71.7 MBytes), indicating excellent reliability in short-range UDP communication. However, at 47 meters (static), data loss becomes a significant issue, which loses 71% of the packets in the RX, and the bitrate drops to 2.42 Mbit/s. In the movement scenario, were received 65.1 MBytes at 9.10 Bbit/s. The jitter is 0.401 ms and has a data loss of 9%, showing that movement introduces some instability.

| Role | Interval | Transfer | Bitrate | Jitter | Lost Data |
|---|---|---|---|---|---|
| **With communication (Static – 1m)** | | | | | |
| **TX** | 0.00-60.00 sec | 71.5 MBytes | | | |
| **RX** | 0.00-60.05 sec | 71.5 MBytes | 9.99 Mbit/s | 0.348 ms | 0% |
| **With communication (Static – 47m)** | | | | | |
| **TX** | 0.00-60.00 sec | 61.1 MBytes | | | |
| **RX** | 0.00-61.99 sec | 17.9 MBytes | 2.42 Mbit/s | 2.851 ms | 71% |
| **With communication (Movement)** | | | | | |
| **TX** | 0.00-60.00 sec | 71.5 MBytes | | | |
| **RX** | 0.00-60.04 sec | 65.1 MBytes | 9.10 Mbit/s | 0.401 ms | 9% |

Table 4.6: Testing with UDP for jitter and data loss — Wi-Fi-RAD-MAVLink-Bridge.

In conclusion, the Wi-Fi-RAD-MAVLink-Bridge performs well in short-range and movement conditions, with limited impact from movement. However, distance significantly degrades performance, particularly in bidirectional communication and UDP mode, where data loss becomes a significant issue, especially at 47 meters.

The Raspberry Pi and GCS computer antennas have an omnidirectional radiation diagram. Looking at a similar example in a UAV configuration in [73], it is clear that having a directional antenna with a tracker on the GCS would increase the range of a RAD-MAVLink-Bridge configuration. Another change that could be made would be to use an external antenna on the Raspberry Pi, which would increase the range of the network it generates.

The results determined using the MAVProxy GCS software and Wireshark were similar to those in section 4.3.1, as they are identical configurations.

### 4.3.3 Scenario 3: Military Radios — P/PRC-525

Figure 4.6 is a screenshot of the BMS running on the GCS, where it is possible to see the LAFVIN UGV and the two other terminals running the BMS.

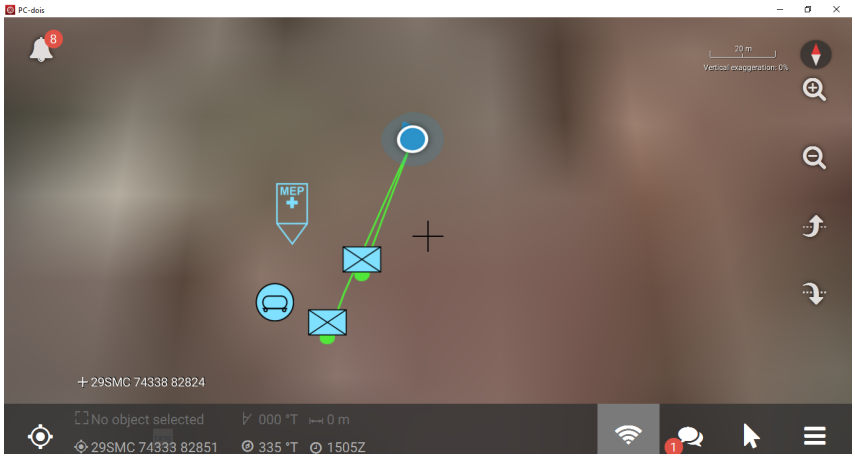


Figure 4.6: BMS — Military Radios results.

In this scenario with the P/PRC-525 military radio, the unidirectional TCP and UDP tests, respectively, in Tables 4.7 and 4.8, were carried out simultaneously. The size of the UDP and TCP packets in the test was adapted to the average size of the packets transmitted in a configuration as in Figure 4.3. The size for TCP packets was 3.3 KBytes (the size of an FFI XML message) and 64 Bytes for UDP packets (the average size of MAVLink packets sent).

When performed simultaneously, it is clear that the P/PRC-525 prioritises sending UDP data, with almost all of its bandwidth being used to send this type of data. This is because the PRC/P 525 has no data flow control, and since sending UDP data doesn't require establishing a formal connection, it prioritises sending this type of data.

As expected, the bitrates are higher in the performance situation where the channel is completely free, compared to the static scenario where MAVLink packets are sent between the Raspberry Pi and the GCS, and XML reports are sent to the BMS.

| Role | Interval | Transfer | Bitrate |
|---|---|---|---|
| **No communication (Channel Performance)** | | | |
| **TX** | 0.00-60.00 sec | 67.7 KBytes | |
| **RX** | 0.00-60.93 sec | 4.65 KBytes | 625 bit/s |
| **With communication (Static)** | | | |
| **TX** | 0.00-60.01 sec | 67.7 KBytes | |
| **RX** | 0.00-61.09 sec | 4.28 KBytes | 574 bit/s |

Table 4.7: Measuring throughput with TCP — P/PRC-525.

The high data losses, on Table 4.8, are because the iPerf3 commands were tested with a channel throughput of 16 kbits/s, and two commands were executed at the same time (unidirectional TCP and unidirectional UDP), the communication channel was overloaded, leading to losses of 62% and 65%. Another factor contributing to these losses is the Raspberry Pi bottleneck performance.

| Role | Interval | Transfer | Bitrate | Jitter | Lost Data |
|---|---|---|---|---|---|
| **No communication (Channel Performance)** | | | | | |
| **TX** | 0.00-60.01 sec | 117 KBytes | | | |
| **RX** | 0.00-61.25 sec | 64.8 KBytes | 8.46 Kbit/s | 42.838 ms | 45% |
| **With communication (Static)** | | | | | |
| **TX** | 0.00-60.01 sec | 117 KBytes | | | |
| **RX** | 0.00-60.61 sec | 37.1 KBytes | 5.01 Kbit/s | 62.137 ms | 68% |

Table 4.8: Testing with UDP for jitter and data loss — P/PRC-525.

In the bidirectional TCP tests in Table 4.9, once again, there are higher bitrates in the performance situation compared to the static scenario. In both scenarios, it is clear that the GCS is much more efficient than the Raspberry Pi at sending data, as the transfer rates from the GCS to the Raspberry Pi are much higher. It is also clear that the GCS can receive a higher percentage of the data sent by the Raspberry Pi than the other way around, once again showing that the Raspberry Pi affects the performance of the communication channel.

| Role | Direction | Interval | Transfer | Bitrate | Retr |
|---|---|---|---|---|---|
| **No communication (Channel Performance)** | | | | | |
| **TX** | GCS → RPI | 0.00-60.01 sec | 96.7 KBytes | | 0 |
| **RX** | GCS → RPI | 0.00-60.86 sec | 31.7 KBytes | 4.16 Kbit/s | 0 |
| **TX** | RPI → GCS | 0.00-60.86 sec | 64.5 KBytes | | 3 |
| **RX** | RPI → GCS | 0.00-60.01 sec | 47.1 KBytes | 6.27 Mbit/s | 0 |
| **With communication (Static)** | | | | | |
| **TX** | GCS → RPI | 0.00-60.01 sec | 80.6 KBytes | | 0 |
| **RX** | GCS → RPI | 0.00-60.86 sec | 15.2 KBytes | 1.99 Kbit/s | 0 |
| **TX** | RPI → GCS | 0.00-60.82 sec | 58.0 KBytes | | 5 |
| **RX** | RPI → GCS | 0.00-60.59 sec | 35.9 KBytes | 4.74 Kbit/s | 0 |

Table 4.9: Bidirectional TCP mode for measuring throughput — P/PRC-525.

