# Plataforma de Drones Aéreos Multi-tecnologia para Suporte a Comunicações Críticas

**Project in Computer and Informatics Engineering**

University of Aveiro

Beatriz Rodrigues, Bernardo Falé, Diogo Correia, Francisco Martinho, João Simões, Lara Rodrigues

**Group 3**

2021/2022

# Plataforma de Drones Aéreos Multi-tecnologia para Suporte a Comunicações Críticas

Department of Electronics, Telecommunications and Informatics
University of Aveiro

Beatriz Rodrigues (93023) bea.rodrigues@ua.pt,
Bernardo Falé (93331) mbfale@ua.pt,
Diogo Correia (90327) diogo.correia99@ua.pt,
Francisco Martinho (85088) martinho.francisco@ua.pt,
João Simões (88930) jtsimoes@ua.pt,
Lara Rodrigues (93427) laravieirarodrigues@ua.pt

January 2022

# Abstract

With the normalisation of the use of drones in a professional context we are reaching highland of what is possible to achieve with a human operator, being the next step the introduction of a tool that allows the operator to plan, visualize, analyse and change a mission in real time. This functionalities are displayed to the operator via a dashboard, which uses an Influx database to allow the dashboard to have real-time access to the information of a mission and a way to keep that same data persistent. The dashboard also has tools to create missions in a style similar to "Scratch" (drag-and-drop), so a user that doesn't have programming knowledge can create and deploy missions that will be translated to a Groovy file and loaded to an Unmanned Aerial Vehicle (UAV)'s. The UAV's communicate via WAVE (802.11p) between them as well as to the ground-station, this is justified with the virtually nonexistent setup time compared with WiFi that is commonly used and the need to establish the communication as fast as possible since the time window can be narrow, the UAV's are also geared with a variety of sensors, a LoRaWAN® board to collect data from another board that is geographically far as well as to serve as a gateway and a tracking device in case the UAV fails to return. All this data is then sent to the ground-station and resent to the Influx database, completing this way the cycle.

# Resumo

Com a normalização da utilização de drones na área profissional estamos a atingir o planalto do que é possível fazer com um operador humano, sendo o próximo passo introduzir uma ferramenta que permita ao operador planear, visualizar, analisar e alterar missões em tempo real. Estas funcionalidades são apresentadas ao operador via dashboard que se apoia numa base de dados Influx, de maneira a permitir um acesso em tempo real aos dados de uma missão bem como providenciar persistência dos mesmos. A dashboard também contém ferramentas estilo "Scratch" (drag-and-drop) para permitir que qualquer operador crie e dê deploy a uma missão, com o mínimo de conhecimentos de programação. Missão esta que posteriormente será traduzida para um ficheiro Groovy e enviada para as Unmanned Aerial Vehicle (UAV)'s. As UAV's comunicam entre si via WAVE (802.11p) e com a ground-station, isto é justificado pelo tempo de setup ser virtualmente nulo em comparação com o WiFi convencional e a necessidade desta comunicação ser feita o mais rapidamente possível, visto que a janela temporal pode ser curta. As UAV's também se encontram equipadas com sensores variados e uma placa de LoRaWAN® para a recolha de informação a longas distâncias, bem como servir de gateway e fazer tracking do UAV em situação do retorno não ser possível. Todos estes dados recolhidos são enviados e tratados pela ground-station e enviados para a base de dados Influx, completando assim o ciclo.

# Contents

# List of Figures

# Abbreviations

| | |
|---|---|
| **AES** | Advanced Encryption Standard |
| **API** | Application Programming Interface |
| **CSS** | Cascading Style Sheets |
| **DB** | Database |
| **FANET** | Flying Ad-Hoc Network |
| **HTML** | Hypertext Markup Language |
| **ICT** | Information and Communication Technologies |
| **IoT** | Internet of Things |
| **ITS** | Intelligent Transportation Systems |
| **JSON** | JavaScript Object Notation |
| **LoRaWAN** | Long Range Wide Area Network |
| **LPWA** | Low Power Wide Area |
| **M2M** | Machine-to-Machine |
| **OBU** | On-Board Unit |
| **RDBMS** | Relational Database Management System |
| **RPA** | Remotely-Piloted Aircraft |
| **RSU** | Road-Side Unit |
| **SPA** | Single-Page Applications |
| **SQL** | Structured Query Language |
| **TSDB** | Time-Series Database |
| **UAS** | Unmanned Aircraft System |
| **UAV** | Unmanned Aerial Vehicle |

# Chapter 1

# Introduction

## 1.1 Motivation

An Unmanned Aerial Vehicle (UAV), commonly known as a drone, is an aircraft without any human pilot, crew or passengers on board. UAV's are a component of an Unmanned Aircraft System (UAS), which include additionally a ground-based controller and a system of communications with the UAV. The flight of UAV's may operate under remote control by a human operator, as Remotely-Piloted Aircraft (RPA), or with various degrees of autonomy, such as autopilot assistance, up to fully autonomous aircraft that have no provision for human intervention [1].

