الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et populaire
وزارة التعليم العالي والبحث العلمي
Ministère d'Enseignement Supérieur et de Recherche Scientifique
جامعة عمار ثليجي الأغواط
UNIVERSITÉ AMAR TELIDJI

كلية العلوم
FACULTÉ DES SCIENCES
قسم الاعلام الالي
Département d'Informatique

Design and Development of an Unmanned Aerial Vehicles Mobility Simulator

---

Réalise par

SAHLI Mohammed Youcef

MOKHTARI Youcef

Proposé par

BENKOUIDER Sara

LAGRAA Nasreddine

2022

Acknowledgment

First of all, I thank Allah our Creator, for giving us the strength, will, and courage to do this humble work.

Many thanks to our supervisor for providing advice and guidance from start to finish for this work.

We would like to thank everyone who contributed directly or indirectly to the completion of this work.

Finally, We would like to express our deep gratitude to our family who have always supported us and to everyone who has participated in completing this thesis. As well as all the teachers who contributed to our education.

Dedicate

I dedicate this work to two people who never stop sacrificing themselves for my success.
To my dear mother and father who supported me throughout my school years.
And to all my family who helped me all my life. To all my dear friends and colleagues.
And for everyone who taught me throughout my school life.
And to my classmates .
thank you all.

.....

Sahli Mohammed Youcef.

Dedicate

That work is not only mine it is for all people that they made me feel better and remaind me that there is another chance for exploiting, so for my parents and specifically my uncle abde lmalek and my aunt lalia, and all teachers that they help me to do that work and friends on real worlds and virtual worlds on Facebook like MA,kanka and group house ... etc, thank you a lot you made me to do all that may allah give you all the best.

.....

Youcef Mokhtari.

ملخص

شهدت المركبات الجوية غير المأهولة ، والمعروفة أيضًا باسم الطائرات بدون طيار ، التي تم استخدامها وتطويرها في البداية في المجال العسكري ، تغييرات عميقة في السنوات الأخيرة وتستخدم بشكل متزايد في المجال المدني. يتم استخدامها في مكافحة الحرائق والإنقاذ وكذلك في تطبيقات محددة مثل المراقبة والهجوم. رحلة الأسطول هي الأكثر استخدامًا لأنها تسمح بتوزيع حكيم للمهام وتحسن بشكل كبير من كفاءة الطائرات بدون طيار.

التجارب الحقيقية باستخدام الطائرات بدون طيار مكلفة ، لذلك يجب تحليل أداء أنظمة الطائرات بدون طيار قبل نشرها. وفقًا لذلك ، طور الباحثون ومهندسو البرمجيات العديد من أدوات المحاكاة والبيئات والأطر لتقييم أنظمة الطائرات بدون طيار. في هذا السياق ، تسلط هذه الدراسة الضوء على أنسب أجهزة المحاكاة لرحلة الطائرات بدون طيار وتحددها.

في هذا المشروع ، قمنا بتطوير محاكي طائرات بدون طيار بهدف محاكاة طيران أسطول طائرات بدون طيار. من خلال المحاكاة الخاصة بنا ، يمكننا إنشاء ملف التنقل لمساعدة الباحثين وللتحليل.

الكلمات الدالة :

المركبات الجوية بدون طيار ، المحاكي ، رحلة الأسطول ، طائرة بدون طيار ، رباعيةالمروحيات ، محاكاة ، رحلة مستقلة ، تتبع المسار.

Résumé

Les véhicules aériens sans pilote, appelés aussi drones, utilisés et développés initialement dans le domaine militaire, ont connu de profondes mutations ces dernières années et sont de plus en plus utilisés dans le domaine civil. Ils sont utilisés pour la lutte contre les incendies, le sauvetage ainsi que dans des applications spécifiques telles que la surveillance et l'attaque. Le vol en flotte est le plus utilisé car il permet une répartition judicieuse des tâches et améliore grandement l'efficacité des drones.
Les expériences réelles utilisant des drones sont coûteuses, par conséquent, les performances des systèmes de drones doivent être analysées avant leurs déploiements. En conséquence, les chercheurs et les ingénieurs en logiciel ont développé plusieurs outils, environnements et cadres de simulation pour l'évaluation des systèmes UAV. Dans ce contexte, cette étude met en évidence et identifie les simulateurs les plus adaptés au vol des drones.
Dans ce mémoire, nous avons développé un simulateur de drone dans le but est de simuler le vol d'une flotte de drones. Avec notre simulateur, nous pouvons générer un fichier de navigation pour aider les chercheurs et pour l'analyse.

Mots clés: UAV, Simulateur, Vol de flotte, Drone, Quadricoptère, Simulation, Vol autonome, Suivi de trajectoire, flotte de drones.

Abstract

Unmanned aerial vehicles, known also as drones, used and developed initially in the military field, have experienced profound changes in recent years and are increasingly used in the civilian field. They are used for firefighting, rescue as well as in specific applications such as surveillance and attack. The fleet flight is the most used because it allows a judicious distribution of the tasks and greatly improves the efficiency of the drones.

Real experiments using UAVs are costly, therefore, the performances of UAVs systems should be analyzed before their deployments. Accordingly, researchers and software engineers developed several simulations tools, environments and frameworks for UAV systems evaluation. In this context, this study highlights and identifies the most suitable simulators for UAVs flight.
In this project, we have developed a drone simulator with the aim of simulating the flight of a drone fleet. With our simulation, we can generate the navigation file to help researchers and for analysis.

Keywords: UAV, Simulator, Fleet Flight, Drone, Quadcopter, Simulation, Autonomous flight, Trajectory tracking.

Contents

List of Figures x

List of Tables xii

Introduction 1

1 Chapter 1 : Unmanned Aerial Vehicles (UAVs) 2
- 1.1 Introduction . . . 2
- 1.2 Unmanned Aerial Vehicle System (UAVS) . . . 2
  - 1.2.1 Ground Control Station . . . 3
  - 1.2.2 Sensors . . . 4
  - 1.2.3 Payloads . . . 5
  - 1.2.4 Storage . . . 5
  - 1.2.5 Power Management . . . 5
  - 1.2.6 Electric Speed Controllers (ESCs) . . . 6
  - 1.2.7 Flight Controller . . . 6
- 1.3 Fleet of UAVs . . . 9
  - 1.3.1 Fleet control strategies . . . 10
  - 1.3.2 Drone's fleet communication architectures . . . 11
- 1.4 Types of UAVs . . . 13
- 1.5 Applications . . . 15
- 1.6 Conclusion . . . 16

2 Chapter 2 : UAVs software components and simulators 18
- 2.1 Introduction . . . 18
- 2.2 UAV software architecture . . . 18
- 2.3 Flight control firmware . . . 19
  - 2.3.1 PX4 . . . 19
  - 2.3.2 ArduPilot . . . 20
  - 2.3.3 PX4 vs ArduPilot . . . 21
- 2.4 MAVLink . . . 22
- 2.5 Ground Control Station (GCS) . . . 23
  - 2.5.1 QGroundControl . . . 23
  - 2.5.2 Mission Planner (MP) . . . 24
  - 2.5.3 MAVProxy . . . 25
  - 2.5.4 Comparison of ground control stations . . . 26
- 2.6 Unmanned Aerial Vehicle simulators . . . 26

2.6.1 Drone fleet simulators . . . 27
2.6.1.1 Gazebo . . . 27
2.6.1.2 Microsoft AirSim . . . 28
2.6.1.3 JMavSim . . . 29
2.6.2 Multiplayer simulators . . . 30
2.6.2.1 X-Plane . . . 30
2.6.2.2 Flight Gear . . . 31
2.6.3 Comparison of UAVs simulators . . . 32
2.7 Conclusion . . . 33

3 Chapter 3 : UAVs Mobility Simulator (UAVMS) 34
3.1 Introduction . . . 34
3.2 Development environment . . . 34
3.3 Architecture system of UAVs Mobility Simulator (UAVMS) . . . 35
3.3.1 UAVs Mobility Simulator navigation model . . . 36
3.3.1.1 Quadcopter Dynamics . . . 36
3.3.1.2 Energy model . . . 37
3.3.1.3 Fleet control strategy in the UAVMS . . . 37
3.3.1.4 Mobility modes . . . 38
3.3.1.5 Navigation files . . . 39
3.3.2 Data entry interfaces . . . 40
3.3.3 UAVMS analyzer . . . 44
3.3.4 Filter . . . 44
3.3.5 Final Results Interfaces . . . 44
3.3.6 Network Simulator (NS2) support . . . 48
3.4 Conclusion . . . 49

General Conclusion 50

Bibliography 51

Use case diagram of UAVs Mobility Simulator 55

Class diagram of UAVs Mobility Simulator 56
.1 First section . . . 56
.2 Second section . . . 56

List of Figures

1.1 Unmanned Aircraft System (UAS) . . . 3
1.2 Drone Ground controller . . . 4
1.3 Autopilot Pixhawk flight controllers . . . 6
1.4 Flight Controller Architecture . . . 7
1.5 Drones fleet . . . 9
1.6 Fleet control strategies . . . 10
1.7 Drone's fleet communication architectures . . . 12
1.8 Applications of UAVs . . . 16

2.1 UAV firmwares architecture . . . 19
2.2 PX4 logo . . . 19
2.3 ArduPilot logo . . . 20
2.4 MAVLink (Micro Air Vehicle Link) . . . 22
2.5 Vehicle mission in QGroundControl . . . 23
2.6 Drone mission in mission planner . . . 24
2.7 MAVProxy running under Ubuntu . . . 25
2.8 Classification of drone simulators . . . 26
2.9 Multi UAV Simulation with Gazebo . . . 27
2.10 Microsoft AirSim simulator . . . 28
2.11 JMavSim simulator . . . 29
2.12 X-Plane simulator . . . 30
2.13 Flight Gear simulator . . . 31

3.1 Architecture system of our simulator (UAVMS) . . . 35
3.2 Example of a mission JSON file . . . 36
3.3 Quadcopter Control Motor Speeds . . . 37
3.4 Fleet Control Strategy On Mobility simulator . . . 38
3.5 Mobility modes of UAVMS . . . 38
3.6 Example of a log.txt file . . . 39
3.7 Example of a log.csv file . . . 39
3.8 Example of a scenario.tcl file . . . 40
3.9 The main simulation interface . . . 41
3.10 The mission creation interface . . . 41
3.11 Determine the path on the map . . . 42
3.12 Determine the zone on the map . . . 42
3.13 Determine the start and end points on the map . . . 43
3.14 Define the parameters of the drone . . . 43

3.15 Random Walk Model (Fleet) analyse 44
3.16 File Navigation Filter 45
3.17 Scenario of one drone (trajectory tracking) 45
3.18 Scenario of fleet in form of line (trajectory tracking) 46
3.19 Scenario of fleet in form of circle (trajectory tracking) 46
3.20 Scenario of one drone (random walk model) 47
3.21 Scenario of fleet in form of circle (random walk model) 47
3.22 Scenario of each drone individually (random walk model) 48
3.23 Running The Navigation File In NS2 48

24 Use case diagram 55

25 Class diagram of UAVs Mobility Simulator GUI section 56
26 Class diagram of UAVs Mobility Simulator mobility section 57

List of Tables

1.1 Summary of measurements and statistics for representative channels . . 9
1.2 Classification based on the structure of the UAV 14
1.3 Comprehensive classification of UAVs 15

2.1 Comparison between PX4 and ArduPilot 22
2.2 Comparison between GCSs 26
2.3 Comparative analysis of UAV simulators 33

Introduction

Unmanned aerial vehicles (UAVs), commonly known as drones, have been the subject of concerted research over the past few years, owing to their autonomy, flexibility, and broad range of application domains. Indeed, UAVs have been considered as enablers of various applications that include military, surveillance and monitoring, telecommunications, delivery of medical supplies, and rescue operations.
Drones can provide reliable, cost-effective and time-efficient solutions to a variety of real-world scenarios. In addition, the mission can be divided into a group of drones or send a fleet of drones to take care of a mission such as search and rescue. This exciting new avenue for the use of UAVs warrants a rethinking of the research challenges with fleet control and navigation being the primary focus.

Developing and establishing scientific research on the drone itself consumes many resources, one mistake that can consume large financial and human resources. For this reason, it has become important to have a simulator that supports the fleet's flight of a drone for scientific research.
The goal of this project is to design and develop a simulator that supports the fleet's flight for a drone.
This thesis includes three chapters. In the first chapter a study on UAVs types, classifications, applications, systems, models, communication techniques, fleet control strategies. In the second chapter, some of the most important existing UAVs simulators were discussed, in addition to some open source flight controllers and ground control software. In the third chapter, our simulator was designed, developed and presented.

Chapter 1

Unmanned Aerial Vehicles (UAVs)

1.1 Introduction

Drones are becoming increasingly popular due to their diverse nature and the potential to solve various real-life challenges in various fields and industries. Drones can reach places and situations that are dangerous to humans and impossible for conventional tools and vehicles to reach. It can also fly at high altitudes in dangerous areas such as volcanoes, or low in urban areas, depending on the type of drone. Drones can adapt according to the circumstances, the type of aircraft and the mission which increases their value, since the drones operate without a pilot on board.
This chapter gives some necessary background on UAV technology and aerial robotics, their types and domain of application, in addition to the fleet formation flight concept, and the communication between the UAVs with Ground Control Station or other UAVs in the fleet.

1.2 Unmanned Aerial Vehicle System (UAVS)

The diagram below in Figure 1.1 provides a overview of the building blocks of Unmanned Aircraft System, such as Ground Control Station (GCS) and the Communication systems and the power management to the flight controller and payloads and the other sensors and component.