In 60 seconds, this connection sent around 126 MAVKLink messages (120 HEARTBEAT messages and 6 GPS_RAW_INT messages) and 6 XML report messages to the BMS.

The data used by one UGV in 60 seconds can be obtained as follows: $(120*63)+(6*101)+(3.3K*6) = 27.966KBytes$

Considering a radio link in SECOM -V (16 Kbit/s), the theoretical number of UGVs per channel:

$$\frac{16K}{\frac{27.966K*8}{60}} = 4 \tag{4.1}$$

Analysing the results of the above tables and taking into account the frequency of message sending

shown in Section 4.2, maybe only one UGV can be supported by the communication channel with this configuration. This is because the Raspberry Pi has a bottleneck on the communication channel, and because there is no traffic control, the channel prioritises sending UDP. This proves to be a problem, as it is more critical for the system to send an FFI XMl than a HEARTBEAT message.

### 4.3.4 Scenario 4: Military Radios — TR5000H

When communicating using the TR5000H tactical radios, a view was obtained on the BMS similar to Figure 4.6. The size of the UDP and TCP packets and the simultaneous execution of the tests were done in the same way as in Section 4.3.3.

As in Section 4.3.3, the unidirectional TCP and UDP were carried simultaneously; they are represented, respectively, in Table 4.10 and 4.11. As expected, the transfer rate is lower in the static situation compared to the performance scenario.

In these tests, it can be seen that the TR5000H does control traffic since it does not let UDP occupy all the capacity. More throughput is used for TCP packets, which can be explained by the fact that TCP packets are larger, so more throughput is used for this transmission.

| Role | Interval | Transfer | Bitrate |
|---|---|---|---|
| **No communication (Channel Performance)** | | | |
| **TX** | 0.00-60.00 sec | 90.2 KBytes | |
| **RX** | 0.00-61.90 sec | 43.9 KBytes | 5.66 Kbit/s |
| **With communication (Static)** | | | |
| **TX** | 0.00-60.01 sec | 80.6 KBytes | |
| **RX** | 0.00-61.35 sec | 26.4 KBytes | 3.50 Kbit/s |

Table 4.10: Measuring throughput with TCP — TR5000H.

As with the P/PRC-525, there was a high level of data loss on Table 4.11 because the iPerf3 commands were tested with a channel throughput of 22 kbits/s, but as two commands were executed at the same time ( unidirectional TCP and unidirectional UDP), the communication channel was overloaded, leading to losses of 82% and 88%. Once again, the Raspberry Pi contributed to these higher losses.

| Role | Interval | Transfer | Bitrate | Jitter | Lost Data |
|---|---|---|---|---|---|
| **No communication (Channel Performance)** | | | | | |
| **TX** | 0.00-60.01 sec | 161 KBytes | | | |
| **RX** | 0.00-60.41 sec | 28.1 KBytes | 3.71 Kbit/s | 84.297 ms | 82% |
| **With communication (Static)** | | | | | |
| **TX** | 0.00-60.01 sec | 161 KBytes | | | |
| **RX** | 0.00-60.96 sec | 18.5 KBytes | 2.49 Kbit/s | 165.808 ms | 88% |

Table 4.11: Testing with UDP for jitter and data loss — TR5000H.

In the bidirectional TCP tests in Table 4.12, there are higher bitrates in the performance situation compared to the static, like in the other scenarios. In both situations, it is clear that the GCS is much more efficient than the Raspberry Pi at sending data, as the transfer rates from the GCS to the Raspberry Pi are higher. It is also clear that the GCS can receive a higher percentage of the data sent by the Raspberry Pi than the other way around, once again showing that the Raspberry Pi affects the performance of the communication channel.

| Role | Direction | Interval | Transfer | Bitrate | Retr |
|---|---|---|---|---|---|
| **No communication (Channel Performance)** | | | | | |
| **TX** | GCS → RPI | 0.00-60.01 sec | 77.3 KBytes | | 0 |
| **RX** | GCS → RPI | 0.00-60.86 sec | 26.0 KBytes | 3.41 Kbit/s | 0 |
| **TX** | RPI → GCS | 0.00-60.86 sec | 64.5 KBytes | | 5 |
| **RX** | RPI → GCS | 0.00-60.01 sec | 38.8 KBytes | 5.17 Kbit/s | 0 |
| **With communication (Static)** | | | | | |
| **TX** | GCS → RPI | 0.00-60.01 sec | 72.8 KBytes | | 0 |
| **RX** | GCS → RPI | 0.00-60.87 sec | 17.2 KBytes | 2.32 Kbit/s | 0 |
| **TX** | RPI → GCS | 0.00-60.87 sec | 38.7 KBytes | | 5 |
| **RX** | RPI → GCS | 0.00-60.01 sec | 20.7 KBytes | 2.75 Kbit/s | 0 |

Table 4.12: Bidirectional TCP mode for measuring throughput — TR5000H.

As in Subsection 4.3.3, were sent, in 60 seconds, around 126 MAVKLink messages (120 HEARTBEAT messages and 6 GPS_RAW_INT messages) and 6 XML report messages to the BMS.

Data used by one UGV in 60 seconds: $(120 * 63) + (6 * 101) + (3.3K * 6) = 27.966KBytes$

Considering a radio link in balanced mode (22 Kbits/s), the theoretical number of UGVs per channel:

$$\frac{22K}{\frac{27.966K*8}{60}} = 5 \tag{4.2}$$

Analysing the results of the above tables and taking into account the frequency of message sending shown in Section 4.2, maybe this configuration can support one more UGV on the communication channel. One more time, the Raspberry Pi constitutes a bottleneck in the communication channel.

## 4.4 Summary

In summary, four scenarios were tested with different configurations: Wi-Fi and a hybrid mode using Wi-Fi and CNR. As for Subsection 4.3.1, it is clear that this is an excellent application when there is already a network infrastructure in place, as it allows for high throughput and does not require additional equipment, as the Raspberry Pi already has a Wi-Fi antenna. As for Subsection 4.3.2, is suitable for short distances and can be improved using directional antennas and a tracker.

As for Subsections 4.3.3 and 4.3.4, which use a hybrid mode of communication using Wi-Fi and CNR, it was clear that in both scenarios, the Raspberry Pi had a bottleneck in communication, which meant that the actual capacity of the channel was lower than theoretically predicted. The scenario with the TR5000H shows a better distribution of traffic since it does traffic control, and it can be concluded that it is more suitable for this type of application.

# Chapter 5

# Conclusions

In this thesis, the first architecture for integrating UxVs into the BMS was developed. It was possible to drive a UGV without the need for specific autopilot hardware such as Pixhawk line or Ardupilot Mega. Made integrations in BMS developed for future exercises with entities outside the Army using UxVs, such as ARTEX, and data regarding the throughput required to drive a UGV, different network architectures, and limitations of the used hardware.

As for the Wi-Fi network tests, given that a communication architecture was being used with relatively low-quality image transmission and a relatively long position sending interval (10 seconds), it was noticeable that even in conditions with less available bandwidth, it was possible to establish communication with the UGV. When there was less signal available, the first thing that failed was the video transmission, with some cuts in the image.

As for the tests with Wi-Fi-RAD-MAVLink-Bridge, it was possible to observe that at short distances, it was possible to command and control the UGV. However, the signal was considerably degraded after about 40 meters. This configuration is interesting for short distances and can be extended using directional antennas. It can be applied as an alternative to another radio communication mode, focussing on short distances.