Every day new drones, with new and multiple functionalities, are launched in the world market. This incredible technology, extremely versatile, can be used for a thousand and one applications, and the possibilities for professional performance are very promising, both for well-established companies and entrepreneurs. They are an important part of the economy for their advanced mechanisms and impressive functionalities. The growing interest of users in drone technology has introduced new uses and purposes for these devices, which make them an excellent alternative for some specific objectives/missions.

Within several research projects that integrate drones, communication, and imaging, an Information and Communication Technologies (ICT) infrastructure has been developed that enables an innovative set of services and applications supported by aerial drones. Some of the real scenarios considered for the infrastructure include:

- monitoring and support in emergency situations and natural disasters, such as forest fires and earthquakes;
- patrolling urban areas and supporting law enforcement;
- tourism applications such as live-streaming of video from drones traversing places of interest.

The developed infrastructure supports the control, through a base of operations, of fleets of aerial drones, allowing the execution of missions autonomously, which already have some complexity. Multi-drone missions allow the control of multiple drones performing varied tasks as part of a central mission. Sensory data collection (telemetry), and video live-streaming by cameras integrated into the drones is also possible.

Nonetheless, this platform can now be further developed and improved by adding new features.

## 1.2 Goals

The goals of this work will focus mainly on two areas of work. First, it is intended to enrich the set of predefined missions that the current platform already supports. For this, it will be necessary to develop all the drone control logic needed to complete the mission, using whenever necessary the set of sensors that equip the various drones. In parallel, and in a complementary logic, the intention is to make the communication network formed by the drones, through new communication technologies, more efficient and more resilient. The various communication technologies available will be analyzed and routes for routing traffic (mission control and services developed with the drone platform) will be created, according to the existing communication conditions.

In summary, the specific objectives of this project are:

a. Addition and configuration of new communication technologies to the drone platform;

b. Development of traffic management mechanisms on the drone network;

c. Extension of the dashboard for monitoring the network and the performance of the traffic management mechanisms;

d. Designing of two to three missions to be executed on the platform and implementation of their respective logic.

## 1.3 Document structure

In addition to the introduction, this document has two more chapters. On Chapter 2 it is described the state of the art. The tools and technologies that are being used for the development of this project are presented. The information needed to support the use of these technologies is also provided. Chapter 3 presents the architecture proposed to achieve all the goals of this project and is also where the system requirements (functional and non-functional) are described, as well as the actors and use cases.

# Chapter 2

# State of the Art

## 2.1 Unmanned Aerial Vehicles

An UAV is an aircraft that is guided autonomously, by remote control, or both; it carries several sensors and technologies that allow the drone (in this case) to communicate with other entities. These autonomous flying vehicles are equipped with software and hardware that allows the collecting of the data; These blocks of data are essential to the stability of the UAV, especially when the autonomous mode is enabled. The UAV's are able to make processes faster and more flexible, while also improving the precision.They have also become more affordable. Also, most systems in which UAV's are integrated include a ground-station control module and a communication channel between both. UAV's can intervene in numerous scenarios ranging from agriculture, tourism, surveillance and sensing missions, logistics transportation, weather monitoring, fire detection, communication relaying, and emergency search and rescue. However, such new applications have a critical requirement in common, which is the need for a communications system that allows UAV's to directly connect to each other for data exchange.

### 2.1.1 Autonomous Control

The available UAVs are capable of executing tasks without human input, namely automated tasks. Autonomy can be considered at different levels. Remotely sending an action to be performed by the drone can be considered an autonomous task, since after receiving the command, the drone moves by itself without human interference. A higher level of autonomy would include having these commands being sent automatically, without having the user to provide each one individually. In order to achieve this, research has been done to equip the drone with an On-Board Unit (OBU) that we will address later.