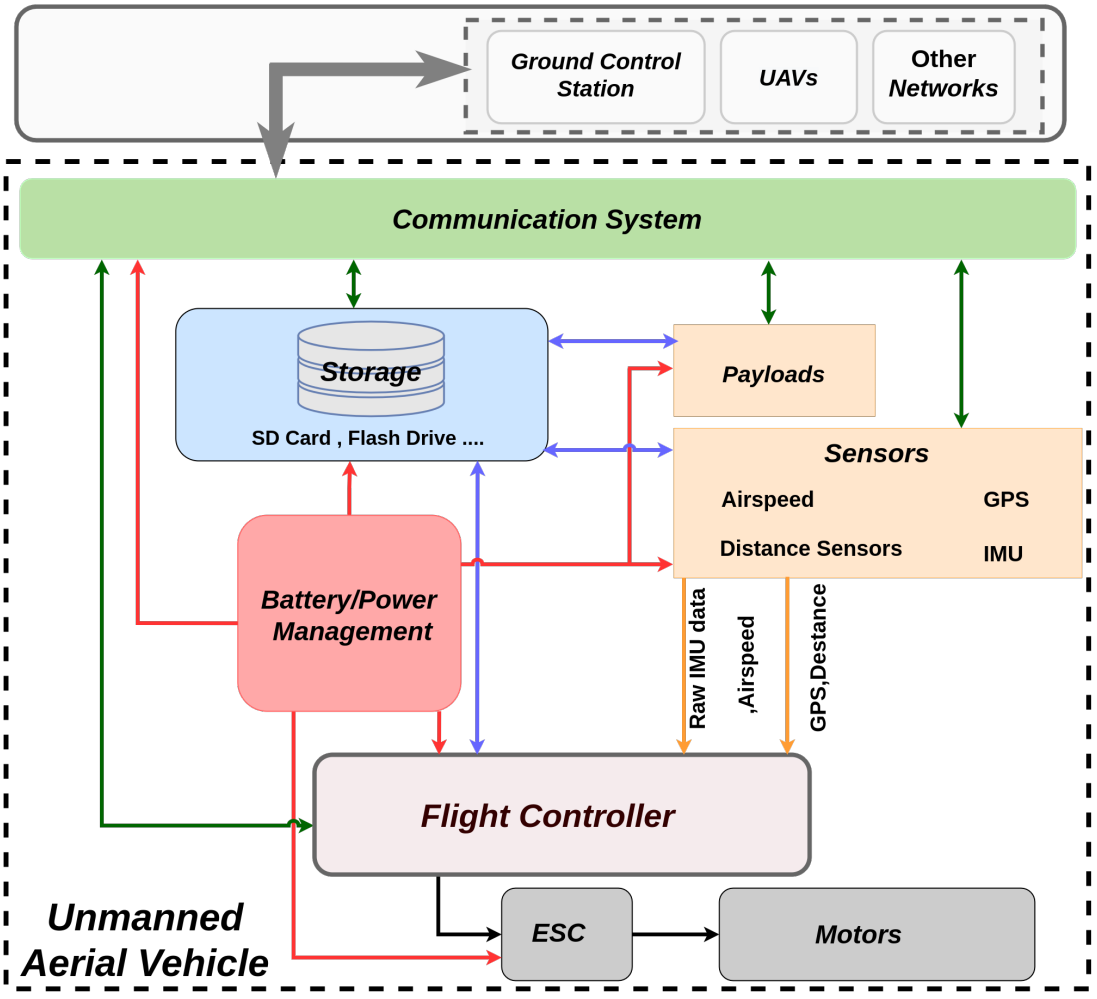

FIGURE 1.1: Unmanned Aircraft System (UAS)

1.2.1 Ground Control Station

The ground control station is a remote control device for the UAV and can also be software in any operating system, as shown in Figure 1.2. Its main responsibilities include:

- View and monitor flight status data in real time.
- Displaying navigation view.
- Displaying images received from the video receiver.
- Intervene and make UAV-related decisions, mission planning, and specific operation if required, Sending real-time commands to the avionic system.
- Facilitating the users and pilots in the automatic control, especially in unexpected occasions such as an emergency landing, and Flight data recording as a backup to record data on board.

FIGURE 1.2: Drone Ground controller [1]

The most obvious advance of GCS development in small scale UAV field is the increasing prevalence of the open source GCS software toolkits. The most successful softwares are Mission Planner, QGroundControl and MAVProxy, which will be studied in the next chapter.

1.2.2 Sensors

Sensors use to determine vehicle state (needed for stabilization and to enable autonomous control). The vehicle states include: position/altitude, heading, speed, airspeed, orientation (attitude), rates of rotation in different directions, battery level, etc [2].
The system minimally requires a gyroscope, accelerometer, magnetometer (compass) and barometer. A GPS or other positioning system is needed to enable all automatic modes, and some assisted modes. Fixed wing and VTOL-vehicles should additionally include an airspeed sensor (very highly recommended).
The minimal set of sensors is incorporated into Pixhawk Series flight controllers (and may also be in other controller platforms). Additional/external sensors can be attached to the controller [2].

GPS : The GPS driver is responsible for the position publication read from the GPS sensor. Depending on the sensor vendor, it may support different protocols. GPS values are directly used for the position and attitude estimation, without passing trough the sensors [3].

Airspeed : Airspeed is correlated with the wind speed estimation, necessary for stability and control purposes. Before being processed, it needs to be corrected, that is why it is sent to the sensor hub [2].

Distance Sensors : Distance sensors are generally laser or ultrasonic, a light or sound impulse is sent from the transmitting apparatus to the outside of the drone, in the desired direction, and the Time Of Flight of the signal bouncing back on the eventual barrier surface determines the distance from the object to be avoided [4].

IMU : An inertial measurement unit (IMU) is a device that integrates multi-axes, accelerometers, gyroscopes, and other sensors to provide estimation of an objects orientation in space. Measurements of acceleration, angular rate, and attitude are typical data outputs [5].

IMU output are necessary for the position and attitude estimation as well as the autonomous flight, position controller and attitude and rate controller. Nevertheless, raw data must be adapted before being used for the estimation.

1.2.3 Payloads

Depending on the type of aircraft and its field of application or the type of mission, some payloads can be added, for example: a packages for delivery or high-quality camera for search, surveillance and rescue missions.
Whenever willing to control a camera (or any payload) connected to the vehicle, it is necessary to specify how Flight Controller can interact with it. One of methods is using MavLink to get the input and output [6] .

1.2.4 Storage

Missions and plans are physically stored within the drone internal memory. example , plans are may stored in JSON files containing basic information about mission goals and step to be performed (Home position, Rally Points, Geo-fence, Waypoints and mission commands) [5]. Moreover, SD card allows to store captured images/video to be elaborated afterwards.

1.2.5 Power Management

The energy consumed is affected by acceleration, weight, engine speed, wind speed and many other variables, and the battery may also be affected by the temperature in the

outside environment.
The energy consumed is also affected by the communication characteristics between the drone and other wireless devices (frequency, bandwidth...). In [7, 8] the energy consumption of communications in unmanned aircraft.

1.2.6 Electric Speed Controllers (ESCs)

Many drones use brushless motors that are driven by the flight controller via an Electronic Speed Controller (ESC) (the ESC converts a signal from the flight controller to an appropriate level of power delivered to the motor). Each one of them has an output for a motor and an input for a battery [9] .

1.2.7 Flight Controller

A flight controller is a circuit board that manages the drone's flight, control the power, obstacle avoidance..etc,

Exemple Pixhawk Autopilot in Figure 1.3.

(a) PIXHAWK (b) Pixfalcon (c) Pixhawk 2

FIGURE 1.3: Autopilot Pixhawk flight controllers [10]

Everything a flight controller does can be classified within one of three categories: Sensing, controlling, and communicating [5] .

Perception (sensing) The flight controller filters a lot of this information and fuses some to get more efficient and precise information The flight controller is connected to a set of sensors. The drone filters a lot of this information and fuses some to get more efficient and precise information in the sensors hub module.

Controlling Aside from sensing what's going on, a flight controller unsurprisingly controls the motion of the drone. The drone can rotate and accelerate by creating speed differences between each of its four motors. The flight controller uses the data gathered by the sensors to calculate the desired speed for each motors. The flight controller sends this desired speed to the Electronic Speed Controllers (ESC's), which translates into a signal that the motors can understand.

Calculating the movements, fusing and filtering the sensory information, and estimating the safety and durability of a flight is all done by an algorithms.

Communicating An key part of a flight controller is communications management. Part of the sensor's job is to provide information that must be clearly translated for the pilot to read, which means efficiently. An obvious thing to communicate is its battery level, which can decide if a pilot wants to fly further or return to the charge.Flight controllers need to communicate also with other computer systems and UAVs.

The following figure 1.4 shows some of the flight controller components that we may need and the relationships between them:

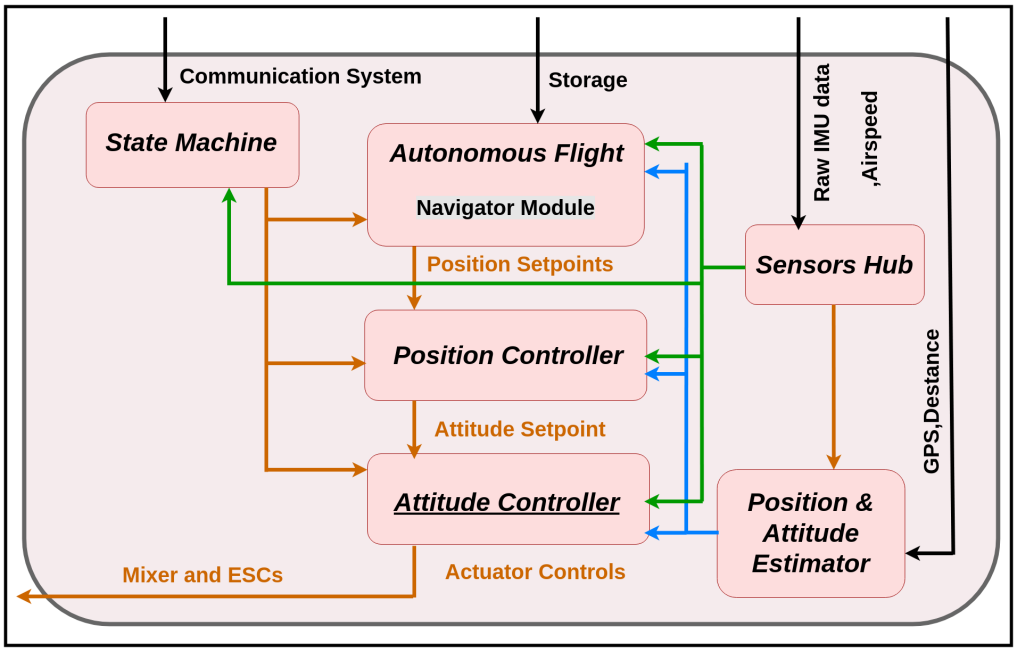

Figure 1.4: Flight Controller Architecture

A. Sensors Hub The Sensors Hub module assumes a key role to the entire system. It takes low-level output from drivers, turns them into a more valuable form (filtering), and publishes them, letting the other modules to take benefit of the clean measurements [11] [5] .

B. State machine State module is used to change the drone state, for instance flight modes, or to perform basic operations like taking off or land. It is able to receive input both from the sensor hub and the communication model. Data coming from the commander module and position and attitude estimators are then used as inputs for the autonomous flight, position controller and attitude controller [11].
Flight Modes define how the autopilot responds to user input and controls vehicle movement. They are essentially grouped into three categories: manual, assisted or autonomous modes, depending on the level of control the autopilot should produce. The pilot transitions between flight modes may be performed using switches on the remote control or with a ground control station [5].

C. Autonomous Flight The autonomous flight or navigator module, based on the commander input, the mission plan (SD card) and the estimated states, creates position setpoints. Such a setpoints are the inputs for the position controller. The position controller, creates attitude setpoints, that are used by the attitude and rate controller to create the actuator controls [11] [5]. Actuator controls are then converted into suitable values to be applied to the actuators.

D. Mixer Mixer takes force commands (example: turn right) and translates them into individual motor commands, while ensuring that some limits are not exceeded. This translation is specific for a vehicle type and depends on various factors, such as the motor arrangements with respect to the center of gravity, or the vehicle's rotational inertia [11].

E. Estimator Estimator takes one or more sensor inputs, combines them, and computes a vehicle state (for example the attitude from IMU sensor data) [12] .

F. Position Controller In this controller, the deviation from the desired path is calculated (in the body frame) at every instant.
Using this, the desired the angles are calculated. This is the underlying principle of this controller.

G. Attitude Controller Controlling vehicle attitude requires sensors to measure vehicle orientation, actuators to apply the torques needed to re-orient the vehicle to a desired attitude, and algorithms to command the actuators based sensor measurements of the current and desired attitude.

H. UAV Communication Model Various ground terminals can connect with UAVs, such as ground station (GS), user equipment (UE), vehicles, Internet-of-Thing (IoT) node . Besides, UAV serving as an aerial base station (ABS) can assist or substitute a terrestrial base station (TBS) in specific situations. Thus, UAVs are playing an important role in a slice of arising communication.

The characteristics of drones raise some major challenges in channel modeling, as

Table 1.1: Summary of measurements and statistics for representative channels [13]

UAV Altitude	Frequency Selectivity	Terminal	Environment
580-800 m	WB,968-5060 MHz	GS	Over-water
504-924 m	WB,968-5060 MHz	GS	Suburban,Near-urban
4-16 m	WB, 3.1-5.3 GHz	GS	Open-field,Suburban
15-100 m	WB,2.5 GHz	Cellular	Suburban
15-120 m	NB, 800 MHz	Cellular	Suburban
0-50 m	NB, 2.4 GHz	UAV	Suburban
6-15 m	WB, 60 GHz	UAV	Urban

WB: Wideband, NB: Narrowband

compared to traditional cellular or vehicle communications. the key factors are not only related to external factors including frequency, environment, and weather, but also dependent on internal characteristics of UAVs and ground devices. We highlight key challenges as follows, in Table 1.1.

1.3 Fleet of UAVs

Figure 1.5: Drones fleet [14]

Drones can collaborate to create and fly as a fleet (Cf. figure 1.5), the fleet can be coordinated according to the role of each drone or a whole group of drones. In the case of coordination, each drone has certain tasks to accomplish in such a way that there is a kind of synchronization between them that respects the order of the tasks. In the case of cooperation, which requires strong spatial and temporal coordination between all the personnel in the fleet, the fleet is formed from many drones in order to achieve a specific mission for the entire fleet, because only one drone does not have the ability or time to

complete the required task.
A single drone with the performance and characteristics required to carry out some tasks may be expensive compared to many low-cost drones that perform the same task. Here comes the advantage of the drone fleet because small sizes of aircraft can be used at the same time, as the burden is distributed Task such as mission planning, data processing, and area control over the entire team, reducing cost and time to perform the task.

The fleet of drones can fly in small areas at high speed, but this exposes them to hardware failures, which affects the safety of the flight. This problem is one of the biggest disadvantages of the fleet's flight. To solve this problem, we must eliminate, replace or repair any member who fails according to the situation because it is no longer able to synchronize and coordinate with the rest of the fleet and affects the performance of the fleet and the completion of the mission, so there must be strategies and algorithms to control the flight and coordination of the fleet.

1.3.1 Fleet control strategies

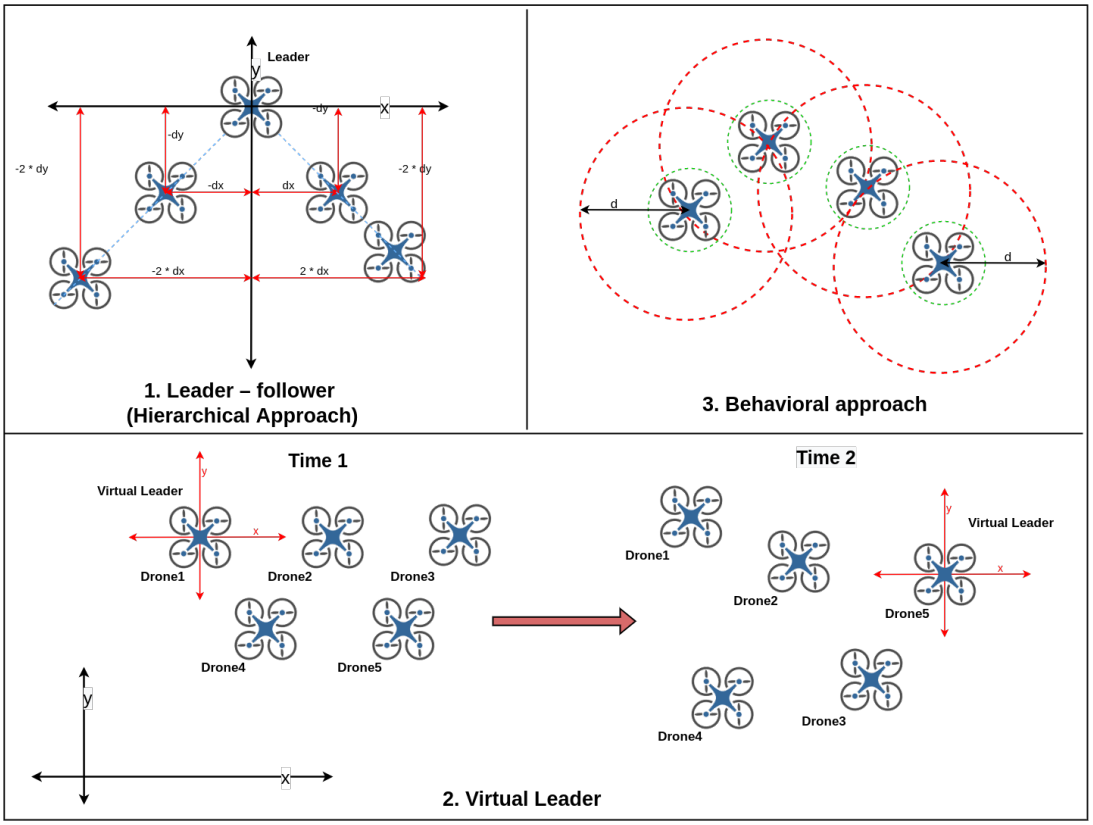

Figure 1.6: Fleet control strategies

Different fleet formation control strategies exist in the literature [15] and this report discusses three of them:

Leader – follower (Hierarchical Approach): (Cf. figure 1.6) This approach is widely used for the multi-agent system, where One drone is the pilot and decides the course and speed of the fleet, followed by the rest of the fleet members [16]. One of the disadvantages of this strategy is when the leader is lost or affected by failure due to external or climatic conditions or related to the drone itself, which will affect the fleet and the entire mission and expose the rest of the drones to great damage [17–19].