It was possible to communicate using military radios, as only MAVLink HEARTBEAT and GPS_RAW_INT messages and FFI XML were sent. Ideally, the frequency at which the messages are sent should be adjusted considering the available bandwidth, and the application itself, in the case of the P/PRC 525, would have to control the traffic. The TR5000H was advantageous in almost all the tests. It performs automatic traffic control and allows a parallel communication mode, where data and voice can be sent simultaneously on two channels, thus advantageous over the P/PRC-525.

It would be important to be more extensive in terms of the test scenarios and frequencies used, but each test carried out in *Paço de Arcos* took a long time, leaving some interesting configurations untested. The development that has been made and the connection to the Mission Planner makes it possible to drive a UGV with autopilot hardware using waypoints; in a configuration like this, military radios would only be necessary, as there would be no need to send video or control commands.

## 5.1 Future Work

As future work, it would be interesting to develop network management software that communicates with MAVLink messages, which allows UGVs to communicate with GCS in a mesh network, as shown in Figure 5.1.

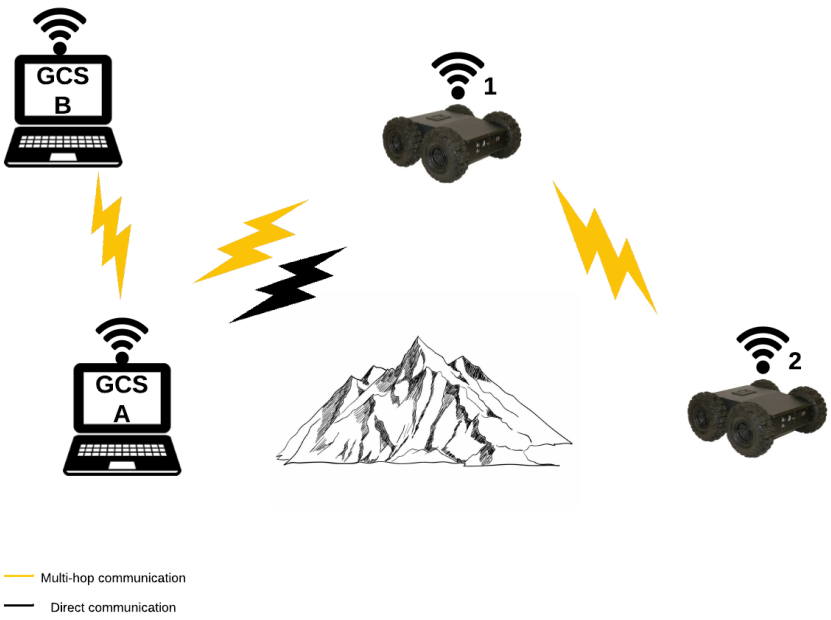


Figure 5.1: General communication architecture of a UGV with the control station in a mesh network.

Communication between the UGV (2) and the GCS (B) uses a multi-hop method, using three hops to establish communication, shown in yellow. In the case of the communication represented in black, this is directly between the UGV (1) and the GCS (A). In this thesis, all the configurations presented and tests carried out were conducted in direct communication, represented by the black connection. The network management software would ideally be developed from scratch, replacing the GCS software MAVProxy in this thesis. There would have to be a GCS managing the network so that if a GCS were taken over by the enemy, it would be possible to change the GCS of a particular UGV so that the enemy would not have control of the UGV. It could also be possible for a UGV to be managed by more than one GCS.

As a future endeavour, it would be interesting to carry out an update to the BMS, in conjunction with Critical Software, that would allow command and control of a UxV, translating from MAVLink to NATO STANAG or working directly with MAVLink and thus eliminating the need for GCS software to receive video and position and send control commands and waypoints. This allow the UGV to move without the need for the operator to receive video — they were all sent via tactical radios, as this would take better advantage of the security provided by military radios. This update would be pretty demanding, but it could start with the most straightforward functions and be inspired by Mission Planner, as it is open-source. The BMS only allows text orders, reports, and events to be sent; an update could allow specific orders to be sent directly to UxVs.

Regarding the configuration with Wi-Fi-RAD-MAVLink-Bridge Communication, it would be helpful to

test it with a directional antenna with a tracker and see the improvement in range and bandwidth. As for Military radio communication, it would be interesting to see the bandwidth in a TR5000V communication in the GCS and the UGV. In this scenario, it would be necessary to use a larger UGV, like Vigilant.

As for tests with military radios, it would be interesting to carry out tests on the move, with a distance of around 7 kilometres between the GCS and the UGV, as this is the maximum range of the TR5000H. Since it is expected that at short distances – a few hundred metres – and in line of sight, there will not be any significant differences. It would also be interesting to see the difference in bitrate in different frequency ranges on both radios.

# Bibliography


[1] H. Saputra and M. Mirdanies. Controlling Unmanned Ground Vehicle Via 4 Channel Remote Control. *Energy Procedia*, 68, 04 2015. doi: 10.1016/j.egypro.2015.03.269.

[2] J. Chen, J. Sun, and G. Wang. From unmanned systems to autonomous intelligent systems. *Engineering*, 12:16–19, 2022. ISSN 2095-8099. doi: https://doi.org/10.1016/j.eng.2021.10.007.

[3] N. S. . T. Organization. Science & Technology Trends 2023-2043 Across the Physical, Biological, and Information Domains, 2023.

[4] J. Yoo. Adaptive Multi-Path Routing Protocol in Autonomous Vehicular Networks. *Mathematics*, 11(21), 2023. ISSN 2227-7390. doi: 10.3390/math11214426. URL https://www.mdpi.com/2227-7390/11/21/4426.

[5] K. Chávez and O. Swed. Small drones for big militaries: The way ahead. *War Room - U.S. Army War College*, 2024. URL https://warroom.armywarcollege.edu/articles/small-drones/. Accessed: 2024-10-11.

[6] X. Liang, S. Zhao, G. Chen, G. Meng, and Y. Wang. Design and development of ground station for UAV/UGV heterogeneous collaborative system. *Ain Shams Engineering Journal*, 12(4):3879–3889, 2021. ISSN 2090-4479. URL https://www.sciencedirect.com/science/article/pii/S2090447921002136.

[7] B. Hament and P. Oh. Unmanned aerial and ground vehicle (UAV-UGV) system prototype for civil infrastructure missions. In *2018 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–4, 2018. doi: 10.1109/ICCE.2018.8326346.

[8] S. Beycimen, D. Ignatyev, and A. Zolotas. A comprehensive survey of unmanned ground vehicle terrain traversability for unstructured environments and sensor technology insights. *Engineering Science and Technology, an International Journal*, 47:101457, 2023. ISSN 2215-0986. URL https://www.sciencedirect.com/science/article/pii/S2215098623001350.

[9] A. Rossiter. Bots on the Ground: An Impending UGV Revolution in Military Affairs? *Small Wars & Insurgencies*, 31(4):851–873, 2020. doi: 10.1080/09592318.2020.1743484.

[10] Peter Felstead. Russian UGV Developments Influenced by Ukraine War. *European Security and Defence*, 06 2024. https://euro-sd.com/2024/06/articles/38818/russian-ugv-developments-influenced-by-ukraine-war/.

[11] D. H. Stolfi, M. Brust, G. Danoy, and P. Bouvry. Uav-ugv-umv multi-swarms for cooperative surveillance. *Frontiers in Robotics and AI*, 8, 02 2021. doi: 10.3389/frobt.2021.616950.

[12] R. Szabolcsi and J. Menyhárt. Loads Affecting UGVs Technical Status. *Review of the Air Force Academy*, 13(3):201–208, 2015. doi: 10.19062/1842-9238.2015.13.3.2.