## 2.2 Flying Ad-Hoc Networks

Flying Ad-Hoc Networks (FANET's) are a subclass of Mobile Ad-Hoc Networks (MANET's) that provides wireless communications between moving UAV's, and is the concept of having a network composed by different UAV's connected in an ad-hoc. Some of the most distinct characteristics of FANET's is that the communicating nodes are highly mobile and that most of these nodes use wireless communication protocols, such as IEEE802.11(p,a,n). However, a considerable amount of problems arise with these "characteristics", such as low band-width, (relative) high connectivity time, and low transmission rates. In a FANET, drones can communicate with each other through OBUs, in an environment commonly described as V2V communication, and can communicate with Roadside Units (RSU's) in a V2I paradigm.

### 2.2.1 Entities

The FANET is composed by various elements:

- **On-Board Unit (OBU)**: It is equipped on the UAV, and is responsible for the communication between drones and road-side units. The available OBU's are the PC Engines APU 2, and they are equipped with sufficient processing power to conclude any given task. These OBU's are able to connect to other interfaces through the wireless protocols previously referenced.
- **Road-Side Unit (RSU)**: The static node in a FANET. It is usually installed near high traffic roads and is normally connected by cable or fiber to a fixed network. The three main objectives of the RSU's are to extend the communication range of the network and to provide internet connectivity to the OBU.

### 2.2.2 Domains and Characteristics

The FANET is divided in 3 communication domains :

- In-Drone domain: Connected OBU's
- Ad-hoc domain: Where V2V and V2I communications are formed
- Infrastructure domain: Connections between the RSU's and the internet

After carefully analysing FANET's, we can say that the UAV's movement is somewhat limited, although mobile, namely because of the connection constraints and several external factors, such as the weather conditions. On the other hand, the drones are equipped sensors that make possible the prediction of movements through the retrieval of critical data i.e speed, direction, and position. If there is a high number of vehicles equipped with OBU's, the network can have a large number of connected elements, and therefore, can be considered a large scale network, which is good. After the collecting of information by the UAV's, the data is ready to be transmitted, thus the ease of developing useful applications and services.

## 2.3 Mission

A mission consists in interacting with the drone by sending high-level commands.

## 2.4 Mission Planning

The mission planning framework aims at improving the process of building a specific mission. Using Groovy, it allows the user to write a mission script with commands close to natural language, a command may be created and integrated in Groovy like a plugin. In a mission script, the user can assign any number of available drones and instruct them to perform commands according to the restrictions they see fit. Multiple drones may be controlled in the same mission, either working independently or collaborating towards a goal. During said missions a drone will also send back telemetry of itself and its sensors, a senor can also be easily created and import as a plugin in the framework [2].

## 2.5 Communication

### 2.5.1 Wave

IEEE 802.11p(WAVE) is an approved amendment to the IEEE 802.11 standard to address the specific problems in vehicular communications, which reduces the connection setup from values of 1-2 seconds to around 10-20 milliseconds (which is very critical in vehicular environments with very

low connectivity times).
It defines enhancements to 802.11 (the basis of products marketed as WiFi) required to support Intelligent Transportation Systems (ITS) applications. This includes data exchange between high-speed vehicles and between the vehicles and the roadside infrastructure, so called V2X communication (V2V and V2I communications are performed using wave). This protocol can establish rapid connections (10- 20ms) and can provide up to 1000 meters of communication range, but can not reach high bandwidth values (higher than 27 Mbps).
The WAVE communication is performed with a mini-PCIe wireless card allowing communication on the 5.9 GHz band [3].

### 2.5.2 LoRaWAN

#### Overview

The Long Range Wide Area Network (LoRaWAN)® is a Low Power Wide Area (LPWA) end-to-end system architecture designed to wirelessly connect battery operated 'things' to the internet in regional, national or global networks with ease; It is the most popular technology used among low power networks, due not only to its low-cost nature, but also for its simplistic implementation. Concerning safety and authentication, LoRa makes use of Advanced Encryption Standard (AES) encryption and two layers of security; One layer for network security, that is used to secure devices, and the application layer, which is used to protect application data.

#### Features

LoRaWAN® has a variety of features that allows it to be a great solution of a wide variety of scenarios like Internet of Things (IoT), Machine-to-Machine (M2M), smart city & industrial applications, in those features we can include the following:

- Low-cost, mobile and secure bi-directional communication
- Low power consumption
- Easily scalable
- Supports redundant operation
- Supports geolocation
- Allows devices to be updated remotely
- Open-source

#### Network Architecture

LoRaWAN® is the key of the network architecture, it configures the end device's and connects them to an IP connected network. Figure 2.1 encapsulates how LoRaWAN® is generally used, starting from a device connected via a gateway to a much larger network where the data is then used.