Virtual Leader : This approach solves the problem of the commander and followers approach, whereby replacing the fleet leader with a virtual one (Cf. figure 1.6). All members follow the mission path set by the virtual leader, just like the approach of the leader and followers. The main difference between the two approaches is that the virtual leader assigns virtual leaders in the event of an error or failure, which maintains the optimal formation under different maneuvers and conditions [19, 20].

Behavioral approach (Decentralized approach): Each agent follows specific rules in order to perform group behavior [19, 21, 22] .

These rules are:

- Avoid collision with neighbors: Everyone in the fleet must ensure a predetermined security distance with their neighbors (Cf. figure 1.6). It can include a flight controller built into each drone or other additional controller. This control device generates an alert when the distance with the neighbors becomes less than the safety distance or beyond the range of the neighbors.
- Speed matching with neighbors: The speed of all members of the fleet must match, this is done either by using a set of sensors located inside each drone or by using communications between fleet members.
- Each member must stay close to its neighbors and maintain formation using distance and location sensors, this is done by setting a general target for the fleet which can be a meeting point or reference path known by all members.

The behavioral approach represents a self-organizing structure that maintains formation in various conditions and challenges, is easy to maintain, and relies on many sensors, algorithms, and rules for members to follow.

1.3.2 Drone's fleet communication architectures

Different architectures could be used to ensure the communication between drones and between drones and GCS (Ground Control Station)[23] ,summarized in Fig 1.7.

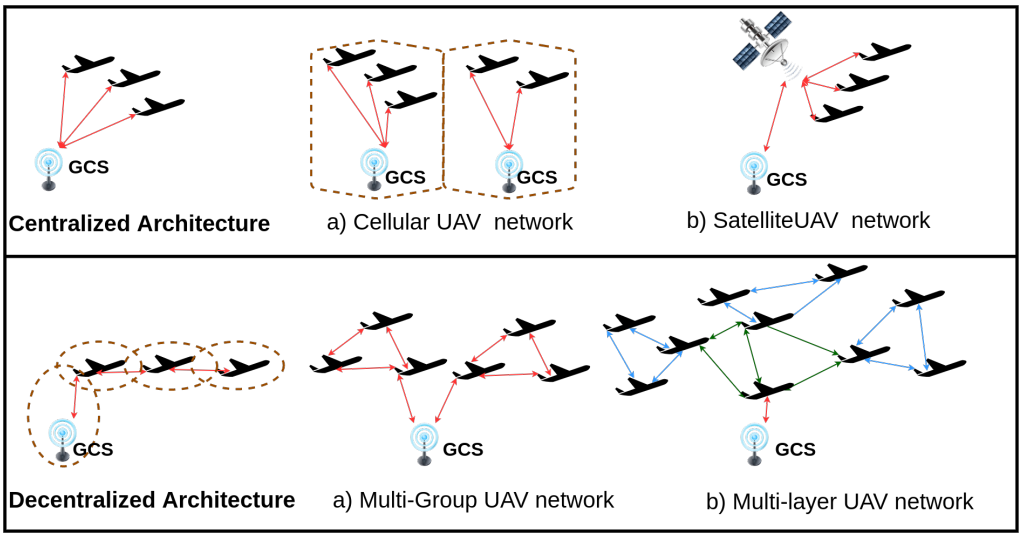

FIGURE 1.7: Drone's fleet communication architectures

1. Centralized Architecture In the centralized architecture, all the drones in the fleet are not connected to each other and instead connect to the GCS, making the only connection they have in common is the GCS which is the central node of the network. All orders, information and sensitive data are sent and received between UAVs via GCS, which adds delays in messaging and synchronization between the fleet. [24] (Cf. figure 1.7).

Since drones fly long distances to accomplish their mission, communication between drones and GCS requires a high transmission rate. Because of the limitations of small or medium drones in the size and capacity of payloads, the addition of advanced wireless transmitters can be a problem, in addition to the fact that the central structure lacks durability because the central architecture constitutes a weak point in the center so that if the center fails, the result will affect the entire network and the connection will be disturbed or even disconnected.

A. Cellular network architecture (Semi-centralized) The cellular communications network relies on the infrastructure of the existing mobile operators, dividing the area into different zones with the base station in each zone responsible for managing a group of drones (Cf. figure 1.7). The advantage of this structure is low-power transmitters, but it is difficult to cover all areas and maintain this infrastructure, especially in harsh climatic conditions or after natural disasters [19, 25].

B. Satellite communication architecture Satellite communication is a potential solution to ensure communication between two distant nodes (Cf. figure 1.7). Using a satellite communication has a negative effect, since it causes latency

in transmission and the signal could be dropped because of obstacles such as trees or mountains [19].

2. **Decentralized Architecture** One of the defects of the central architecture is the single point of failure and the delay of communication. On the contrary, the decentralized structure allows the drones to communicate with each other (directly or indirectly) without going through a single central point (GCS, for example), which eliminates the problem of a single point of failure and increases the durability of the structure. And increases the power of the transmitter and the speed of communication [19, 26].
The UAV Ad Hoc Network, commonly known as UAANET, is the most popular ad hoc network system, consisting of one or more groups of UAVs with one or more base stations.
Considering the ability to fly one group or several far from each other, the UAVs can act as a base station or relay for information transmission due to their low weight and the possibility of adding a low cost transceiver for each UAV.
Furthermore, the case wherein different types of UAVs in the network can be divided in two distinct communication architectures: multi-layer UAV ad hoc network and multi-group UAV network [19].

   A. Multi-Group UAV network This network architecture is suitable for missions that require a large number of UAVs and many groups with different flight characteristics so that communications within the group follow the same principle as a dedicated UAV network (Cf. figure 1.7). This structure lacks strength because communication between groups depends on communication between the leader of each group with the base station [19].

   B. Multi-layer UAV network In this architecture there is only one UAV that communicates directly with the GCS so that several layers are identified for several groups of diverse UAVs (Cf. figure 1.7). The bottom layer includes the drones in a group consisting of a UAV ad hoc network. The upper tier includes group leaders for drones [19].

1.4 Types of UAVs

Drones are probably mostly recognized for describing solutions to specific problems. And with many researchers and manufacturers willing to build new drones to support daily life, Having a systematic classification allows operators to choose a device that meets their specific requirements. Naturally, depending on the application and goals, one needs to use an appropriate type of UAV that can meet various requirements imposed by the

sought quality-of-service (QoS) and the nature of the environment.

UAVs can be categorized, based on flying mode (lighter than air, heavier than air) or type of gear :fixed-wing, rotary wings and hybrid systems.

Compared to rotary-wing UAVs fixed-wing UAVs such as small aircrafts have more weights, higher speed, and they need to move forward in order to remain aloft. In contrast, rotary-wing UAVs such drones and multirotor drones, can hover and remain stationary over a given area.Hybrids drone types merge the benefits of fixed-wing and rotor-based designs. This drone type has rotors attached to the fixed wings, allowing it to hover and take off and land vertically. This new category of hybrids are only a few on the market, but as technology advances, this option can be much more popular in the coming years [27] [28]. Table 1.2 shows the pros and cons of each type.

Table 1.2: Classification based on the structure of the UAV

		Advantages	Disadvantages	Technical Uses
Lighter Than Air		long-endurance steady and vibration-free. does not need fuel to fly	low-altitude and slow. Expensive.	High-resolution images at low cost Scientific surveys, meteorological observations, and militarysurveillance
Heavier Than Air	Multi-Rotor	Ease of use.VTOL and hover flight. Good camera control. Can operate in a confined area . Provides better control of the aircra	require a lot of energy . Short flight times. Small payload capacity.	Photography & Videography. 3D scans. Aerial Inspection.
	Fixed-Wing	Long endurance. Large area coverage. Fast flight speed. More forgiving in the air. High altitude, carry more weight .	Launch and recovery needs a lot of space. No VTOL/hover. Harder to fly, more training needed. Expensive.	Agriculture, Inspection, Construction. Aerial Mapping. Pipeline and power line inspection.
	Single-Rotor	VTOL and hover flight Long endurance (with gas power) Heavier payload capability	Aren't as stable or forgiving in a bad landing. Expensive Require a lot of maintenance and care . More dangerous,	Drone surveying Carrying heavy payloads
	Hybrid	VTOL Long-endurance flight	Still in the nascent stage. Not perfect at either hovering or forward fligh	Drone Delivery

vertical take-off and landing (VTOL) aircraft is one that can hover, take off and land vertically without relying on a runway. They can also be distinguished according to their functions: tactical drones, strategic drones and combat drones (Unmanned Combat Air Vehicle UCAV).
The classification of UAVs/drones can be seen differently, such as categorized by the parameters like altitude, endurance, maximum takeoff, payloads, the weight and size of the drones, flight range, endurance, speed, etc.. [28] [29].

Table 1.3: Comprehensive classification of UAVs

	Category	Weight (kg)	Altitude (m)	Endurance (hours)	Data Link Range (km)	Mission
Micro /Mini UAVs	Micro (MAV)	0.1	250	1	<10	Scouting, sampling, surveillance inside building
	Mini	<30	150-300	<2	<10	film and broadcast industry, Farming,pollution measurements, surveillance inside building, communication relay
Tactical UAVs	Close Range(CR)	150	3000	2-4	10-30	mine detection, search and rescue
	Short Range(SR)	200	3000	3-6	30-70	mine detection
	Medium Range(MR)	150-500	3000-5000	6-10	70-200	mine detection, Sampling
	Long Range (LR)	-	5000	6-13	200-500	communication relay
	Endurance	500-1500	5000-8000	12-24	>500	communication relay,Sampling
	Medium Altitude Long Endurance (MALE)	1000-1500	5000-8000	24-48	>500	communication relay,sampling, Weapons delivery
Strategic UAVs	High Altitude, Long Endurance (HALE)	2500-12500	15000-20000	24-48	>2000	communication relay,airport security
Special Task UAVs	Lethal (LET)	250	3000-4000	3-4	300	anti-radar,anti-ship,Anti-aircraft
	Decoys (DEC)	250	50-5000	<4	0-500	Aerial and naval deception
	Stratospheric (Strato)	-	20000-30000	>48	>2000	-
	Exo-stratospheric (EXO)	-	>30000	-	-	-

In such cases, the UAVs are recognized with their parameters, either structural or aerodynamics or other sources. With these, it is significant for the particular application and the author [30] [31] tabulated each UAV with parameters specification considering the example with mission and systems.
The Figure 1.3 presents a comprehensive classification of UAVs that demonstrates both the wide variety of UAV systems and capabilities as well as the multiple dimensions of differentiation.

1.5 Applications

The main goal of the UAVs is to complete a mission that could be military, scientific, economic, or even commercial in nature (Cf. figure 1.8).

Figure 1.8: Applications of UAVs [19]

It is appropriate to enumerate some applications in which we refer to the use of the UAVs [19]:

a) Civil applications :

- - Aerial topography for geographical researches and Meteorological Measurements
- - Agriculture spraying and monitoring
- - Search and rescue and Firefighting and forestry fire detection
- - Pollution Studies and land monitoring
- - Pipelines and Power line inspection
- - Delivery of parcels
- - Urban planning
- - Detection of mobile vehicles on the ground

b) Military applications (Navy, Army and Air Force) :

- - Electronic intelligence
- - Reconnaissance
- - Radar system jamming and destruction
- - Shadowing and Surveillance of enemy fleets
- - Elimination of unexploded bombs

1.6 Conclusion

In this chapter, an introduction to UAVs, their various characteristics, fleet composition and communication structures are reviewed.
There are also many drone software such as controllers and simulators that help the UAV

and its systems to grow and develop in many areas in a safe space with minimal damage and resources.
In the next chapter we will discuss this software and open source simulation and which one you choose to work depends on their characteristics .

Chapter 2

UAVs software components and simulators

2.1 Introduction

Autonomous drones are gaining a lot of interest both in private companies and in universities. This lead to a subdivision of the market in two different categories of systems: open-source and private. The main interest of this thesis will be on open-source systems, with focus on open-source firmware.
Open-source firmware can be divided in 2 different categories based on the automation level they can provide: fully autonomous firmware are those not requiring human intervention at all (or just partially, according to the flight mode that is chosen), whereas non-autonomous firmware are those requiring human control for any operation (no level of automation). In this chapter, we will be focused on the first category.

2.2 UAV software architecture

The simulators allow PX4 and ArduPilot flight code to control a computer modeled vehicle in a simulated "world". You can interact with this vehicle just as you might with a real vehicle, using GCS, an offboard API, or a controller/gamepad (Cf. figure 2.1).
PX4 and ArduPilot supports both Software In the Loop (SITL) simulation, and Hardware In the Loop (HITL) simulation [32]. The SITL simulation allows you to test the behaviour of the code on computer (PX4 flight stack runs on computers). With Hardware-in-the-Loop (HITL) simulation the flight controller firmware is run on real hardware (real flight controller board). The simulators (running on a development computer) are connected to the flight controller hardware via USB/UART.