[13] E. Almoaili and H. Kurdi. Path Planning Algorithm for Unmanned Ground Vehicles (UGVs) in Known Static Environments. *Procedia Computer Science*, 177:57–63, 2020. ISSN 1877-0509. URL https://www.sciencedirect.com/science/article/pii/S1877050920322766.

[14] J. Y. Turpo and J. Chicoma-Moreno. Tactical Robot Prototype for SWAT and Emergency Response Teams in Peru. In *Engineering, Integration, and Alliances for a Sustainable Development. Hemispheric Cooperation for Competitiveness and Prosperity on a Knowledge-Based Economy*, 2020. URL https://api.semanticscholar.org/CorpusID:226733270.

[15] Z. Zheng, Y. Shi, T. Wang, J. Liu, J. Fang, Y. Wei, and T.-S. Chua. UAVs in Multimedia: Capturing the World from a New Perspective. *National University of Singapore*, 07 2023.

[16] M. Lopatka. UGV for Close Support Dismounted Operations – Current Possibility to Fulfil Military Demand. *Challenges to national defence in contemporary geopolitical situation*, 2020:16–23, 10 2020. doi: 10.47459/cndcgs.2020.2.

[17] D. Axe. Russia's First-Ever Robotic Ground Assault Ended Badly — For the Robots. *Forbes*, 03 2024. URL https://www.forbes.com/sites/davidaxe/2024/03/29/russias-first-ever-robotic-ground-assault-ended-badly--for-the-robots/.

[18] S. Kumar, B. Sharma, J. Vanualailai, and A. Prasad. Stable switched controllers for a swarm of ugvs for hierarchal landmark navigation. *Swarm and Evolutionary Computation*, 06 2021. doi: 10.1016/j.swevo.2021.100926.

[19] B. Sivam, S. M G, and P. Sreelatha. Survey on video compression techniques for efficient transmission. *Journal of Physics: Conference Series*, 1916:012211, 05 2021. doi: 10.1088/1742-6596/1916/1/012211.

[20] S. Rowshanrad, S. Namvarasl, B. Jamasb, and M. Keshtgary. Video Codec Standards Comparison for Video Streaming Over SDN. *ARPN Journal of Systems and Software*, 5, 03 2015.

[21] H. Zeng, Z. Zhang, and L. Shi. Research and Implementation of Video Codec Based on FFmpeg. In *2016 International Conference on Network and Information Systems for Computers (ICNISC)*, pages 184–188, 2016. doi: 10.1109/ICNISC.2016.049.

[22] Y. Cheng, Q. Liu, C. Zhao, X. Zhu, and G. Zhang. Design and Implementation of Multimedia Format Converter Based on FFmpeg. In Y. Wu, editor, *Software Engineering and Knowledge Engineering: Theory and Practice*, pages 857–865, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-25349-2.

[23] G. Project. GStreamer Application Development Documentation, 2024. URL `https://gstreamer.freedesktop.org/documentation/application-development/?gi-language=c`. Accessed: 2024-08-14.

[24] K. Pahlavan and P. Krishnamurthy. Evolution and Impact of Wi-Fi Technology and Applications: A Historical Perspective. *International Journal of Wireless Information Networks*, 28:1–17, 11 2020. doi: 10.1007/s10776-020-00501-8.

[25] J. Ge, T. Li, and T. Geng. The wireless communications for unmanned surface vehicle: An overview. In Z. Chen, A. Mendes, Y. Yan, and S. Chen, editors, *Intelligent Robotics and Applications*, pages 113–119, Cham, 2018. Springer International Publishing. ISBN 978-3-319-97586-3.

[26] D. H. Scheidt. *Command and Control of Autonomous Unmanned Vehicles*, pages 1273–1298. Springer Netherlands, Dordrecht, 2015. ISBN 978-90-481-9707-1. doi: 10.1007/978-90-481-9707-1_110.

[27] E. Mozaffariahrar, F. Theoleyre, and M. Menth. A Survey of Wi-Fi 6: Technologies, Advances, and Challenges. *Future Internet*, 14(10), 2022. ISSN 1999-5903. doi: 10.3390/fi14100293.

[28] M. Olsson, S. Sultana, S. Rommer, L. Frid, and C. Mulligan. Chapter 2 - Architecture Overview. In M. Olsson, S. Sultana, S. Rommer, L. Frid, and C. Mulligan, editors, *EPC and 4G Packet Networks (Second Edition)*, pages 17–64. Academic Press, second edition edition, 2013. ISBN 978-0-12-394595-2. URL `https://www.sciencedirect.com/science/article/pii/B9780123945952000025`.

[29] I. Prlić, J. Šiško, V. Varnai, L. Pavelic, J. Macan, S. Kobešćak, M. Hajdinjak, M. Jurdana, Z. Cerovac, B. Zauner, M. Mihić, and S. Avdagić. Wi-Fi technology and human health impact: A brief review of current knowledge. *Archives of Industrial Hygiene and Toxicology*, 73:94–106, 06 2022. doi: 10.2478/aiht-2022-73-3402.

[30] C. Yeh, G. D. Jo, Y.-J. Ko, and H. K. Chung. Perspectives on 6G wireless communications. *ICT Express*, 9(1):82–91, 2023. ISSN 2405-9595. URL `https://www.sciencedirect.com/science/article/pii/S240595952100182X`.

[31] M. Girmay, V. Maglogiannis, D. Naudts, J. Fontaine, A. Shahid, E. De Poorter, and I. Moerman. Adaptive cnn-based private lte solution for fair coexistence with wi-fi in unlicensed spectrum. In *IEEE Conference on Computer Communications Workshops*, 07 2020. doi: 10.1109/INFOCOMWKSHPS50562.2020.9162663.

[32] K. F. Haque, A. Abdelgawad, and K. Yelamarthi. Comprehensive Performance Analysis of Zigbee Communication: An Experimental Approach with XBee S2C Module. *Sensors*, 22(9), 2022. ISSN 1424-8220. doi: 10.3390/s22093245.

[33] S. Jasthi, P. Ponnammal, A. Cherukuri, and A. Neti. Unmanned ground vehicle for military purpose. *International Journal of Pure and Applied Mathematics*, 119:13189–13192, 01 2018.

[34] M. Sathiyanarayanan, S. Azharuddin, S. Kumar, and G. Khan. Command Controlled Robot for Military Purpose. *International Journal for Technological Research in Engineering*, 1:2347–4718, 05 2014.

[35] A. Team. SiK Telemetry Radio, 2024. URL `https://ardupilot.org/copter/docs/common-sik-telemetry-radio.html`. Accessed: 2024-08-14.

[36] SIYI. *HM30 FULL HD Transmission System User Manual*, 01 2024.

[37] CubePilot. HereLink Overview, 2024. URL `https://docs.cubepilot.org/user-guides/herelink/herelink-overview`. Accessed: 2024-08-14.

[38] Auterion. MAVLink Communication Protocol: the open standard, 03 2022. URL `https://auterion.com/mavlink-communication-protocol-the-open-standard/`. Accessed: 2024-08-27.

[39] MAVLink. MAVLink Common Messages, 2024. URL `https://mavlink.io/en/messages/common.html`. Accessed: 2024-08-14.

[40] P. Pratik, P. Agarwal, et al. Review on mav link for unmanned air vehicle to ground control station communication. *Think India Journal*, 22(4):7093–7103, 2019.

[41] ArduPilot. Mission Planner Overview, 2024. URL `https://ardupilot.org/planner/docs/mission-planner-overview.html`. Accessed: 2024-08-14.