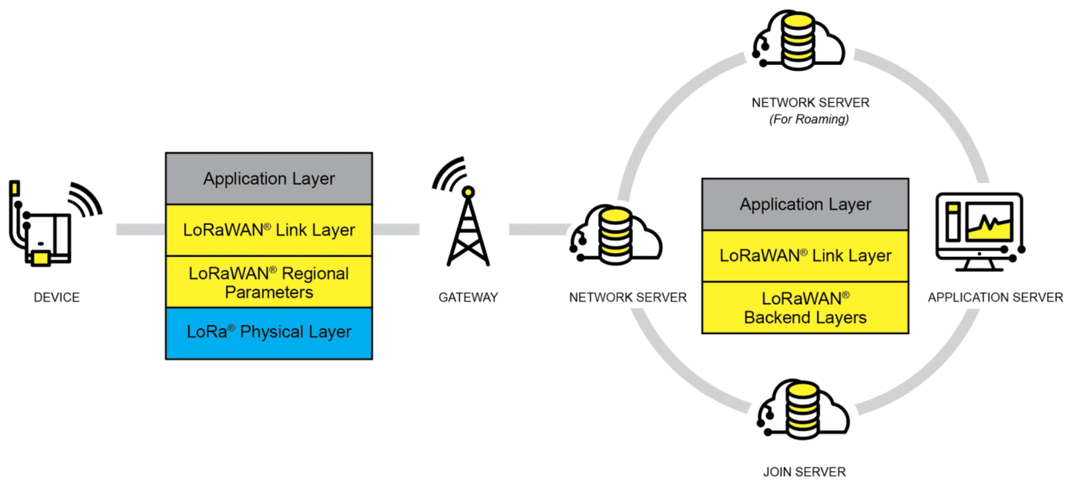

Figure 2.1: Network Architecture [4]

Components of an ordinary network:

**Devices:** Regular devices such like sensors, detectors, and actuators. They send the gathered data to the gateway. Unlike other networking tech, there are no unicast links between devices and gateways; Any LoRaWAN® module can receive transmission, as long as they are within the same parameters.

**Gateways:** This is where the messages are forwarded from the Devices to the Servers, and vice versa.

**Servers:** It monitors the previous components, and is responsible for removing the duplicates caused by the forwarding of the same information; It also routes the messages to the corresponding application layers.

## 2.6 Data Persistence

### 2.6.1 Database

The telemetry that the Drones feed the ground-station is not kept forever. It is in fact just saved momentarily with MongoDB, and then lost whenever new information comes along. There is no data persistence.

This was good enough for the expectations set, so far, for the management of the information created on a mission. It had to be possible to watch the mission's development in real time on the Dashboard, but it didn't need to be stored for future reference, hence the "JSON-like" type of database.

Upon looking at the new objectives set for us, it is clear that the type of database has to change. The project is going towards saving the information from every mission, in order to be referred to in any time in the future.

#### Time-series Database

A Relational Database Management System (RDBMS), like MySQL, was thought first as an alternative to MongoDB (because of its popularity, and our knowledge on databases), but then, after some research, we came to the conclusion that a Time-Series Database (TSDB) would be the way to go.

One reason is its ability to handle time-related data, where the data is stored in "collections" that are aggregated over time. In this type of system, every point that is stored comes with a timestamp. Having in consideration our need to browse missions that occur in different points in time, a RDBMS would work, but a TSDB makes the work easier and more efficient.

TSDB are optimized for a fast ingestion rate. RDBMS on the other hand, because it has to re-index the data for it to be accessed faster, the performance tend to decrease with the size of the collection, as seen in Figure 2.2 [5].