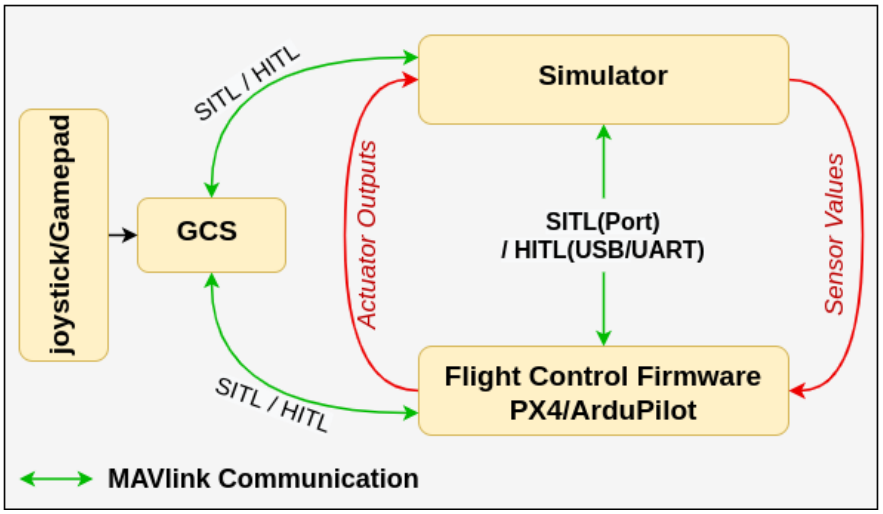

Figure 2.1: UAV firmwares architecture

2.3 Flight control firmware

There are many different open-source firmware available on the market: ArduPilot, PX4, Paparazzi, FlexiPilot, Airrails, SmartAP, Armazila, Autoquad and SLUGS, but only the first two can rely on a stable community and complete documentation [10]. Both ArduPilot and PX4 are valid choices as far as the available functionalities are concerned, wide range of hardware compatibility, good software stability, similar architecture, ROS compatibility and strong community are only a flavour of their entire capabilities, nonetheless, some differences are present. Before choosing one of the two software it is important remarking that both alternatives will lead to similar and optimal performance.

2.3.1 PX4

PX4 [33] is an open source flight control software for drones and other unmanned vehicles. The project provides a flexible set of tools for drone developers to share technologies to create tailored solutions for drone applications [10] .

PX4 provides a standard to deliver drone hardware support and software stack, allowing an ecosystem to build and maintain hardware and software in a scalable way,and allows you to concentrate on your device progress without keeping a complicated development environment.

Figure 2.2: PX4 logo [33]

PX4 is part of Dronecode, a non-profit organization

administered by Linux Foundation to foster the use of
open source software on flying vehicles. Dronecode also hosts QGroundControl, MAVLink the MAVSDK.

Features :

- PX4 is highly modular and extensible both in terms of hardware and software.
- PX4 is Open Source co-developed with a global development community.
- Configurability, PX4 offers APIs and SDKs for developers working with integrations. All the modules are self-contained and can be easily exchanged against a different module without modifying the core. Features are easy to deploy and reconfigure.
- PX4 facilitates the work on localization algorithms, obstacle detection, and autonomous capabilities.
- Validated By the real world to ensure the codebase's safety and reliability.
- PX4 offers an ecosystem of supported devices, advancement of communications, peripherals integration, and power management solutions.
- Great safety features including automated failsafe behaviour.The features are easily configurable and tunable for custom systems.
- All simulators and GCS communicate with PX4 using the MAVLink API.
- The simulators that work with PX4 for HITL and/or SITL simulation is Gazebo ,FlightGear ,JSBSim ,jMAVSim ,AirSim.

2.3.2 ArduPilot

ArduPilot [34] is a trusted, versatile, and open source autopilot system supporting many vehicle types: multi-copters, traditional helicopters, fixed wing aircraft, boats, submarines, rovers and more. The source code is developed by a large community of professionals and enthusiasts. ArduPilot provides a comprehensive suite of tools suitable for almost any vehicle and application. The first ArduPilot open code repository was created in 2009 [10] .

FIGURE 2.3: ArduPilot logo [34]

Features :

- The ArduPilot Project provides an advanced, full-featured and reliable open source autopilot software system.
- As an open source project, it is constantly evolving based on rapid feedback from a large community of users.

- It is capable of controlling almost any vehicle system imaginable.
- Users benefit from a broad ecosystem of sensors, companion computers and communication systems. Since the source code is open, it can be audited to ensure compliance with security and secrecy requirements.
- Installed in over 1,000,000 vehicles world-wide, and with advanced data-logging, analysis and simulation tools.
- It is also used for testing and development by large institutions and corporations such as NASA, Intel and Insitu/Boeing, as well as countless colleges and universities around the world.
- Fully autonomous, semi-autonomous and fully manual flight modes, programmable missions with 3D waypoints, optional geofencing.
- Safefails for loss of radio contact, GPS and breaching a predefined boundary, minimum battery power level.
- Photographic and video gimbal support and integration.
- The ArduPilot firmware element can also be interfaced to other simulators like Gazebo ,Xplane ,JSBSim ,Air-Sim ,RealFlight ,Morse , in SITL (software in the loop) simulation
- The simulators that work with ArduPilot for HITL simulation is X-Plane and FlightGear simulation (only Plane not Copter or Rover) ,and its no longer supported .

2.3.3 PX4 vs ArduPilot

We made a comparison on the key aspects of the two firmwares in Table 2.1, in terms of airframe, flight modes, simulation and licensing [33–38].

Table 2.1: Comparison between PX4 and ArduPilot

Flight control firmware	PX4	ArduPilot
License	BSD (can modify and sell products without making the modifications open)	GPL (people modifying and then selling ArduPilot are obligated to make their modifications open)
Airframes	Planes, multicopters, VTOL airframes and UGVs/rovers	Planes, copters, VTOL airframes, rovers, subs and antenna trackers
Flight modes	13 Modes	23 Modes
Ground control station	QGroundControl	Mission Planner, QGroundControl, APM planner 2, MAVProxy
Simulation and testing	Gazebo, JMAVSim and X-Plane on SITL and HITL. AirSim and JSBSim and FlightGear on SITL only.	JSBSim, Air-Sim, RealFlight and Morse for SITL. X-Plane and FlightGear forSITL and HITL.

2.4 MAVLink

Figure 2.4: MAVLink (Micro Air Vehicle Link)

MAVLink (Micro Air Vehicle Link) [39] is a very lightweight messaging protocol for communicating with drones, and between onboard drone components (Cf. figure 2.4). MAVLink follows a modern hybrid publish-subscribe and point-to-point design pattern.

Key Features

- Very efficient. MAVLink 1 has just 8 bytes overhead per packet, including start sign and packet drop detection. MAVLink 2 has just 14 bytes of overhead (but is a much more secure and extensible protocol). Because MAVLink doesn't require any additional framing it is very well suited for applications with very limited communication bandwidth.
- Very reliable. MAVLink has been used since 2009 to communicate between many different vehicles, ground stations (and other nodes) over varied and challenging

communication channels (high latency/noise). It provides methods for detecting packet drops, corruption, and for packet authentication.

- Many different programming languages can be used, running on numerous microcontrollers/operating systems (including ARM7, ATMega, dsPic, STM32 and Windows, Linux, MacOS, Android and iOS).
- Allows up to 255 concurrent systems on the network (vehicles, ground stations, etc.) both offboard and onboard communications (e.g. between a GCS and drone, and between drone autopilot and MAVLink enabled drone camera).

2.5 Ground Control Station (GCS)

There is a large variety of GCS software application which runs on a ground-based device such as QGroundControl, Mission Planner, MAVProxy.

2.5.1 QGroundControl

QGroundControl [40] is an open source ground control station (GCS) developed and written in C++ using the Qt libraries . This GCS operate on different platforms such as Windows, Mac OS X, Linux, Android and iO. It provides easy and straightforward usage for beginners, while still delivering high end feature support for experienced users [41]. Figure 2.5 shows an example of a drone mission control by QgroundControl.

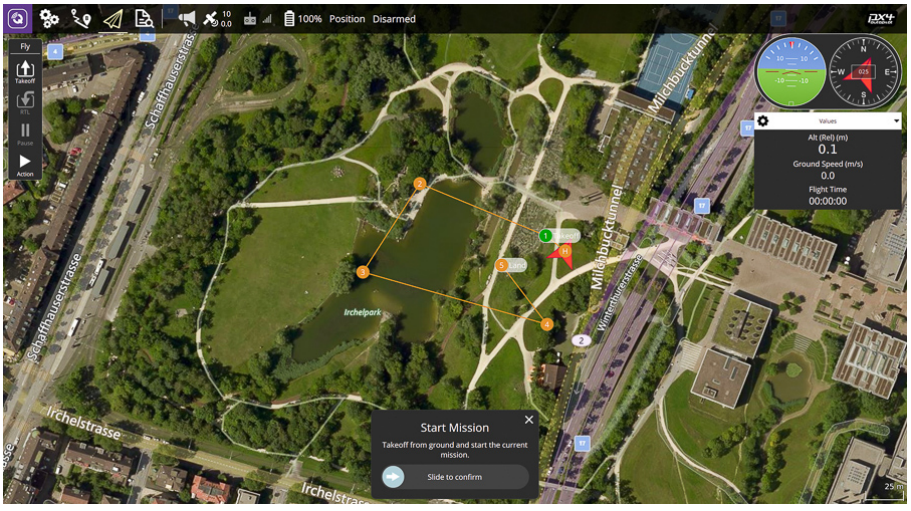

Figure 2.5: Vehicle mission in QGroundControl [40]

- Supports the MAVlink protocol and offers the opportunity to visu- alize details of the MAVLink protocol messages, exchanged between it and the flying vehicle.

- QGroundControl provides full flight control and vehicle setup for PX4 or ArduPilot powered vehicles.
- Mission planning for autonomous flight.
- Flight map display showing vehicle position, flight track, waypoints and vehicle instruments.
- Video streaming with instrument display overlays.
- Support for managing multiple vehicles.

2.5.2 Mission Planner (MP)

This GCS [42] is developed by Michael Oborne using Python Programming Language. Unlike QgroundControl which is compatible with all platforms, MP is compatible with Windows only [41] .Figure 2.6 shows an example of a drone mission control by Mission Planner.

Mission Planner can be used as a configuration utility or as a dynamic control supplement

FIGURE 2.6: Drone mission in mission planner [42]

for your autonomous vehicle. Here are just a few things you can do with Mission Planner:

- Setup, configure, and tune your vehicle for optimum performance.
- Plan, save and load autonomous missions into you autopilot with simple point-and-click way-point entry on Google or other maps.
- Download and analyze mission logs created by your autopilot.
- Interface with a PC flight simulator to create a full hardware-in-the-loop UAV simulator.

- Monitor your vehicle's status while in operation.
- Operate your vehicle in FPV (first person view).

2.5.3 MAVProxy

MAVProxy [43] is a fully-functioning GCS for UAV's written in python , designed as a minimalist, portable and extendable GCS for any autonomous system supporting the MAVLink protocol (such as one using ArduPilot), it can be extended via add-on modules, or complemented with another ground station, such as Mission Planner, QGroundControl etc, to provide a graphical user interface. Figure 2.7 shows an example of a drone control by MAVProxy on Ubuntu.

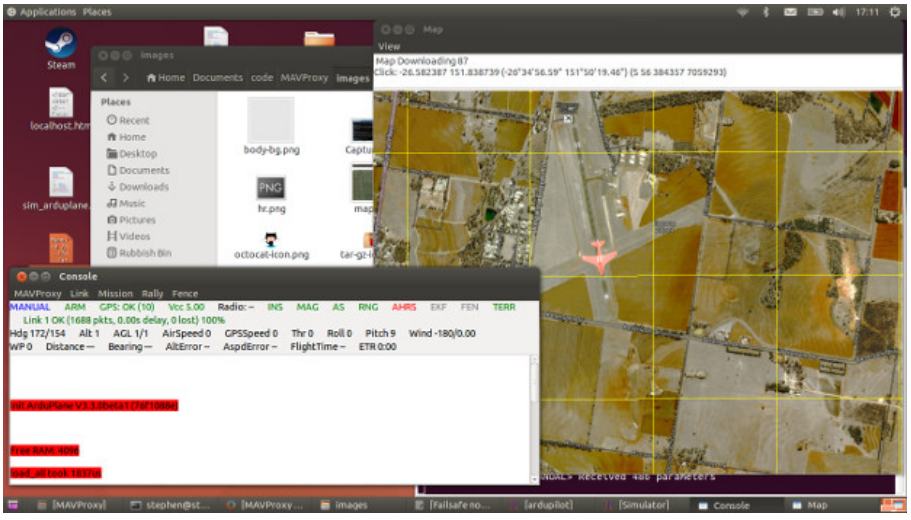

Figure 2.7: MAVProxy running under Ubuntu [43]

Features :

- It has a number of key features, including the ability to forward the messages from your UAV over the network via UDP to multiple other ground station software on other devices.
- MAVProxy is commonly used by developers (especially with SITL) for testing new builds.
- MAVProxy used in to implement attacks on UAV system in order to exploit gaps and vulnerabilities of the MAVLink protocol.
- It is a command-line, console based app. There are plugins included in MAVProxy to provide a basic GUI.
- Can be networked and run over any number of computers.
- The light-weight design means it can run on small netbooks with ease.

2.5.4 Comparison of ground control stations

Following table 2.2 gives a comparison between all of the above mentioned GCS Software.

Table 2.2: Comparison between GCSs

GCS	QGroundControl	Mission Planner	MAVProxy
Interface	Graphical	Graphical	Command
Commercial/Free	Free	Free	Free
licenses	open source	open source	open source
Support MAVLink	Yes	No	Yes
Programming language	Qt, C++	Python	Python
Pilot	PX4 Pro & ArduPilot(APM)	Ardupilot & PX4	Ardupilot
Operating systems	Windows, OS X, Linux, iOS and Android	Windows only	Linux, OS X, Windows, and others

2.6 Unmanned Aerial Vehicle simulators

Simulator is a program or machine that simulates a real-life situation or a physical phenomenon, which means creating a virtual version of it.

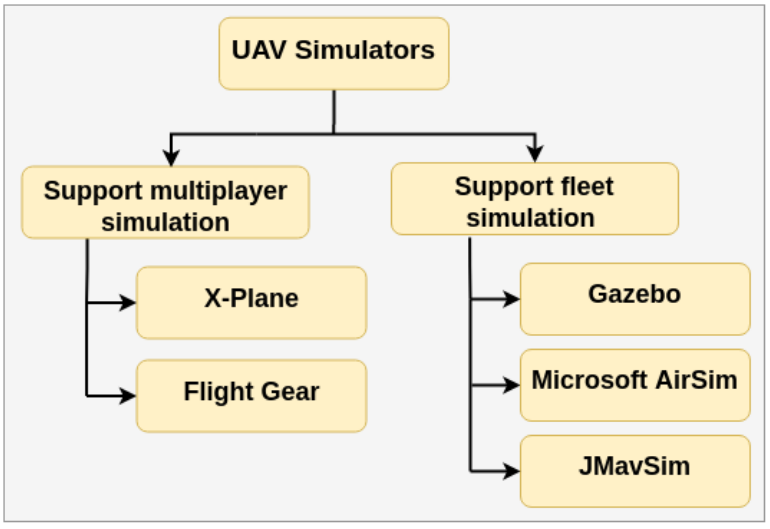

Figure 2.8: Classification of drone simulators

UAV simulators can be divided according to what we want and what we may need in this work, which depends on the type of UAV fleet flight into two types (Cf. figure 2.8):

simulators that can simulate a fleet of UAVs in one device by a single user, and simulators that rely on multiple players or users so that the simulations are based on A number of devices, each user is responsible for one UAV in one device.

2.6.1 Drone fleet simulators

2.6.1.1 Gazebo

Gazebo [44] is a simulator implemented at the University of Southern California. It is used with ROS (Robot Operating System) [45], Gazebo makes it possible to rapidly test algorithms, design robots, perform regression testing, and train AI system using realistic scenarios. Gazebo offers the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments and high-quality graphics, and convenient programmatic and graphical interfaces (Cf. figure 2.9). Gazebo is free with a vibrant community [41].