[42] ArduPilot. MAVProxy, 2024. URL `https://ardupilot.org/mavproxy/`. Accessed: 2024-08-14.

[43] Q. Team. QGroundControl, 2024. URL `http://qgroundcontrol.com/`. Accessed: 2024-08-14.

[44] D. Serrano. *Introduction to JAUS for Unmanned Systems Interoperability - Joint Architecture for Unmanned Systems*. NATO, Department of Intelligent Systems, ASCAMM Technology Center, Parc Tecnologic del Valles, Av. Universitat Autónoma, 23, 08290 Cerdanyola del Valles, SPAIN, 2015.

[45] *STANAG 4677: Dismounted Soldier Systems Standards and Protocols for Command, Control, Communications and Computers (C4) Interoperability (DSS C4 Interoperability)*. NATO Standardization Agency, 2014. Version: 3 October 2014.

[46] *STANAG 5527: Friendly Force Tracking Systems (FFTS) Interoperability*. NATO Standardization Agency, 2021. Edition 2.

[47] A. Ferreira. Comunicações Táticas entre Viaturas Militares. Master's thesis, Universidade de Lisboa, Lisboa, Portugal, 2018.

[48] J. Barroso. O Sistema de Informação e Comunicações Táctico (SIC-T) do Exército Português. Master's thesis, IUM (Instituto Universitário Militar), 2008. URL http://hdl.handle.net/10400.26/11629.

[49] Exército Português. Equipamentos de Comunicação do Exército Português, 2023. URL https://www.exercito.pt/pt/meios/equipamentos?menu=comunicacao. Accessed: 2023-12-23.

[50] V. Sequeira. Sistema de Comando e Controlo BMS - Battlefield Management System. *Revista Militar da Brigada Mecanizada - Atoleiros*, 34:54–57, 04 2020.

[51] North Atlantic Treaty Organization. STANAG 5527: Friendly Force Tracking. Technical report, NATO, Washington, D.C., USA, 2017.

[52] Critical Software. *Manual do Utilizador BMS – Conhecimento Situacional para Baixos Escalões*, 01 2024.

[53] Critical Software. *Installation and Configuration Manual BMS- Situational Awareness Solution for Low Echelons*, 01 2024.

[54] C. Gomes. Realidade Aumentada Aplicada ao Sistema de Combate do Soldado. Master's thesis, Instituto Superior Técnico, 2022.

[55] EID. EID Awarded Dismounted Soldier System Contract for the Portuguese Army, 2021. URL https://www.eid.pt/eid-awarded-dismounted-soldier-system-contract-for-the-portuguese-army/. Accessed: 2024-08-14.

[56] EID. *Manual de Operação e Manutenção do Rádio Táctico HF/VHF/UHF TR-525*, ed. 4 edition, Ano de publicação.

[57] R. de Transmissões. Revista Mensagem 2021. *Mensagem*, 03 2021. Boletim Informativo do Regimento de Transmissões.

[58] Rohde and Schwarz. Soveron VR Vehicular Tactical Radio For vehicular and semi-mobile platforms. Data Sheet, 2017.

[59] Rohde and Schwarz. Soveron HR Handheld Tactical Radio. Data Sheet, 2021.

[60] LAFVIN. *LAFVIN 4WD Smart Robot Car V1 User Manual*, 2024.

[61] Arduino. *Arduino MKR GSM 1400 Datasheet*, 08 2023. https://content.arduino.cc/assets/ABX00087-datasheet.pdf.

[62] Raspberry Pi Ltd. *Raspberry Pi 3 Model B+ Product Brief*, 11 2023. http://pip.raspberrypi.com.

[63] NVIDIA Corporation. *NVIDIA Jetson AGX Orin Technical Brief*, 03 2023. https://www.nvidia.com/content/dam/en-zz/Solutions/gtcf21/jetson-orin/nvidia-jetson-agx-orin-technical-brief.pdf.

[64] Raspberry Pi Ltd. *Raspberry Pi 5 Product Brief*, 04 2024. http://pip.raspberrypi.com.

[65] A. Gadekar, S. Fulsundar, P. Deshmukh, J. Aher, K. Kataria, D. V. Patel, and D. S. Barve. Rakshak: A modular unmanned ground vehicle for surveillance and logistics operations. *Cognitive Robotics*, 3:23–33, 2023. ISSN 2667-2413. URL https://www.sciencedirect.com/science/article/pii/S2667241323000083.

[66] N. N. Izwani. Experimental analysis on passive bandwidth estimation tools for iperf, netperf and spruce for mesh wireless local area network (WLAN). *Faculty of Computer System And Software Engineering, Universiti Malaysia Pahang.*, 2014. URL https://api.semanticscholar.org/CorpusID:62080231.

[67] A. V. Rodrigues, R. S. Carapau, M. M. Marques, V. Lobo, and F. Coito. Unmanned systems interoperability in military maritime operations: MAVLink to STANAG 4586 bridge. In *OCEANS 2017 - Aberdeen*, pages 1–5, 2017. doi: 10.1109/OCEANSE.2017.8084866.

[68] J. Glówka, J. Pellenz, T. Nussbaumer, J. Roning, R. Kozik, M. Baksaas, P. Bounker, G. De Cubber, A. Tiderko, A. Volk, F. Schneider, and K. Mathiassen. Demonstrating Interoperability Between Unmanned Ground Systems and Command and Control Systems. *International Journal of Intelligent Defence Support Systems*, 6:1, 01 2021. doi: 10.1504/IJIDSS.2021.10037444.

[69] E. D. I. D. Programme. Unmanned Ground Vehicles Autonomy trials, 2023. Acedido: 2023-12-23.

[70] M. Robotics. The THeMIS UGV, 2024. URL https://milremrobotics.com/defence/. Accessed: 2024-08-14.

[71] ARX Landsysteme. *Technical datasheet Gereon RCS*, 2024.

[72] NATO. *Fact Sheet Exercise REPMUS 23*, 09 2023.

[73] G. Nugroho and D. Dectaviansyah. Design, manufacture and performance analysis of an automatic antenna tracker for an unmanned aerial vehicle (UAV). *Journal of Mechatronics, Electrical Power, and Vehicular Technology*, 9:32, 07 2018. doi: 10.14203/j.mev.2018.v9.32-40.