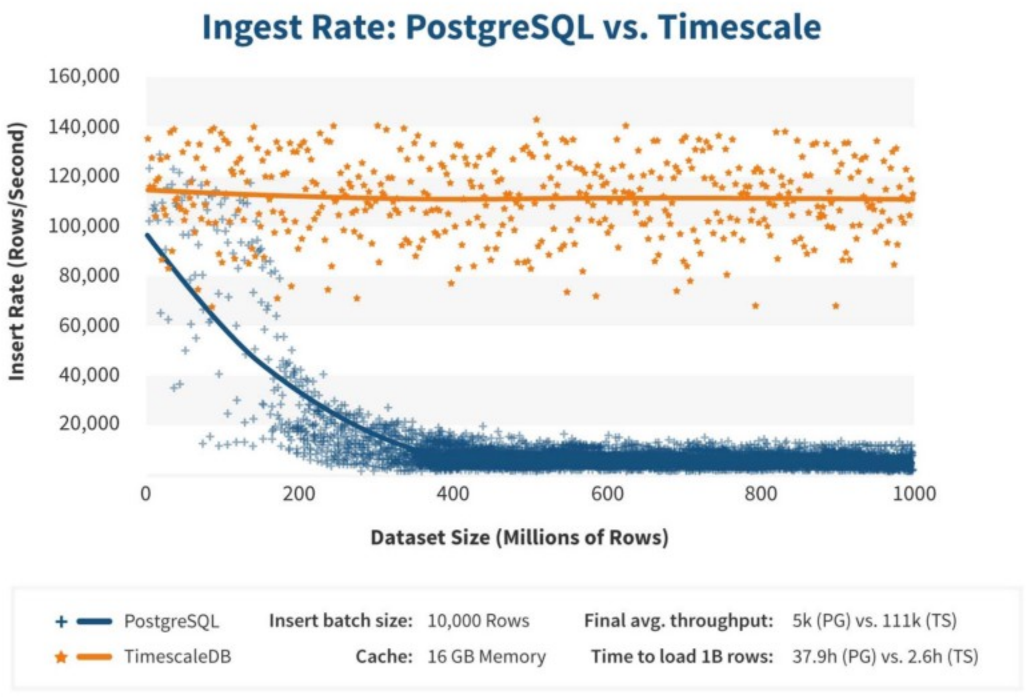

Figure 2.2: Different behaviour observed between RDBMS and TSDB [5]

#### InfluxDB

We chose InfluxDB because of its popularity among time-series databases, and also because of its easy integration with Grafana. This is important because Grafana is a tool that creates time-series data visualization to Dashboards, which is exactly what we need.

InfluxDB works with retention policy, that defines how long it keeps the data stored. This might be useful when dealing, for example, with test missions that are not meant to be archived. You could either want to check some values once and forget about it (like what is happening now with MongoDB), or save it only until next week, when you know they won't be needed anymore and would just become accumulated waste in the future, if not discarded.

### 2.6.2 API

The Application Programming Interface (API), "the connector of it all", was created with Django. It allows the ground-station to save the telemetry from the Drones into MongoDB, and then makes it easily accessible, through routes, to the Dashboard.

We opted to use this already existing module, with the difference being the database it will be connecting to, as mentioned before.

## 2.7 Dashboard

The dashboard is the module that has the objective to create an abstraction layer between the code behind the functionalities and the final user. It is the interface the user will use to interact with the mission. It will also give him a lot of information: values of the sensor, location, commands to control the drone, battery information, live video, and much more.

It allows the upload and tracking of missions, connect to multiple drones and ground-stations, displaying a map for easy interpretation. This dashboard also contains a section dedicated to the display and control of the drone camera quality settings, and multiple charts that illustrate the continuous status of the attached drone sensors.

The front-end is built with Nuxt.js, which is a JavaScript framework mainly based on Vue.js, made to create interactive Single-Page Applications (SPA), using also HTML and CSS for the structure and visuals of the website [6].

The back-end uses MongoDB as a database and Django to connect to the ground-station for control and management, like described on 2.6.2.

This module is very well developed, so the plan here is to continue using it and just introduce the following new features:

- Get and display as much relevant data to the user as possible from API;
- Implement the possibility to further modify the missions already running;
- Allow creating missions without needing to have programming skills;
- Others that may be considered useful throughout the project's development.

Chapter 3

# System Requirements and Architecture

## 3.1 System Requirements

This section will present the system requirements specification. The following subsections begin with a description on the requirements elicitation process, followed by a context description, actors classification, use-case diagrams, non-functional requirements, functional requirements, and the system's assumptions and dependencies.

### 3.1.1 Requirements Elicitation

In order to collect the system requirements the team gathered firstly with its supervisors, professor Susana Sargento, professor Miguel Luís, professor Pedro Rito, and investigator Margarida Silva; And identified the already documented and undocumented problems. The following phase consisted in analyzing the State of the Art, particularly, looking for projects and researching for studies that might be similar to the one being developed. Thus, we were able to recognize some innovative ideas. Finally, a brainstorm was performed to assemble as much concepts as possible; After the triage and filtering process we proposed some solutions that helped us define the most important functional and non-functional requirements, that will be addressed later; In this process we sought to select the most clear, concise and testable solutions.