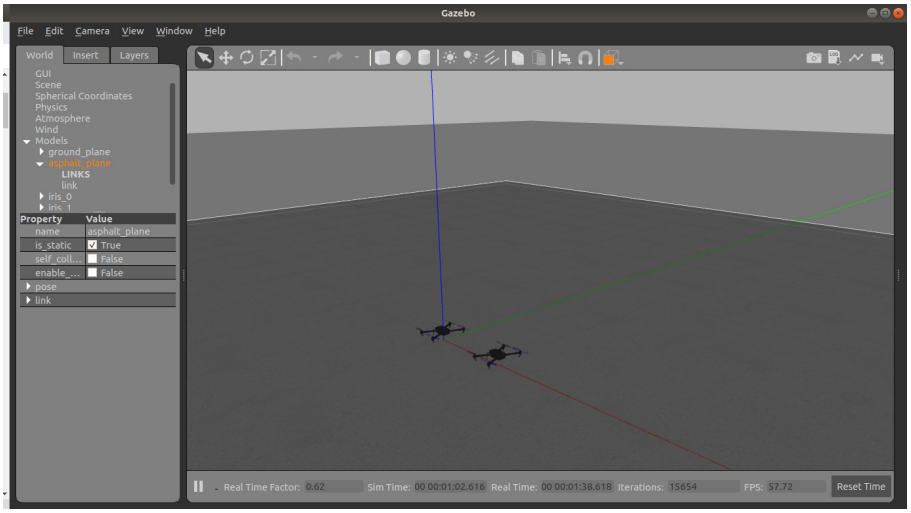

Figure 2.9: Multi UAV Simulation with Gazebo

Features :

- Dynamics simulation :Access multiple high-performance physics engines .
- Advanced 3D graphics with realistic rendering of environments with high-quality lighting, shadows,and textures.
- Sensors and noise models: Generate sensor data, optionally with noise, from laser range finders, 2D/3D cameras...
- Develop custom plugins for robot, sensor, and environment control.
- Graphical User Interface .

- Extensive command line tools for increased simulation introspection and control.

2.6.1.2 *Microsoft AirSim*

AirSim [46] is a simulator developed by Microsoft for drones, cars and more, built on Unreal Engine (AirSim now also have an experimental Unity release). The simulator is revealed in February 2017. It is open-source, cross platform, and supports software-in-the-loop simulation with popular flight controllers such as PX4 ArduPilot and hardware-in-loop with PX4 for physically and visually realistic simulations. There exist APIs for both Python and C++. It is developed as an Unreal plugin that can simply be dropped into any Unreal environment. Similarly, AirSim now have an experimental release for a Unity plugin [41]. Figure 2.10 shows an example of a drone simulation in AirSim simulator .

FIGURE 2.10: Microsoft AirSim simulator

Features :

- Programmatic control and Manual drive using remote control (RC) or arrow keys .
- Suitable for research and project development
- Its easy to generate training data from AirSim for deep learning.
- Control the various options available for weather effects .
- The novel feature of this simulator is the support for protocols such as Micro Air Vehicle Link (MAVLink) which aids in creating more realistic simulations .
- The great visuals can be a downside because it demands powerful GPUs to run.
- Cinematographic Camera
- ROS2 wrapper
- Optical flow camera

- Support for multiple drones in Unity
- Control manual camera speed through the keyboard

2.6.1.3 JMavSim

JMavSim [47] is a simple and lightweight multirotor simulator developed by PX4 , It is easy to set up and can be used to test that your vehicle can take off, fly, land, and responds appropriately to various fail conditions (e.g. GPS failure). Figure 2.11 shows an example of a drone simulation in JMavSim simulator .
JMavSim has a Micro Aerial Vehicle Link(MAVlink) interface . This simulator can be integrated with both ROS and flight controller firmware. It uses the UDP protocol for communication [41].

Figure 2.11: JMavSim simulator [48]

Features :

- Supported only Quad vehicles .
- You can use QGroundControl to fly a mission.
- Change Simulation Speed (increased or decreased with respect to realtime).
- You can connect jMAVSim with a SITL version of PX4 or Start JMAVSim and PX4 Separately.
- JMAVSim can be used for multi-vehicle simulation.
- The simulation can be interfaced to ROS the same way as onboard a real vehicle.

2.6.2 Multiplayer simulators

2.6.2.1 X-Plane

X-Plane [49] is a commercial product of Laminar Research. This simulator is supported by Windows, Linux, Mac OS. It is also available as mobile version for Android and iOS. X-Plane can be used with additional hardware to provide realistic results. It supports many commercial and military aircraft models. The simulator can be extended with help of TCP or UDP sockets to provide multiple instances of different air crafts. The X-Plane uses Plane Maker to imitate UAV flight. It can be used to evaluate physical forces acting on multiple parts of the UAV (Cf. figure 2.12). The native communication protocol used is UDP protocol [41] .

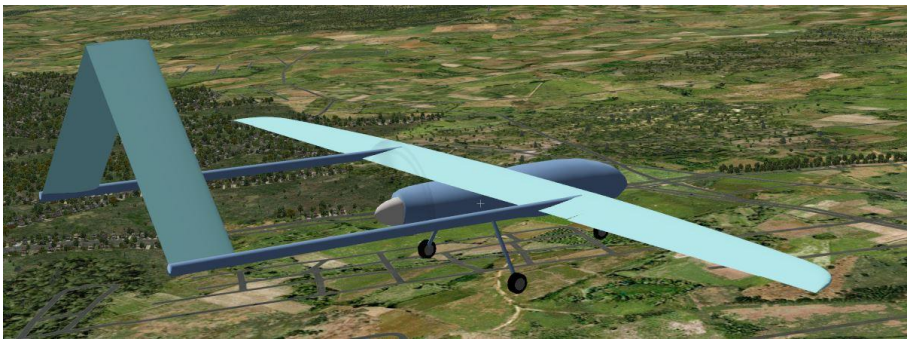

FIGURE 2.12: X-Plane simulator

Features :

- Weather modeling (including downloading current real-world weather from the Internet and low visibility flying conditions ,seasons and Rain/Snow on Runways).
- Simulating system failures (user initiated or completely random).
- Aircraft customization (you can create your own aircraft using the included Plane Maker app).
- Airfoil Maker (to make airfoils for your custom aircraft).
- X-Plane (the actual flight simulator).
- With over 3,000 airports with 3D buildings and Trees.
- Airports populated with static aircraft plus dynamic airport environments.
- Latest road placement and global scenery from Open Street Map.
- X-Plane lighting engine is completely photometric and runs in true HDR at all times. This includes updates to how we do night lighting and artificial light sources. We are finishing up a very careful pass over a wide variety of light sources – urban lights, street lights, and most importantly light sources that affect pilots.

- X-Plane water is 3D. Not only does this make ocean waves more realistic, but this 3D water interacts with the flight model.
- Fleet formation flight for multiplayer.

2.6.2.2 Flight Gear

Flight Gear [50] Flight Simulator is an open source multi platform flight simulator. This simulator can be used on operating systems such as Linux, Windows, Mac OS, and Solaris. Flight Gear source code is released under the GNU General Public License. External 3D modelling software with an XML , to enlist the features of UAVs should be used to model UAV in this simulator. Different multiple UAVs can be initialized at different instances. The multiple mode can coordinate and communicate these multiple UAVs in optimal environment. This feature facilitates simulation of ad hoc network [41].

Flight Gear supports three types of Flight Dynamics Models (FDMs), which contain mathematical equations to calculate the physical forces acting in a simulated UAV. Forces could be drag, thrust, and lift. Flight Gear supports SITL and HITL . Figure 2.13 shows an example of a drone simulation in Flight Gear simulator.

Figure 2.13: Flight Gear simulator [51]

Features [52]

Extensive and Accurate World Scenery Data Base :

- Over 20,000 real world airports included in the full scenery set. Correct runway markings and placement, correct runway and approach lighting.
- Scenery includes all lakes, rivers, roads, railroads, cities, towns, land cover, etc.

- Nice scenery night lighting with ground lighting concentrated in urban areas (based on real maps) and headlights visible on major highways. This allows for realistic night VFR flying with the ability to spot towns and cities and follow roads.
- Scenery tiles are paged (loaded/unloaded) in a separate thread to minimize the frame rate hit when you need to load new areas.

Accurate and Detailed Sky Model : FlightGear implements extremely accurate time of day modeling with correctly placed sun, moon, stars, and planets for the specified time and date. FlightGear can track the current computer clock time in order to correctly place the sun, moon, stars, etc. in their current and proper place relative to the earth.

Flexible and Open Aircraft Modeling System : FlightGear has the ability to model a wide variety of aircraft.

Moderate Hardware Requirements : The intention of FlightGear is to look nice, but not at the expense of other aspects of a realistic simulator.

Networking options A number of networking options allow FlightGear to communicate with other instances of FlightGear, GPS receivers, external flight dynamics modules, external autopilot or control modules. A multi player protocol is available for using FlightGear on a local network in a multi aircraft environment, for example to practice formation flight or for tower simulation purposes.

2.6.3 Comparison of UAVs simulators

Following table 2.3 gives a comparison between all of the above mentioned simulators [41].

Table 2.3: Comparative analysis of UAV simulators

Simulator	Flight Gear	X-Plan	JMavSim	Gazebo	AirSim
Commercial/ Free	Free	Commercial	Free	Free	Free
licenses	open source (GPL)	closed source (proprietary)	open source (BSD 3)	open source (Apache V2)	open source (MIT)
Vehivles	Airplanes, UAVs, Some multirotor	Airplanes, UAVs, Some multirotor	Multirotor	Multirotor and Any robots	Multirotor
Interface ROS	No	No	Yes	Yes	No
Sensors	Diversityof sensors	Easy incorporation of sensors	No incorporation of sensors	Easy moification of sensors	Monocular, depth cameras No lidar
Motion Capture	No	No	No	No	Yes
Obstacles	Yes	Yes	No	Yes	Yes
Ease of Development	Medium	Medium	High	High	Medium
Fleet Flight	Multiplayer	Multiplayer	Fleet	Fleet	Fleet

2.7 Conclusion

In this chapter, A review introduced of different UAVs simulators and flight controller softwares and some other tools and libraries. And it details the requirements, the goals, the strengths and the weakness of each studied tools.
This investigation helps researchers to identify and to select the adequate UAVs performances analysis tools that satisfy their needs.
We, in turn, designed and created a humble multirotor drone simulator, which we will study in the next chapter.

Chapter 3 UAVs Mobility Simulator (UAVMS)

3.1 Introduction

There has been some recent work in drone simulators, there are those who are open source, there are those who support fleet movement, there are those who support artificial intelligence...etc.
We, in turn,we have developed a drone simulator that we called UAVs Mobility Simulator (UAVMS), which focuses on the quadcopter type.

In this chapter, we will introduce our UAVs Mobility Simulator (UAVMS) system design and modeling, the services and libraries used in this project and the user interfaces of the simulator.

3.2 Development environment

Here the necessary services and libraries and all languages used in this project.

Matplotlib.pyplot [53] is a state-based interface to matplotlib. It provides an implicit, MATLAB-like, way of plotting. It also opens figures on your screen, and acts as the figure GUI manager.
pyplot is mainly intended for interactive plots and simple cases of programmatic plot generation.

Numpy [54] is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting,

selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

Pandas [55] is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool,developed in the Python programming language for data processing and analysis. In particular, it presents data structures and manipulations for numerical tables and time series. It is free software released under the BSD license.

PythonRobotics [56] This is a collection of robotics algorithms implemented in the Python programming language. The focus of the project is on autonomous navigation, and the goal is for beginners in robotics to understand the basic ideas behind each algorithm. In this project, the algorithms which are practical and widely used in both academia and industry are selected.

JXMapViewer [57] is an open source Swing component, a special JPanel that presents a slippy map in a java app. Java GUI developers can add it to any Swing application, the way you would with any other JPanel. It loads in raster tiles and presents them.

Java API for JSON The Java API for JSON Processing provides portable APIs to parse, generate, transform, and query JSON using object model and streaming APIs [58] .

3.3 Architecture system of UAVs Mobility Simulator (UAVMS)

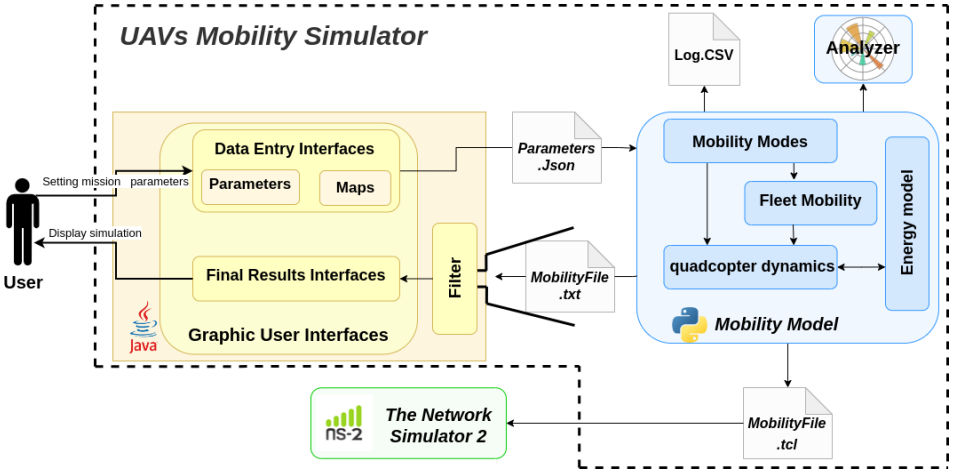

FIGURE 3.1: Architecture system of our simulator (UAVMS)

Our simulator system contains two important sections (Cf. figure 3.1), the first is responsible for user interfaces in addition to maps and displaying the simulated result to the user.

The second section is responsible for navigating a drone or fleet, and displaying mission analysis to the user.

We will explain all the parts of this system in the order of the simulation process, starting from entering parameters to displaying the simulation

3.3.1 UAVs Mobility Simulator navigation model

This is responsible for the navigation model including path tracking and drone fleet flight. It takes the mission parameters from the first section via a JSON file (Cf. figure 3.2).

```
{"mission":{
        "waypoint":[[71,53,3],[231,364,3],[882,187,3]],
        "dt":0.1,
        "Energy":15000,
        "zone":[[71,53,3],[231,364,3],[882,187,3]],
        "sim_end":80,
        "name":"mission-name",
        "nb_quad":10,
        "nb_loop":1,
        "type":"path_fleet"
        ,"Weight":0
}}
```

Figure 3.2: Example of a mission JSON file

3.3.1.1 Quadcopter Dynamics

The quad-rotor architecture has been chosen for this research for its low dimension, good maneuverability, simple mechanics and payload capability. However, thanks to its structure, it is quite easy to choose the four controllable variables and to decouple them to make the control task easier. The four quad-rotor targets are thus related to the four basic movements which allow the helicopter to reach a certain height and attitude.

Throttle- This is the control of the power being fed to the drone. It is that power that makes the drone move slowly or faster. For the power to be adequate, all the propellers must be moving at the same speed (Cf. figure 3.3).

Yaw- This is the clockwise or anticlockwise rotation of the drone that allows you to command the drone into making patterns or circles in the air. If the drone rotates in the right direction, propellers 1 and 4 must move at high speed while 2 and 3 moves at low speed (Cf. figure 3.3).