# Appendix A

# Scripts

## A.1 Script 0 — Arduino

```
#include "IR_remote.h"
#include "keymap.h"

IRremote ir(12);

#include <Servo.h>

// Define pins
const int IR_RECEIVER_PIN = 12;
const int MOTOR_LEFT_FORWARD = 2;
const int MOTOR_LEFT_BACKWARD = 4;
const int MOTOR_RIGHT_FORWARD = 7;
const int MOTOR_RIGHT_BACKWARD = 8;
const int MOTOR_LEFT_SPEED = 5;
const int MOTOR_RIGHT_SPEED = 6;
const int SERVO_PIN = A2;

String BT_value;
volatile boolean IR_Mode_Flag;
volatile char IR_Car_Mode;
Servo servo_A2;
volatile boolean BT_Mode_Flag;

void setup() {
  Serial.begin(9600);
  BT_value = "";
  BT_Mode_Flag = false;

  pinMode(MOTOR_LEFT_FORWARD, OUTPUT);
  pinMode(MOTOR_LEFT_BACKWARD, OUTPUT);
  pinMode(MOTOR_RIGHT_FORWARD, OUTPUT);
  pinMode(MOTOR_RIGHT_BACKWARD, OUTPUT);
```

```
33
34    ir.enableIRIn();  // Inicializa o receptor IR
35    servo_A2.attach(SERVO_PIN);
36  }
37
38  void loop() {
39    // Reading commands via Bluetooth
40    if (BT_Mode_Flag) {
41      while (Serial.available() > 0) {
42        BT_value += (char)Serial.read();
43        delay(2);  // Short delay to ensure that the entire message has been received
44      }
45      if (!BT_value.equals("")) {
46        processBluetoothCommand(BT_value);
47        BT_value = "";  // Limpa o buffer
48      }
49    }
50    // Reading commands via Raspberry Pi
51    else {
52      while (Serial.available() > 0) {
53        BT_value += (char)Serial.read();
54        delay(2);  // Short delay to ensure that the entire message has been received
55      }
56      if (!BT_value.equals("")) {
57        processRaspberryPiCommand(BT_value);
58        BT_value = "";  // Limpa o buffer
59      }
60    }
61
62    // Controle via IR
63    IR_remote_control();
64  }
65
66  void processBluetoothCommand(String command) {
67    if (command.length() > 0) {
68      Serial.print("Received: ");
69      Serial.println(command);
70      if (command.length() <= 4 && command.charAt(0) == '%' && command.charAt(command.
          length() - 1) == '#') {
71        if (IR_Mode_Flag) {
72          STOP();
73          IR_Car_Mode = ' ';
74          IR_Mode_Flag = false;
75        }
76        switch (command.charAt(1)) {
77          case 'F':
78            Move_Forward(255);
79            delay(300);  // Move by 300 milissegundos
80            STOP();
```

```
81            break;
82          case 'B':
83            Move_Backward(255);
84            delay(300);  // Move by 300 milissegundos
85            STOP();
86            break;
87          case 'L':
88            Rotate_Left(230);
89            delay(300);  // Move by 300 milissegundos
90            STOP();
91            break;
92          case 'R':
93            Rotate_Right(230);
94            delay(300);  // Move by 300 milissegundos
95            STOP();
96            break;
97          case 'S':
98            STOP();
99            break;
100         default:
101           STOP();
102           break;
103       }
104     }
105   }
106 }
107
108 void processRaspberryPiCommand(String command) {
109   if (command.length() > 0) {
110     Serial.print("Received from RPI: ");
111     Serial.println(command);
112     if (command.length() <= 4 && command.charAt(0) == '%' && command.charAt(command.
          length() - 1) == '#') {
113       if (IR_Mode_Flag) {
114         STOP();
115         IR_Car_Mode = ' ';
116         IR_Mode_Flag = false;
117       }
118       switch (command.charAt(1)) {
119         case 'F':
120           Move_Forward(255);
121           delay(300);  // Move by 300 milissegundos
122           STOP();
123           break;
124         case 'B':
125           Move_Backward(255);
126           delay(300);  // Move by 300 milissegundos
127           STOP();
128           break;
```

```
129           case 'L':
130             Rotate_Left(230);
131             delay(300);  // Move by 300 milissegundos
132             STOP();
133             break;
134           case 'R':
135             Rotate_Right(230);
136             delay(300);  // Move by 300 milissegundos
137             STOP();
138             break;
139           case 'S':
140             STOP();
141             break;
142           default:
143             STOP();
144             break;
145         }
146       }
147     }
148 }
149
150 void Move_Forward(int car_speed) {
151   digitalWrite(MOTOR_LEFT_FORWARD, LOW);
152   digitalWrite(MOTOR_LEFT_BACKWARD, HIGH);
153   digitalWrite(MOTOR_RIGHT_FORWARD, LOW);
154   digitalWrite(MOTOR_RIGHT_BACKWARD, HIGH);
155   analogWrite(MOTOR_LEFT_SPEED, car_speed);
156   analogWrite(MOTOR_RIGHT_SPEED, car_speed);
157 }
158
159 void Move_Backward(int car_speed) {
160   digitalWrite(MOTOR_LEFT_FORWARD, HIGH);
161   digitalWrite(MOTOR_LEFT_BACKWARD, LOW);
162   digitalWrite(MOTOR_RIGHT_FORWARD, HIGH);
163   digitalWrite(MOTOR_RIGHT_BACKWARD, LOW);
164   analogWrite(MOTOR_LEFT_SPEED, car_speed);
165   analogWrite(MOTOR_RIGHT_SPEED, car_speed);
166 }
167
168 void Rotate_Left(int car_speed) {
169   digitalWrite(MOTOR_LEFT_FORWARD, LOW);
170   digitalWrite(MOTOR_LEFT_BACKWARD, HIGH);
171   digitalWrite(MOTOR_RIGHT_FORWARD, HIGH);
172   digitalWrite(MOTOR_RIGHT_BACKWARD, LOW);
173   analogWrite(MOTOR_LEFT_SPEED, car_speed);
174   analogWrite(MOTOR_RIGHT_SPEED, car_speed);
175 }
176
177 void Rotate_Right(int car_speed) {
```

```
178   digitalWrite(MOTOR_LEFT_FORWARD, HIGH);
179   digitalWrite(MOTOR_LEFT_BACKWARD, LOW);
180   digitalWrite(MOTOR_RIGHT_FORWARD, LOW);
181   digitalWrite(MOTOR_RIGHT_BACKWARD, HIGH);
182   analogWrite(MOTOR_LEFT_SPEED, car_speed);
183   analogWrite(MOTOR_RIGHT_SPEED, car_speed);
184 }
185
186 void STOP() {
187   digitalWrite(MOTOR_LEFT_FORWARD, HIGH);
188   digitalWrite(MOTOR_LEFT_BACKWARD, HIGH);
189   digitalWrite(MOTOR_RIGHT_FORWARD, HIGH);
190   digitalWrite(MOTOR_RIGHT_BACKWARD, HIGH);
191   analogWrite(MOTOR_LEFT_SPEED, 0);
192   analogWrite(MOTOR_RIGHT_SPEED, 0);
193 }
194
195 void IR_remote_control() {
196   switch (IR_Car_Mode) {
197     case 'b':
198       Move_Backward(255);
199       delay(300);
200       STOP();
201       IR_Car_Mode = ' ';
202       break;
203     case 'f':
204       Move_Forward(255);
205       delay(300);
206       STOP();
207       IR_Car_Mode = ' ';
208       break;
209     case 'l':
210       Rotate_Left(255);
211       delay(300);
212       STOP();
213       IR_Car_Mode = ' ';
214       break;
215     case 'r':
216       Rotate_Right(255);
217       delay(300);
218       STOP();
219       IR_Car_Mode = ' ';
220       break;
221     case 's':
222       STOP();
223       IR_Car_Mode = ' ';
224       break;
225   }
226   if (ir.getIrKey(ir.getCode(), 1) == IR_KEYCODE_UP) {
```

```
227     IR_Car_Mode = 'f';
228     IR_Mode_Flag = true;
229   } else if (ir.getIrKey(ir.getCode(), 1) == IR_KEYCODE_LEFT) {
230     IR_Car_Mode = 'l';
231     IR_Mode_Flag = true;
232   } else if (ir.getIrKey(ir.getCode(), 1) == IR_KEYCODE_DOWN) {
233     IR_Car_Mode = 'b';
234     IR_Mode_Flag = true;
235   } else if (ir.getIrKey(ir.getCode(), 1) == IR_KEYCODE_RIGHT) {
236     IR_Car_Mode = 'r';
237     IR_Mode_Flag = true;
238   } else if (ir.getIrKey(ir.getCode(), 1) == IR_KEYCODE_OK) {
239     IR_Car_Mode = 's';
240     IR_Mode_Flag = true;
241   }
242 }
```

Listing A.1: Script 0 — Arduino.