### 3.1.2 Actors

The targeted users of this project are those that may need to preform tasks such as disaster prevention, patrolling zones of difficult access, etc.. To use this system, the users should present a level of expertise that allows them to create missions, access history and view live missions. The main actors are described as follows:

- Security Forces: Security Forces can create missions, access history and view live missions. They use drones equipped with cameras and sensors that allow them to preform tasks such as surveillance and monitoring public gathering restrictions.
- Civil Protection: They are capable of creating missions, access history and view live missions. They use drones equipped with cameras and sensors that allow them to preform tasks such as patrolling difficult access zones and disaster prevention.
- Drone Admin: The Drone administrator is the user with the most authority within the system. He is capable of creating missions, access history, access internal and SOS logs and view live missions.

### 3.1.3 Use Cases

Figure 3.1 presents the use case model of whole system, the package represent the web application that will provide the exchange of information between the users and the drones.

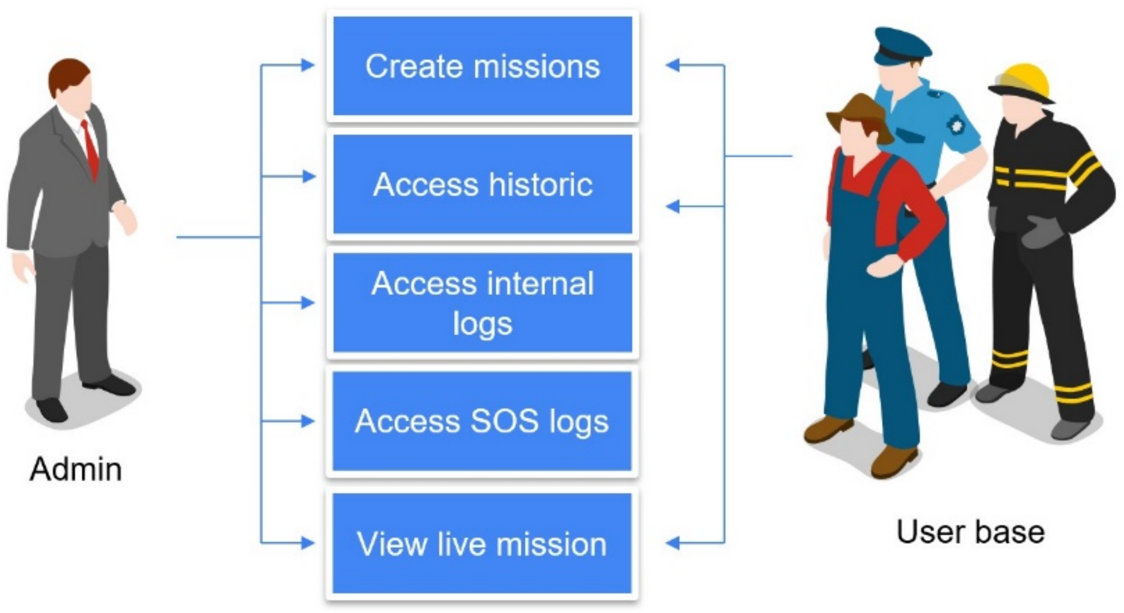

Figure 3.1: Use cases

### 3.1.4 Non-functional Requirements

- **Usability:** Creating and executing a mission must be intuitive as well as analysing the data. **Priority:** High
- **Efficiency:** The system must be capable of showing live data with low latency as well as update persistence date in windows inferior to 10 seconds. **Priority:** High
- **Capacity:** The system must capable of recording relevant historic of all deployed missions. **Priority:** High
- **Availability:** Since the system is used in a critical context its up time must be as close as possible to 100%. **Priority:** High
- **Security:** The dashboard as well as the OBU's must be secure and robust against attackers. **Priority:** Medium
- **Recoverability:** In the event of crashing (software or hardware) the UAVs must be easily recoverable and deployable. **Priority:** High
- **Maintainability:** The system must be constructed with future proofing in mind. **Priority:** Medium

### 3.1.5 Functional Requirements

The following list presents requirements specifying functional requirements:

- **Server running InfluxDB and Django API:** InfluxDB is key since it will allow for a quick and efficient access to all the available data, allowing the user to read as close to real time data as possible in a dashboard built in Django. Django has been chosen for being a powerful and robust web framework, allowing us to create the most complete dashboard possible, as well as giving ease to maintain it and/or modify it. **Priority:** Critical
- **APU running voyage in a docker image:** Voyage is a light weight distro based on Debian, giving us a familiar interface to work giving all the essential tools for a low cost, it was also was chosen for the fact of the playbooks we were given access are based on Debian, other light weight distros could have been chosen for the same effect. **Priority:** High
- **APU driving WAVE board:** One of the main points of this project is to ensure that the communications are in a reliable and fast manner, for that reason WAVE was chosen for it's high frequency, low setup time, for this reason all the modules that communicate (OBUs & groundstation) must be equipped and able to drive a WiFi WAVE enabled board. **Priority:** Critical
- **APU driving LoRa WAN board:** This board has three purposes, the main objective is to enable the APU to collect data that would be impossible for it reach for being to far away, the second objective is to relay information APU to APU functioning as a gateway until it reaches a groundstation, the third is to add a security measure, since in previous works it has been proven that a APU can go rogue and not answer even to it's owner remote controller, giving the user a way to track and collect the APU since it will function as a beacon broadcasting it's location. **Priority:** High
- **Ground station must have access to the internet:** The ground station is the man in the middle of all the missions, it's job is to collect the data from the APU's via WAVE,WiFi,LoRa and send it to the dashboard so it can be viewed and subsequently stored.
- **Easy to program missions interface:** A dashboard with the ability to create missions with the minimum knowledge of how to program needs to be a focus so the maximum amount of users can use it, this is possible by creating a drag and drop logic system like scratch, to create said missions, that will then be translated to a Groovy file and seamlessly loaded. **Priority:** High

### 3.1.6 Assumptions and Dependencies

Unfortunately regardless of how many counter measures and precautions we take there are always assumptions that have to be made and are out of our control, the following list illustrates all the assumptions that are made:

- The ground station has a constant connection to the internet, to mitigate this problem the groundstation will also have cellular internet, however we will always be dependent of a internet provider to connect the groundstation to the dashboard if it isn't being hosted locally.
- If UAV's crash or fail to return will be structurally stable enough and have enough power to start the SOS LoRa beacon.
- The server will have enough capacity to handle any peak of information sent by the groundstations and maintain the service stable.

## 3.2 System Architecture

### 3.2.1 Domain Model

The domain model on Figure 3.2 describes the relations between the entities within our system.

When looking at those entities, we begin with two that preform the manual interactions. One is the Admin, that has access to the Groundstation and can directly setup and deploy drones. The other is a User that represents anyone that wants to interact with a Drone through the Dashboard. The former is not mandatory for the system to properly function, as long as there is a drone connected to the Groundstation, making it a zero or more to many relationship, whereas the latter is a many to one relationship (at least one User using a single platform).

The Groundstation is the heart of the system because it talks to both Drones and Dashboard. It basically fetches the telemetry from the Drones and makes it available to the User through the interface in the Dashboard.

The data that the Drone transmits consists of information from the Drone itself, and also information from the RSUs that it talks to, if that's the case. Drones can also talk to each other to preform "Relay", which works as a sort of connection boost. Drones don't need the RSUs to neither function nor send data so their relationship is of zero or more to many.

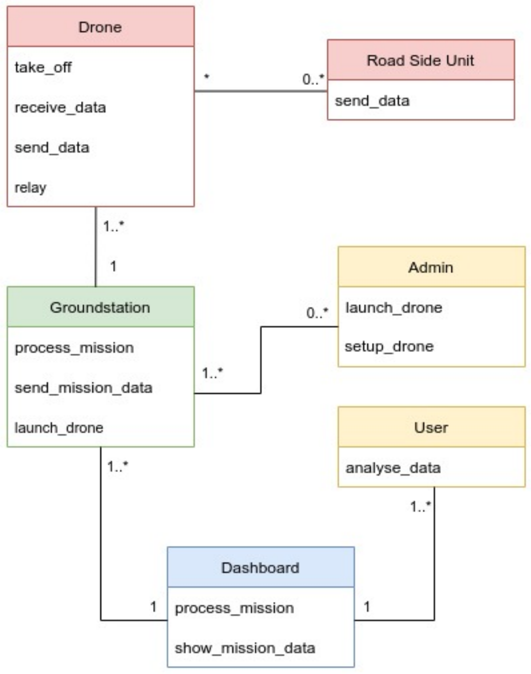

Figure 3.2: Domain Model

### 3.2.2 Deployment Diagram

The diagram on Figure 3.3 is intended to explore the hardware and software of our system.