Pitch- Refers to forward and backward tilting of the drone. For the forward pitch movement to occur, propellers 2 and 1 must move at average speed while 1 and 4 moves at high speed. On the other hand, if you want the drone to move backward,

propellers 1 and 4 must be at a low rate while 2 and 3 be at average speed (Cf. figure 3.3).

Roll– Right or left movement of the drone in the air. For this to occur, propellers 1 and 3 must be moving at average speed while 2 and 4 are at high speed. On the other hand, if you want your drone to roll to the right side, propellers 1 and 3 must be running at high speed while 2 and 4 run at a low speed (Cf. figure 3.3).

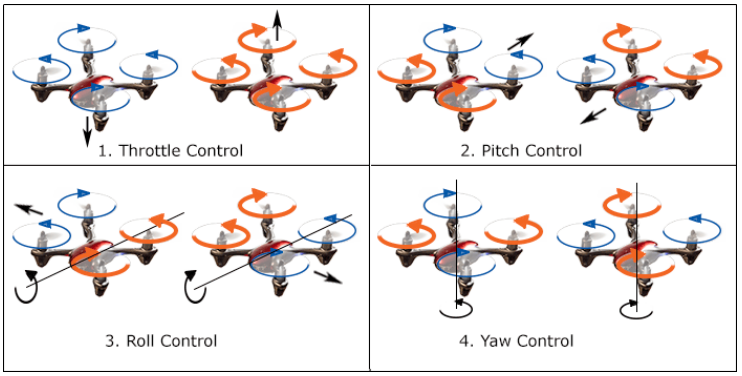

FIGURE 3.3: Quadcopter Control Motor Speeds

Besides the four basic movements of the quadcopter, our simulator navigation model also contains the power model.

3.3.1.2 Energy model

The energy consumed is affected by the communication characteristics between the drone and other wireless devices (frequency, bandwidth...), navigation, payload, sensors and other variables according to the type of drone and the type of mission [59].
In our simulation we implement energy model that depend on thrust, total weight (weights of the vehicle and payload and battery, drag force and flight speed [60].
Other variables, such as communications and sensors, are not currently applied in our simulator. Adding them depends on upcoming releases and updates.

3.3.1.3 Fleet control strategy in the UAVMS

In our simulator, we have three ways of positioning , and controlling a group of drones.

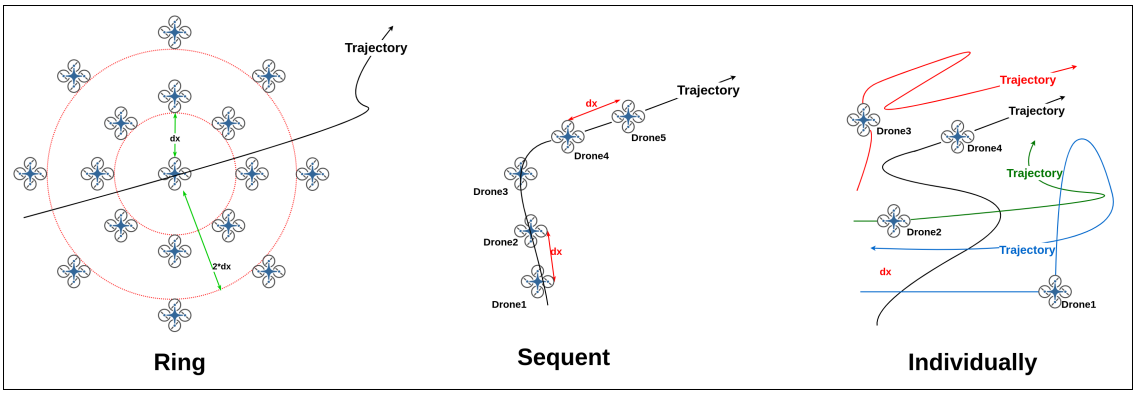

FIGURE 3.4: Fleet Control Strategy On Mobility simulator

Drones placed in a ring In this strategy, one drone is identified that is in the center and surrounded by the rest of the drone in the form of a ring. Each ring is separated from the other by a value of x, which is determined by the length, width and height of the drones.All drones follow the path set by the central drone (Cf. figure 3.4).

Drones in a sequential line In this strategy, all the drones are placed in a sequential row, with each drone being at a distance x from the other. x is determined by the length, width, and height of the drone. All drones in this strategy follow a single path (Cf. figure 3.4).

Each drone is individually In this strategy, each aircraft has its own path and speed and is not related to the rest of the drones (Cf. figure 3.4).

3.3.1.4 Mobility modes

In our simulator we have implemented two main mobility modes (Cf. figure 3.5).

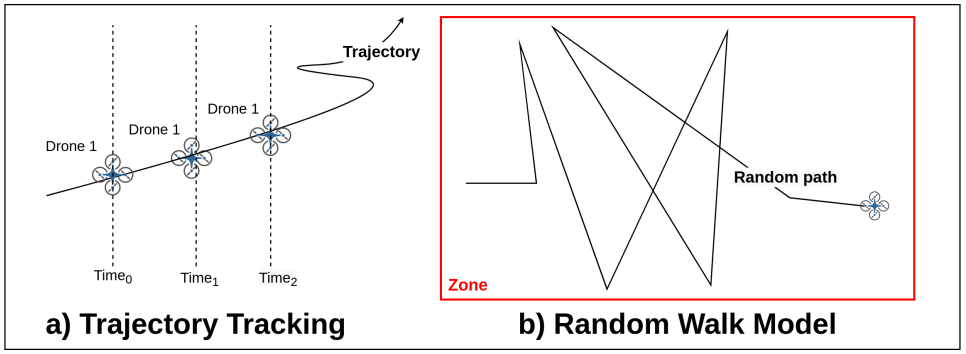

FIGURE 3.5: Mobility modes of UAVMS

Trajectory tracking In this flight mode, all the drones follow the specified path.(Cf. figure 3.5) The number of circles that the drone must make in the path, In addition

to the points that define the path. and the number and settings of the drone, are given by the user.

In this mode, the user can choose the way to position a group of drones across the path,either in the form of a circle (Ring) or a line .

Random Walk Model In this flight mode, each drone follow a random walk model in the selected area , so that a random point is chosen for a new path each time the drone ends the previous path (Cf. figure 3.5) . In this mode, the user can choose the way to position a group of drones across the path,either in the form of a circle (Ring) or each drone is individually .

3.3.1.5 Navigation files

Our simulator outputs the navigation files from the UAVMS navigation model :

- log.txt that it sends to the first partition to complete the simulation. This file contains the time, drone ID, x position, y position, z position, and energy in the exact order of all drones (Cf. figure 3.6).

```
212 13.999999999999966;1;72.63823895649085,77.40217096079022,4.872819456725468;99.96554820252908
213 14.099999999999966;0;158.1058917370159,84.2055374703641,4.934976800017281;99.90902698556575
214 14.099999999999966;1;73.35235192284489,77.45898679377575,4.87155118556858;99.96507524263421
215 14.199999999999966;0;159.7986304647912,84.34026858697202,4.9369278629008795;99.90791367466862
216 14.199999999999966;1;74.08262653150751,77.51709034651805,4.870372829557534;99.96459152745312
217 14.299999999999965;0;161.50031818324905,84.4757105303724,4.938907305153691;99.90679466716344
218 14.299999999999965;1;74.82958426221394,77.5765235650109,4.8691999336018;99.96409672661338
219 14.399999999999965;0;163.2102967851516,84.61181089624327,4.940950212840351;99.905670434111
220 14.399999999999965;1;75.59365938919812,77.63732140872384,4.867968983007811;99.96359054749959
221 14.499999999999964;0;164.92805666916612,84.74852912638254,4.943071439187735;99.90454135108827
```

FIGURE 3.6: Example of a log.txt file

- log.csv that can be used for analytics and graphical data generation. This file contains the time, drone ID, x position, y position, z position, roll, pitch, yaw, velocity and energy in the exact order of all drones (Cf. figure 3.7).

time	id	name	x	y	z	roll	pitch	yaw	velocity	energy
0.1	0	drone0	55	76	0	[0.]	[0.]	0	[0.0, 0.0, 0.0	100
0.2	0	drone0	55	76	1.231696	[0.]	[0.]	0	[0.0, 0.0, 12	99.9898860494371
0.3	0	drone0	55	76	3.0925774	[0.]	[0.]	0	[0.0, 0.0, 18	99.981528944206
0.4	0	drone0	55	76	4.367373	[0.]	[0.]	0	[0.0, 0.0, 12	99.9776876277018
0.5	0	drone0	55	76	4.4429192	[0.]	[0.]	0	[0.0, 0.0, 0.	99.9771739492759
0.6	0	drone0	55	76	4.5164816	[0.]	[0.]	0	[0.0, 0.0, 0.	99.9771302891661
0.7	0	drone0	55	76	5	[-0.0006045]	[0.0076209	0	[0.0, 0.0, 0.0	99.9771302891661
0.8	0	drone0	55.0001182	76	4.9999999	[-0.0006045]	[0.0076209	0	[0.00118240	99.9771302119022
0.9	0	drone0	55.0006779	76.0001	4.9999989	[-0.0006045]	[0.0076209	0	[0.00559633	99.9771298466477
1	0	drone0	55.0022404	76.0002	4.9999928	[-0.0006045]	[0.0076209	0	[0.01562523	99.9771288287265
1.1	0	drone0	55.0055775	76.0004	4.9999714	[-0.0006045]	[0.0076209	0	[0.03337083	99.9771266598741
1.2	0	drone0	55.0115882	76.0009	4.9999159	[-0.0006045]	[0.0076209	0	[0.06010764	99.9771227634472
1.3	0	drone0	55.0211827	76.0017	4.9998014	[-0.0006045]	[0.0076209	0	[0.09594492	99.9771165593925

FIGURE 3.7: Example of a log.csv file

- scenario.tcl that can be used as a scenario in the ns-2 simulator. This file contains the time, drone ID, x position, y position and velocity in the exact order of all drones (Cf. figure 3.8).

```
# Number of vehicles: 2
# Path : [[55,76],[408,104],[730,81],[55,76],]
$node(0) set X_ 55.0
$node(0) set Y_ 55.0
$node(0) set Z_ 55.0
$node(1) set X_ 55.0
$node(1) set Y_ 55.0
$node(1) set Z_ 55.0
$ns_ at 0.1 "$node(0) setdest 55.0 76.0 0.0"
$ns_ at 6.8999999999999915 "$node(1) setdest 55.0 76.0 0.0"
$ns_ at 6.999999999999991 "$node(0) setdest 71.25670016563872 77.29225798526475 7.2101460620358395"
```

FIGURE 3.8: Example of a scenario.tcl file

The UAVs Mobility Simulator also shows a simple analysis of the simulation for the user using matplotlib [53].

3.3.2 Data entry interfaces

The user first gives all the parameters and options for the simulation through the user interfaces, the simulator in turn writes all the options and parameters to the JSON file, and it also executes the Python program responsible for navigating with the JSON file as input.
Below we will introduce the UAVM simulator through the users' view, and how to configure the simulation through the user interfaces .

The first interface (Cf. figure 3.9) that is displayed after running the program is the main simulation interface.

Through this interface, you can :

- Create a new mission via (New).
- Restart the simulation by clicking on (Restart).
- (Reset) to cancel the mission.
- (Start) to start the simulation process.
- (Pause) to pause the simulation process.
- (Resume) to resume the simulation.
- The buttons (Start,Stop,Resume) can only be clicked after creating a mission.

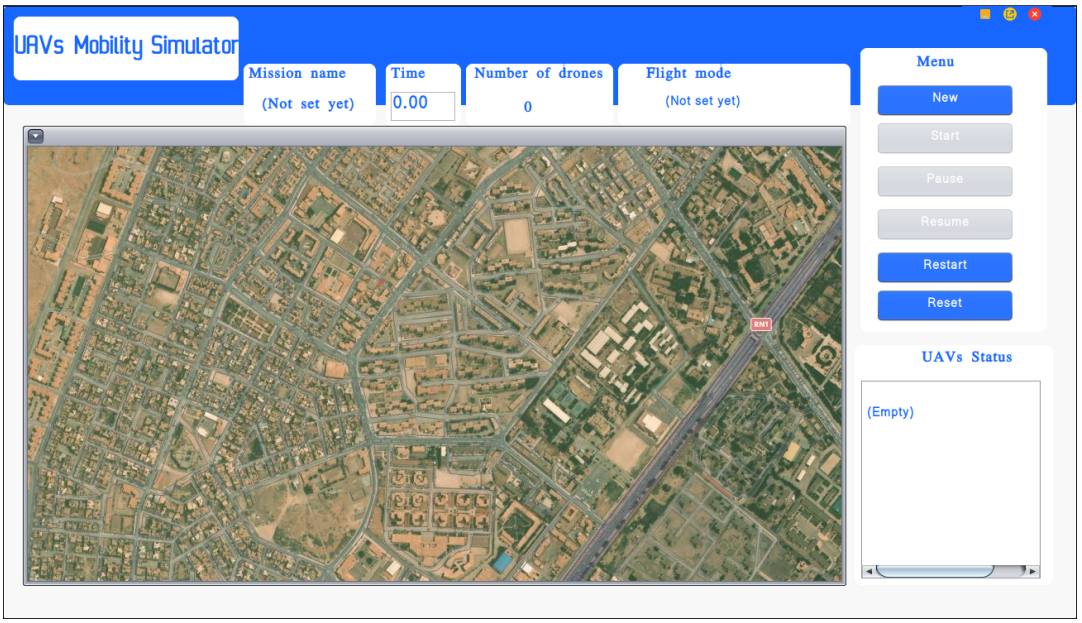

Figure 3.9: The main simulation interface

The next interface (Cf. figure 3.10) appears by pressing (New) to create a new mission. Through this interface, you can create a name for the mission and specify the type of mission.

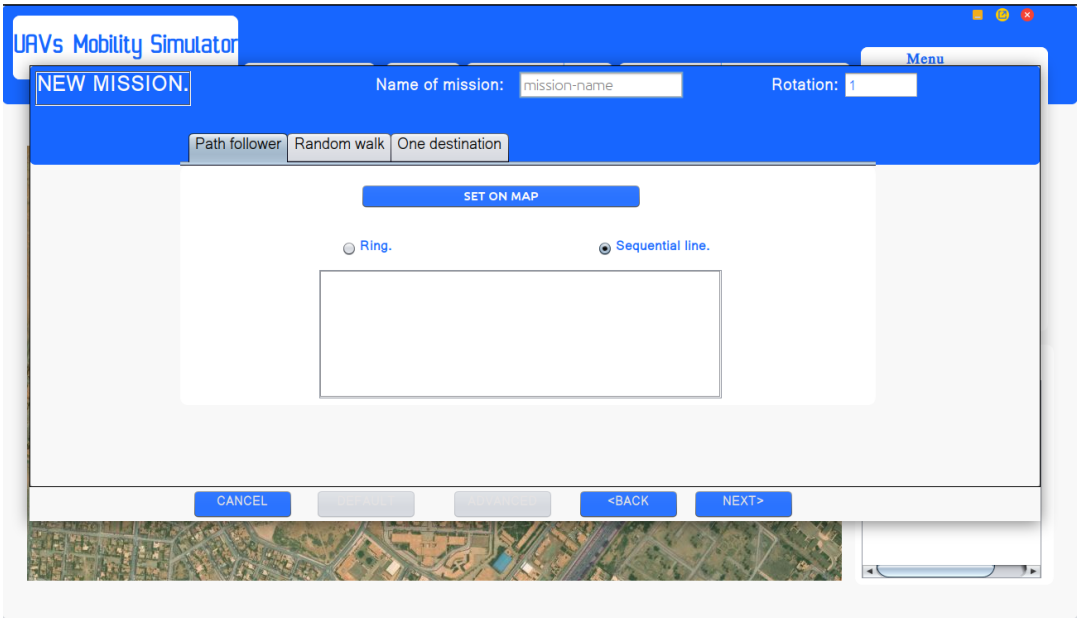

Figure 3.10: The mission creation interface

- To determine the trajectory tracking, you can specify how the drones will travel, one by one in the form of a line , or as a form of a circle . You can also create the route manually or through the map by pressing (Set on map) button (Cf. figure 3.11).