## A.2 Script 1 — Management of the link between Raspberry Pi, Arduino and GCS

```
1
2
3  # MAVProxy command: mavproxy.py --master=udpout:{IP_Wifi_RPI}:14550 --master=udpout:{
       IP_eth0_RPI}:14550 --out=udp:127.0.0.1:14551 --out=udp:127.0.0.1:14552
4
5  import time
6  import gps
7  from pymavlink import mavutil
8  from arduino_controller import ArduinoController
9
10 # Define Arduino ports and network connections
11 arduino_port = "/dev/ttyACM0"  # Adjust as needed
12 ETHERNET_IP = '192.168.1.100'  # Replace with your Ethernet Raspi IP
13 WIFI_IP = '172.18.66.172'      # Replace with your Wi-Fi Raspi IP
14 UDP_PORT = 14550               # Common UDP port for MAVLink
15
16 # Create MAVLink connections for Ethernet and Wi-Fi
17 ethernet_master = mavutil.mavlink_connection('udpin:{}:{}'.format(ETHERNET_IP, UDP_PORT),
       source_system=255, source_component=158)
18 wifi_master = mavutil.mavlink_connection('udpin:{}:{}'.format(WIFI_IP, UDP_PORT),
       source_system=255, source_component=158)
19
20 wifi_master.wait_heartbeat()
21 ethernet_master.wait_heartbeat()
22
```

```
print("Heartbeat received from both Ethernet and Wi-Fi channels")

# Instantiate the Arduino controller
arduino_controller = ArduinoController(port=arduino_port)
arduino_controller.connect()
print("Connection to Arduino established")

# Function to send heartbeat via Ethernet
def send_heartbeat():
    """Sends a heartbeat message with predefined parameters over Ethernet."""
    try:
        # Create a MAVLink heartbeat message
        msg = ethernet_master.mav.heartbeat_encode(
            mavutil.mavlink.MAV_TYPE_GENERIC,  # Type of MAV (Ground Control Station)
            mavutil.mavlink.MAV_AUTOPILOT_GENERIC,
            0,
            0,
            mavutil.mavlink.MAV_STATE_STANDBY
        )
        # Send the heartbeat message via Ethernet
        wifi_master.mav.send(msg)
        ethernet_master.mav.send(msg)
        print("Heartbeat sent via Ethernet and WiFi")
    except Exception as e:
        print(f"Error sending heartbeat: {e}")

# Function to send control commands to Arduino via Wi-Fi
def send_command_to_arduino(command):
    try:
        arduino_controller.send_command(command)
        print(f"Command sent to Arduino: {command}")
    except Exception as e:
        print(f"Error sending command to Arduino: {e}")
    time.sleep(0.1)  # Wait a bit for Arduino to process the command

# Function to send position via Ethernet
def send_position(latitude, longitude, altitude):
    """Sends a position message with given parameters over Ethernet."""
    try:
        # Ensure latitude, longitude, and altitude are numbers
        latitude = float(latitude)
        longitude = float(longitude)
        altitude = float(altitude) if altitude is not None else 0.0  # Use 0.0 as default
            if altitude is None

        # Create a MAVLink message for position
        msg = ethernet_master.mav.gps_raw_int_encode(
            0,
            mavutil.mavlink.GPS_FIX_TYPE_3D_FIX,
```

```
            int(latitude * 1e7),  # Latitude in degrees * 1e7
            int(longitude * 1e7),  # Longitude in degrees * 1e7
            int(altitude * 1e3),  # Altitude in meters * 1000
            0,
            0,
            0,
            0,
            0,
            0,
            0,
            0,
            0,
            1,    # Vehicle heading
            0
        )
        # Send the position message via Ethernet
        ethernet_master.mav.send(msg)
        print(f"Position sent via Ethernet: Lat={latitude}, Lon={longitude}, Alt={
            altitude}")
    except ValueError:
        print(f"Invalid values for latitude ({latitude}), longitude ({longitude}), or
            altitude ({altitude}).")
    except Exception as e:
        print(f"Error sending position: {e}")

# Function to get GPS data
def get_gps_data():
    """Retrieves GPS data."""
    # Initialize the GPS session
    session = gps.gps(mode=gps.WATCH_ENABLE)
    try:
        while True:
            # Receive the next packet
            report = session.next()
            # Check if the packet contains position data
            if report['class'] == 'TPV':
                latitude = getattr(report, 'lat', None)
                longitude = getattr(report, 'lon', None)
                altitude = getattr(report, 'alt', None)
                if latitude is not None and longitude is not None:
                    print(f"GPS Data: Latitude={latitude}, Longitude={longitude},
                        Altitude={altitude}")
                    return latitude, longitude, altitude
            time.sleep(1)
    except KeyboardInterrupt:
        print("GPS data retrieval stopped.")
    except Exception as e:
        print(f"Error retrieving GPS data: {e}")
    return None, None, None
```

```
117
118 def handle_manual_control(msg):
119     """Handles MANUAL_CONTROL messages received via Wi-Fi."""
120     x = msg.x
121     y = msg.y
122     z = msg.z
123     r = msg.r
124
125     print(f"Received MANUAL_CONTROL: x={x}, y={y}, z={z}, r={r}")
126
127     if x > 500:
128         send_command_to_arduino('%F#')  # Move forward
129     elif x < -500:
130         send_command_to_arduino('%B#')  # Move backward
131
132     if z > 500:
133         send_command_to_arduino('%R#')  # Move right
134     elif z < -500:
135         send_command_to_arduino('%L#')  # Move left
136
137 # Main function to execute both tasks in parallel
138 def main():
139     position_interval = 10  # 10-second interval for sending position (adjust as needed)
140     last_position_time = time.time()
141
142     try:
143         while True:
144             # Send heartbeat via Ethernet every second
145             send_heartbeat()
146
147             # Check if it's time to send position data
148             current_time = time.time()
149             if current_time - last_position_time >= position_interval:
150                 # Get GPS data
151                 latitude, longitude, altitude = get_gps_data()
152                 if latitude is not None and longitude is not None:
153                     # Send position via Ethernet
154                     send_position(latitude, longitude, altitude)
155                 last_position_time = current_time  # Update the last sent position time
156
157             # Listen for messages received via Wi-Fi
158             msg = wifi_master.recv_match(blocking=False)
159             while msg:
160                 if msg.get_type() == 'MANUAL_CONTROL':
161                     handle_manual_control(msg)
162                 msg = wifi_master.recv_match(blocking=False)
163
164             time.sleep(1)  # Small pause to avoid overloading the CPU
165
```

```
    except KeyboardInterrupt:
        print("Program terminated by user")
        arduino_controller.disconnect()

if __name__ == "__main__":
    main()
```

Listing A.2: SScript 1 — Management of the link between Raspberry Pi, Arduino and GCS.