The Drone module is represented by two nodes in the diagram, the Endpoint and the Gateway. This happens because a drone can both send its own information directly to the Groundstation, or work as a bridge for other drones to send their information. This module is equipped with an APU running Voyage Linux, communicates using WAVE, and uses LoRaWAN to talk to the Internet. It can also have multiple sensors to collect various types of additional data or assist navigation during missions.

The Groundstation also uses LoRaWAN and WAVE to communicate with the drones and retrieve their data.

The data from drones reaches the user through an API built in Django. This application is meant to facilitate the flow of information from the Groundstation to both the database and the dashboard.

The data persistence is done with InfluxDB, because of its TSDB characteristics, which makes sense for this project.

The user is capable of keeping track of the drone behaviour and telemetry through a dashboard built as an SPA with Nuxt.js, CSS and HTML.

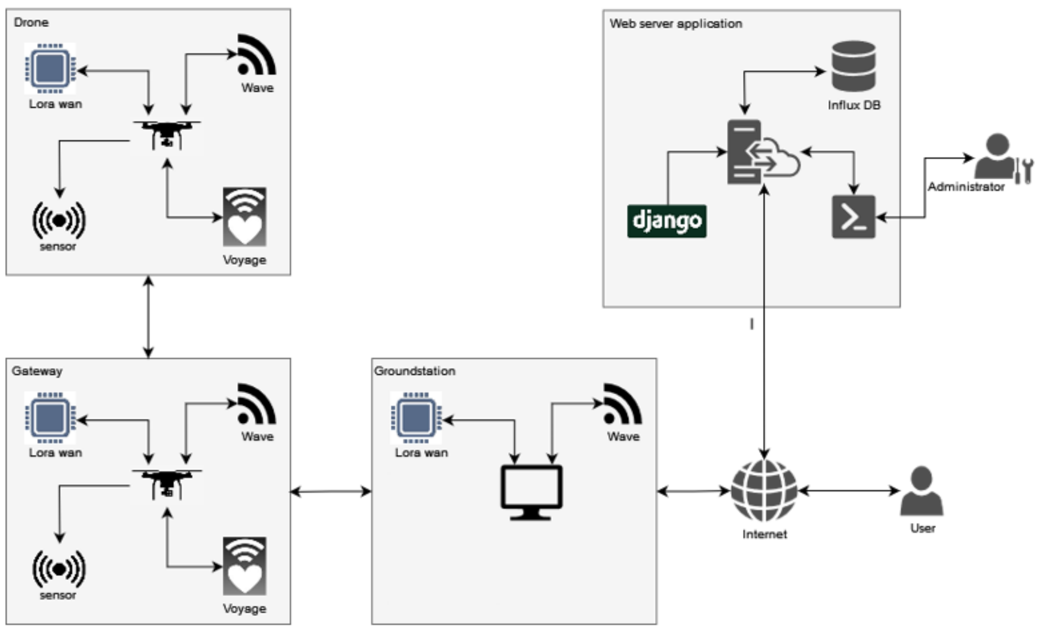

Figure 3.3: Deployment Diagram

# References

[1] Wikipedia contributors. “Unmanned aerial vehicle — Wikipedia, the free encyclopedia”. (2021), [Online]. Available: `https://en.wikipedia.org/w/index.php?title=Unmanned_aerial_vehicle&oldid=1062928707`. (accessed: 09.01.2022).

[2] M. Silva, *A mission planning framework for fleets of connected drones*. 2021, (accessed: 01.12.2021).

[3] A. Figueiredo, *Mobility sensing and V2X communication for Emergency Services*. 2021, (accessed: 01.12.2021).

[4] LoRa Alliance. “What is LoRaWAN® Specification”. (2021), [Online]. Available: `https://lora-alliance.org/about-lorawan/`. (accessed: 01.12.2021).

[5] R. Kiefer. “TimescaleDB vs. PostgreSQL for time-series”. (2021), [Online]. Available: `https://blog.timescale.com/blog/timescaledb-vs-6a696248104e/`. (accessed: 02.12.2021).

[6] Nuxt Team. “Nuxt - The Intuitive Vue Framework”. (2021), [Online]. Available: `https://nuxtjs.org/`. (accessed: 02.12.2021).

[7] D. Amaral, G. Pereira, J. L. Costa, and J. T. Rainho, *Drone Relay - Technical Report*. 2020/2021, (accessed: 03.12.2021).