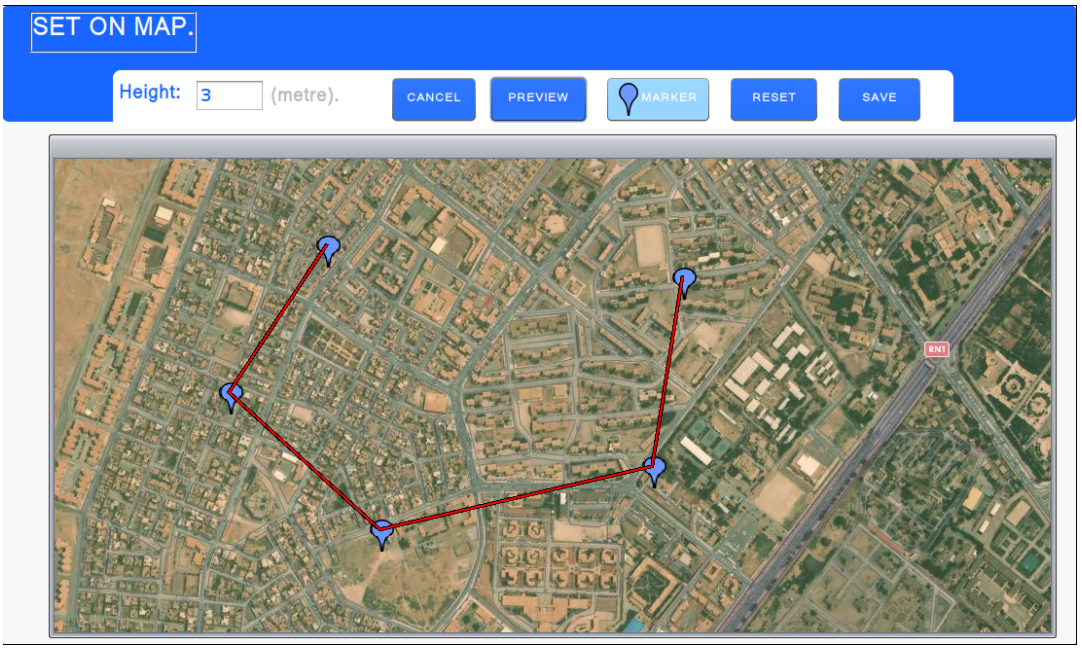

Figure 3.11: Determine the path on the map

To select the path points, press Marker, then Preview to display the path, then Save to save the path.

- For the random walk model, you can specify how the drones will move, individually, or as a group (fleet). You can also specify the area manually or through the map by pressing (Set on map) button (Cf. figure 3.12).

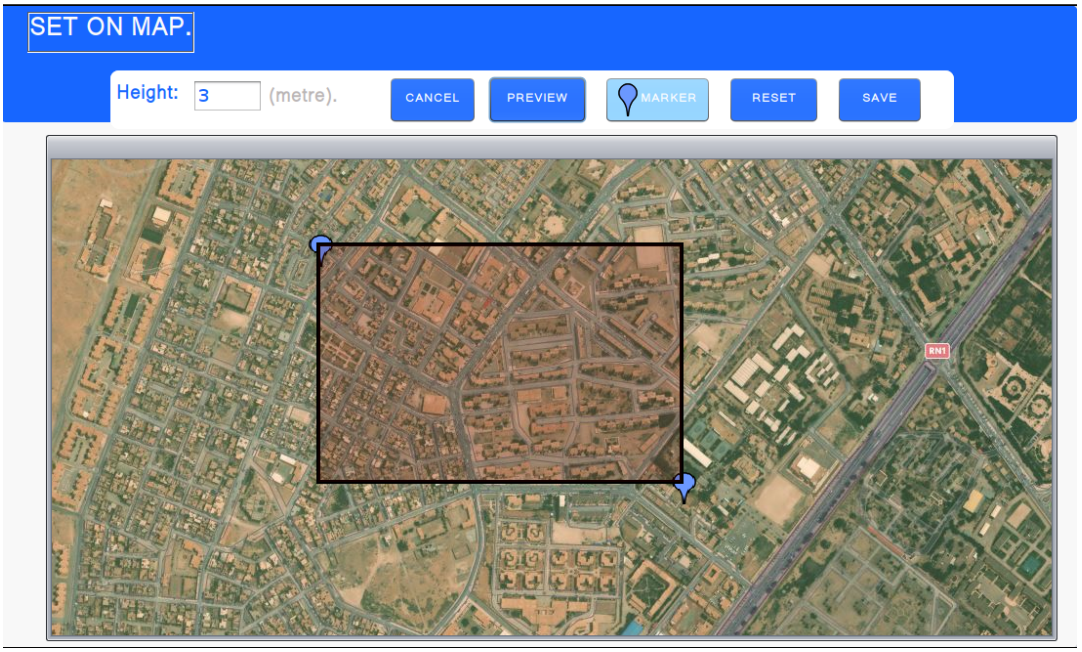

Figure 3.12: Determine the zone on the map

- For the point-to-point navigation model, you can specify how the drones will move, individually, or as a group (fleet). You can also specify the two points manually or through the map by pressing (Set on map) button (Cf. figure 3.13).

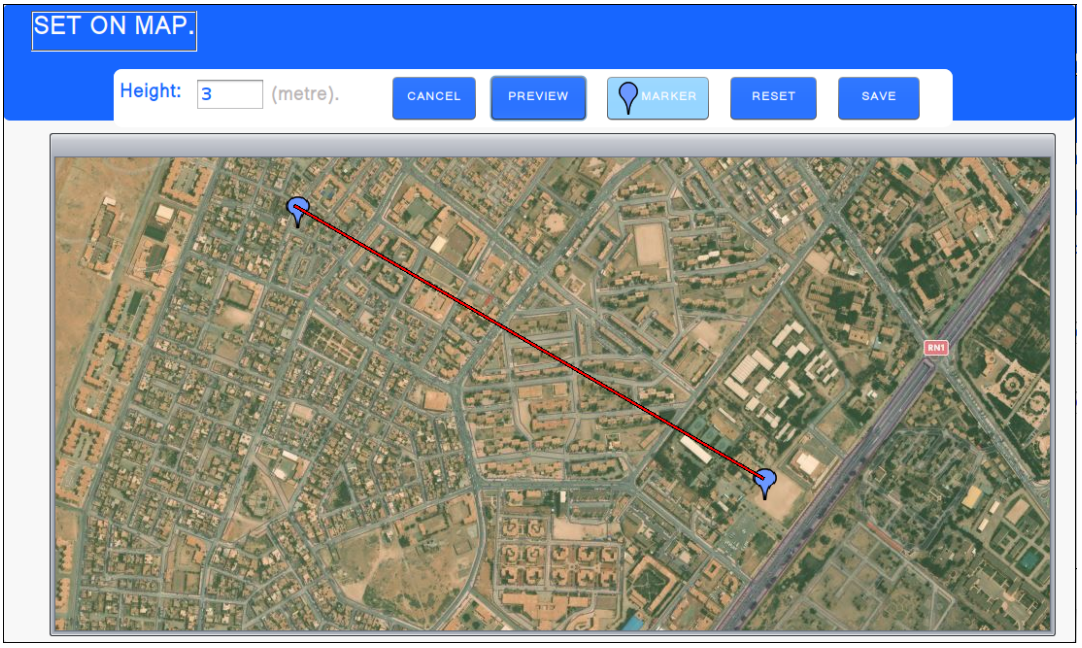

FIGURE 3.13: Determine the start and end points on the map

The last interface (Cf. figure 3.14) is responsible for the additional settings for the drone, simulator, and battery .

FIGURE 3.14: Define the parameters of the drone

- Number of drones.
- Speed.
- Acceleration.

- Weight.
- Capacity of battery.
- Analyse (Enabled or Disabled).

3.3.3 UAVMS analyzer

Our simulator can provide an analysis of battery consumption and remaining power , as well as velocity , as well lateral and vertical position .An example of this is in Figure 3.15 view the analysis of the fleet scenario n the form of a circle (random walk model) .

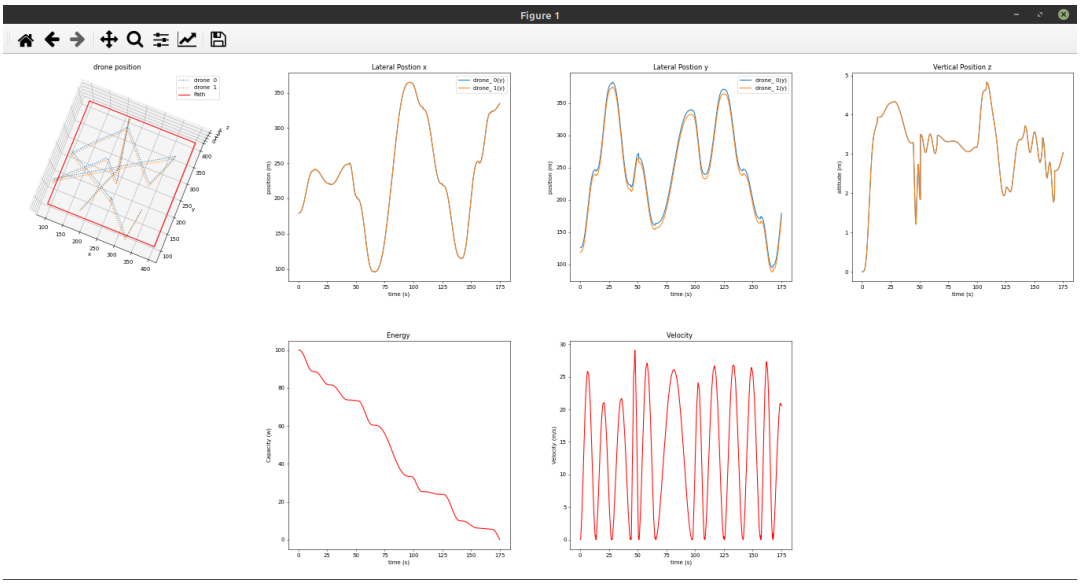

FIGURE 3.15: Random Walk Model (Fleet) analyse

3.3.4 Filter

The filter reads the navigation file, which contains the location and speed, in addition to the percentage of remaining battery power for each drone, all in terms of time t, and then filters the navigation file according to the drone so that it gives each drone its coordinates, characteristics and information (Cf. figure 3.16).

3.3.5 Final Results Interfaces

This section is responsible for tracking and displaying the locations of the drones during the simulation in the map, in addition to displaying the percentage of remaining battery

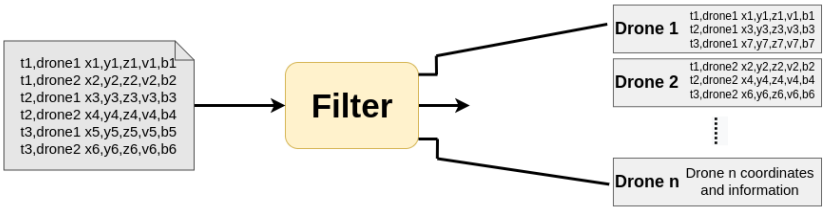

FIGURE 3.16: File Navigation Filter

power by reading the navigation file generated from the navigation section after passing the filter.

In this chapter we present six possible scenarios for our simulator.

A. Scenario of one drone (trajectory tracking) In this scenario, we have set the trajectory tracking mode with one drone (Cf. figure 3.17).

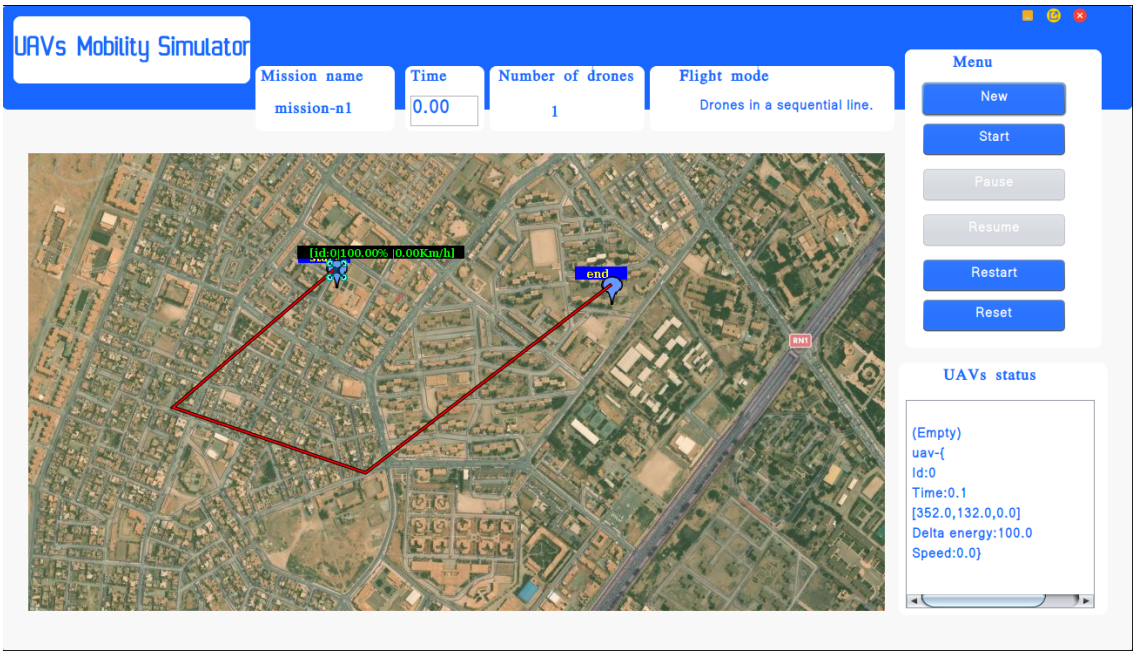

FIGURE 3.17: Scenario of one drone (trajectory tracking)

B. Scenario of fleet in form of line (trajectory tracking) In this scenario, we have set the trajectory tracking mode with the strategy of moving the fleet in sequential line. Figure 3.18 shows the final outcome of this scenario in the map.