## A.3 Script 2 — Management of the link between GCS and BMS

```
import time
import xml.etree.ElementTree as ET
from datetime import datetime
from pymavlink import mavutil
import socket
import xml.dom.minidom
import os

# Server address configuration
SERVER_IP = '192.168.211.42'  # Local server IP address
SERVER_PORT = 48882           # Server port

# Create connection with MAVProxy
master = mavutil.mavlink_connection('udpin:0.0.0.0:14552', source_system=255,
    source_component=158)
master.wait_heartbeat()
print("Heartbeat received from MAVProxy")

# Function to create XML report
def create_bms_xml(latitude, longitude):
    current_time = datetime.utcnow()

    xml_data = f'''<?xml version="1.0" encoding="utf-8"?>
<mtf:FriendlyForceInformation xmlns:mtf="urn:nato:mtf:app-11(d):1:ffi" xmlns:xsi="http://
    www.w3.org/2001/XMLSchema-instance">
    <OperationCodeword>
        <OperationCodeword>TEST</OperationCodeword>
    </OperationCodeword>
    <MessageIdentifier>
        <MessageTextFormatIdentifier>FFI</MessageTextFormatIdentifier>
        <Standard>APP-11(D)</Standard>
        <Version>1</Version>
        <Originator>PRT</Originator>
        <ReferenceTimeOfPublication>
            <DateTimeIso>
                <Year4Digit>{current_time.strftime("%Y")}</Year4Digit>
                <MonthNumeric>{current_time.strftime("%m")}</MonthNumeric>
```

```
                <Day>{current_time.strftime("%d")}</Day>
                <TimeDelimiter>T</TimeDelimiter>
                <HourTime>{current_time.strftime("%H")}</HourTime>
                <MinuteTime>{current_time.strftime("%M")}</MinuteTime>
                <SecondTime>{current_time.strftime("%S")}</SecondTime>
                <TimeZoneZulu>Z</TimeZoneZulu>
            </DateTimeIso>
        </ReferenceTimeOfPublication>
        <MessageSecurityPolicy>NATO</MessageSecurityPolicy>
        <MessageClassification>
            <SecurityClassificationExtended>UNCLASSIFIED</SecurityClassificationExtended>
        </MessageClassification>
    </MessageIdentifier>
    <GeodeticDatum>
        <GeodeticDatum>WGE</GeodeticDatum>
    </GeodeticDatum>
    <TrackInformation>
        <TrackSource>
            <TransponderId>PRTUGVLAVIN01</TransponderId>
            <System>UGV</System>
            <Subsystem>LAFVIN</Subsystem>
            <Nationality>
                <GeographicalEntity>PRT</GeographicalEntity>
            </Nationality>
        </TrackSource>
        <TrackSecurityLabel>
            <TrackSecurityPolicy>NATO</TrackSecurityPolicy>
            <TrackSecurityClassification>
                <SecurityClassificationExtended>UNCLASSIFIED</
                    SecurityClassificationExtended>
            </TrackSecurityClassification>
            <TrackSecurityCategory>RELEASABLE TO PRT ARMY</TrackSecurityCategory>
        </TrackSecurityLabel>
        <TrackPositionalData>
            <Time>
                <Year4Digit>{current_time.strftime("%Y")}</Year4Digit>
                <MonthNumeric>{current_time.strftime("%m")}</MonthNumeric>
                <Day>{current_time.strftime("%d")}</Day>
                <TimeDelimiter>T</TimeDelimiter>
                <HourTime>{current_time.strftime("%H")}</HourTime>
                <MinuteTime>{current_time.strftime("%M")}</MinuteTime>
                <SecondTime>{current_time.strftime("%S")}</SecondTime>
                <TimeZoneZulu>Z</TimeZoneZulu>
            </Time>
            <Location>
                <LatitudeDegrees>{str(int(latitude / 1e7))}</LatitudeDegrees>
                <LatitudeMinutes04DecimalPlaces>{"{:.4f}".format(abs((latitude / 1e7 -
                    int(latitude / 1e7)) * 60))}</LatitudeMinutes04DecimalPlaces>
```

```
                <LatitudinalHemisphere>{"N" if latitude >= 0 else "S"}</
                    LatitudinalHemisphere>
                <Hyphen>-</Hyphen>
                <LongitudeDegrees>{str(abs(int(longitude / 1e7)))}</LongitudeDegrees>
                <LongitudeMinutes04DecimalPlaces>{"{:.4f}".format(abs((longitude / 1e7 -
                    int(longitude / 1e7)) * 60))}</LongitudeMinutes04DecimalPlaces>
                <LongitudinalHemisphere>{"E" if longitude >= 0 else "W"}</
                    LongitudinalHemisphere>
            </Location>
        </TrackPositionalData>
        <TrackIdentificationData>
            <UnitSymbol>SFGPEV----------</UnitSymbol>
            <SymbolStandard>APP-6(B)</SymbolStandard>
            <UnitShortName>UGV-LAFVIN02</UnitShortName>
        </TrackIdentificationData>
    </TrackInformation>
</mtf:FriendlyForceInformation>'''

    filename = "bms_report.xml"
    with open(filename, 'w') as f:
        f.write(xml_data)

    return filename

# Function to send XML file to the server
def send_xml_file(filename, server_ip, server_port):
    with open(filename, 'rb') as f:
        data = f.read()

    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.connect((server_ip, server_port))
            s.sendall(data)
            print(f"File {filename} sent to {server_ip}:{server_port}")
    except Exception as e:
        print(f"Error sending file: {e}")

# Function to get GPS_RAW_INT message data
def get_position_data(msg):
    latitude = msg.lat
    longitude = msg.lon
    altitude = msg.alt
    return latitude, longitude, altitude

def main():
    try:
        while True:
            # Receive MAVLink messages
            msg = master.recv_match(type='GPS_RAW_INT', blocking=True)
```

```
128             if msg:
129                 latitude, longitude, altitude = get_position_data(msg)
130                 print(f"Latitude: {latitude / 1e7}, Longitude: {longitude / 1e7},
                        Altitude: {altitude / 1e3}")
131
132                 # Create XML report with received data
133                 filename = create_bms_xml(latitude, longitude)
134
135                 # Send XML report to the server
136                 send_xml_file(filename, SERVER_IP, SERVER_PORT)
137
138             time.sleep(15)
139     except KeyboardInterrupt:
140         print("Program terminated by user")
141
142 if __name__ == "__main__":
143     main()
```

Listing A.3: Script 2 — Management of the link between GCS and BMS.

# Appendix B

# iPerf3 Packet Loss

| Role | Interval | Lost/Total Datagrams |
|---|---|---|
| **With communication (Static)** | | |
| **TX** | 0.00-60.01 sec | |
| **RX** | 0.00-60.06 sec | 336/51363 (0.65%) |
| **With communication (Movement)** | | |
| **TX** | 0.00-60.00 sec | |
| **RX** | 0.00-60.74 sec | 20406/49768 (41%) |

Table B.1: Packet loss — Wi-Fi.

| Role | Interval | Lost/Total Datagrams |
|---|---|---|
| **With communication (Static – 1m)** | | |
| **TX** | 0.00-60.00 sec | 0/51360 (0%) |
| **With communication (Static – 47m)** | | |
| **TX** | 0.00-60.00 sec | |
| **RX** | 0.00-61.99 sec | 31070/43901 (71%) |
| **With communication (Movement)** | | |
| **TX** | 0.00-60.00 sec | |
| **RX** | 0.00-60.04 sec | 11240/51364 (22%) |

Table B.2: Packet loss — Wi-Fi-RAD-MAVLink-Bridge.

| Role | Interval | Lost/Total Datagrams |
|---|---|---|
| **No communication (Channel Performance)** | | |
| **TX** | 0.00-60.01 sec | |
| **RX** | 0.00-61.25 sec | 1109/1777 (62%) |
| **With communication (Static)** | | |
| **TX** | 0.00-60.01 sec | |
| **RX** | 0.00-60.61 sec | 1086/1679 (65%) |

Table B.3: Packet loss — P/PRC-525.

| Role | Interval | Lost/Total Datagrams |
|---|---|---|
| **No communication (Channel Performance)** | | |
| **TX** | 0.00-60.01 sec | |
| **RX** | 0.00-60.41 sec | 1925/2314 (82%) |
| **With communication (Static)** | | |
| **TX** | 0.00-60.01 sec | |
| **RX** | 0.00-60.96 sec | 2126/2422 (88%) |

Table B.4: Packet loss — TR5000H.