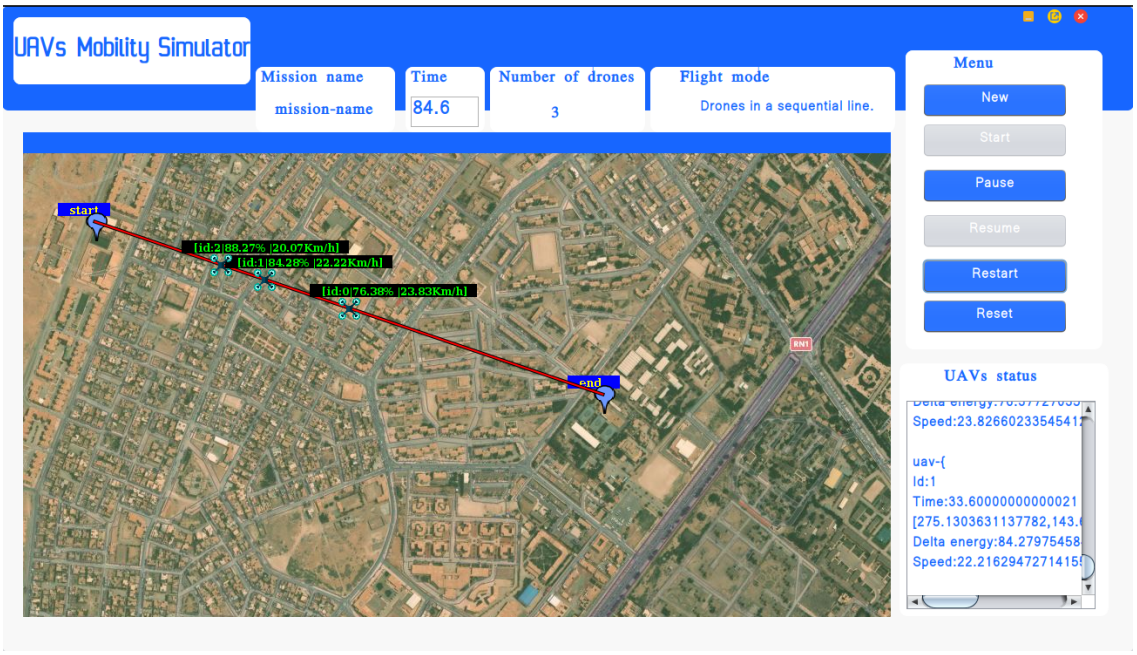

FIGURE 3.18: Scenario of fleet in form of line (trajectory tracking)

C. Scenario of fleet in form of circle (trajectory tracking) In this scenario, we have set the trajectory tracking mode with the strategy of moving the fleet as a circle. Figure 3.19 shows the final outcome of this scenario in the map.

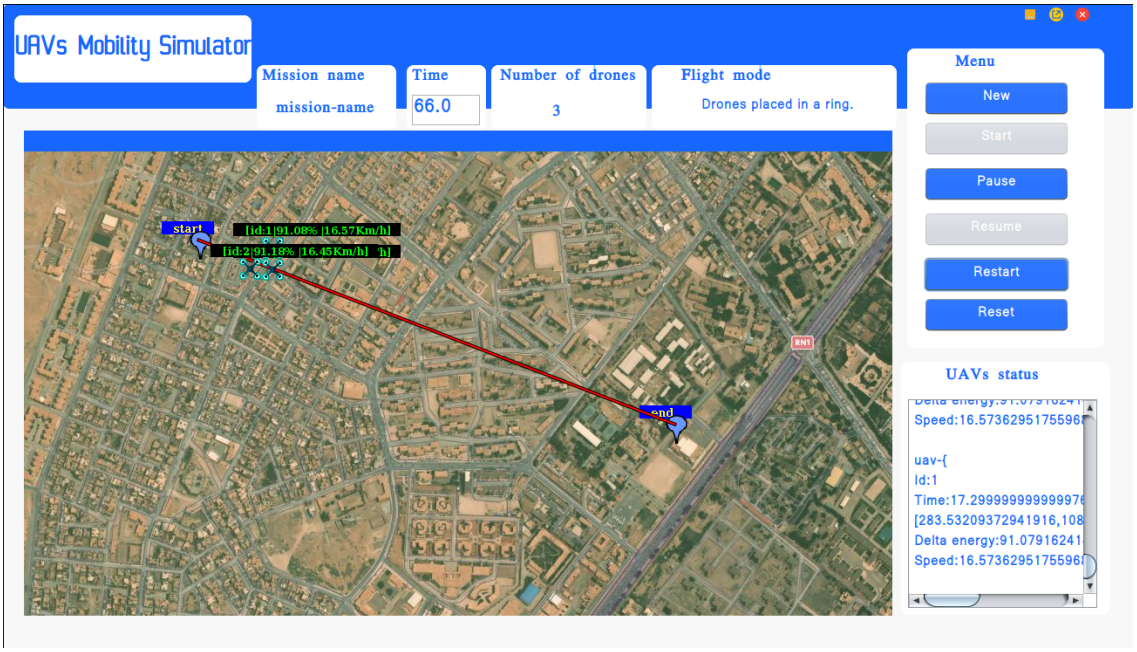

FIGURE 3.19: Scenario of fleet in form of circle (trajectory tracking)

D. Scenario of one drone (random walk model) In this scenario, we have set the random walk mode with one drone (Cf. figure 3.20).

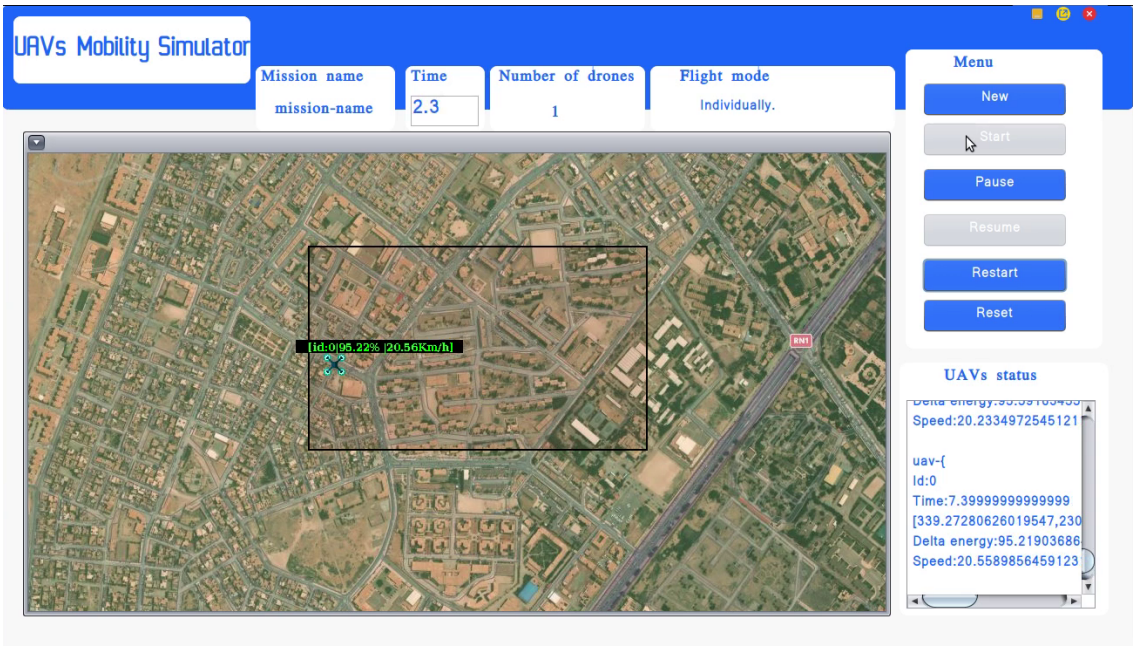

FIGURE 3.20: Scenario of one drone (random walk model)

E. Scenario of fleet in form of circle (random walk model) In this scenario, we have set the random walk mode with the strategy of moving the fleet as a circle. Figure 3.21 shows the final outcome of this scenario in the map.

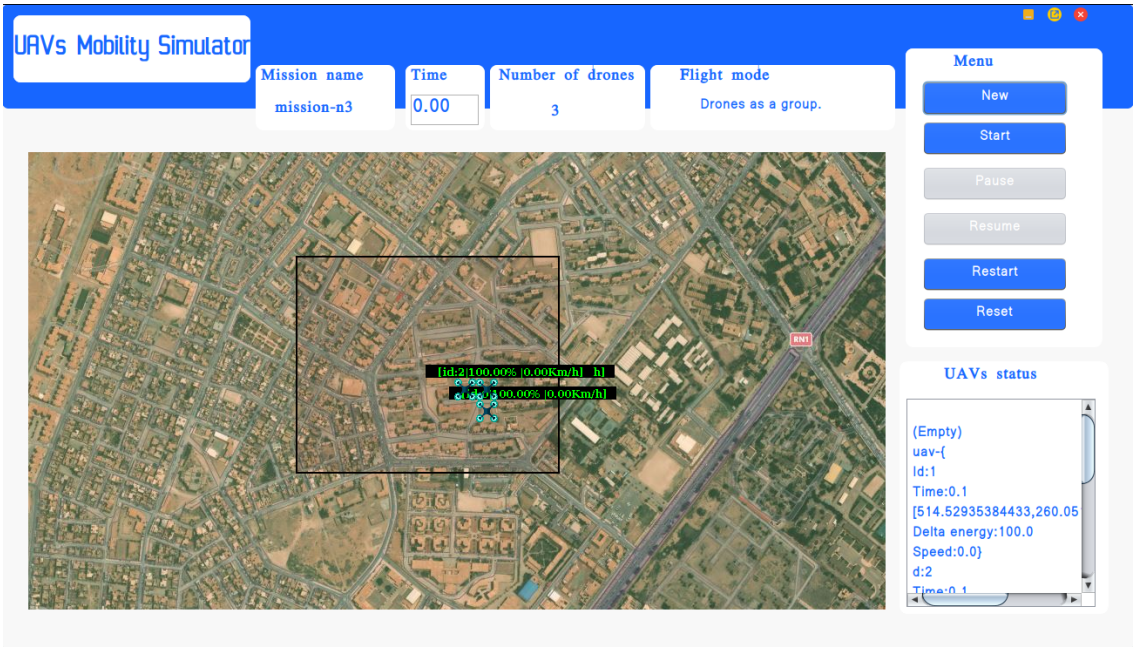

FIGURE 3.21: Scenario of fleet in form of circle (random walk model)

F. Scenario of each drone individually (random walk model) In this scenario, we have set the random walk mode with the strategy of moving each drone is individually. Figure 3.22 shows the final outcome of this scenario in the map.

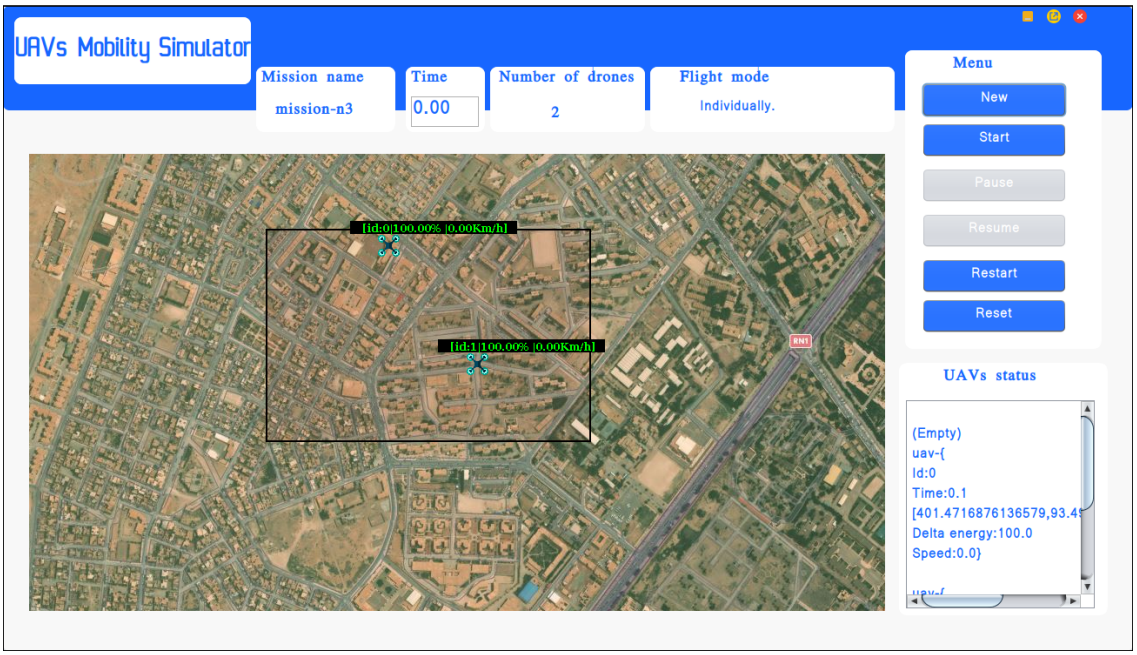

FIGURE 3.22: Scenario of each drone individually (random walk model)

3.3.6 Network Simulator (NS2) support

As mentioned earlier, our simulator creates a nav file for ns2 in tcl language in any of the previously mentioned scenarios. Example of this is in figure 3.23 display the result of the fleet scenario in the form of a line (Trajectory tracking) in the Network Simulator 2.

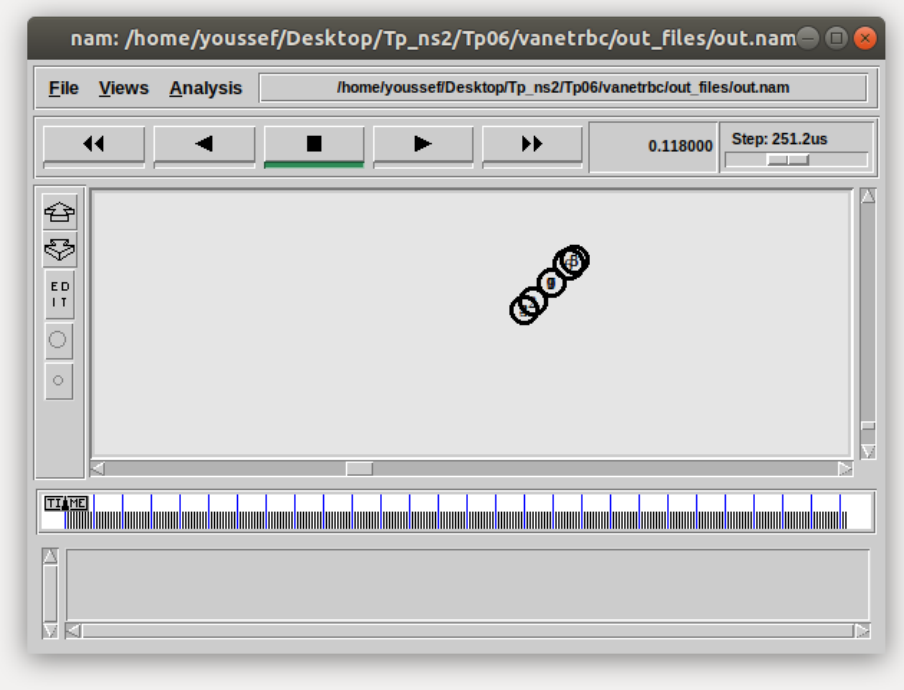

FIGURE 3.23: Running The Navigation File In NS2

3.4 Conclusion

In this chapter, we have designed and modeled the system and overall structure of our navigation simulator (UAVMS), in addition to defining the environment needed for development and mentioning the tools and libraries used, and graphical user interfaces. Finally, we provided all the scenarios in UAVs Mobility Simulator.

General Conclusion

Drones have been a subject of coordinated research over the past few years, due to their autonomy, flexibility, and wide range of application areas. The most important contributor to facilitating the research and development of drones is simulation programs. During this final graduation project, we created a mobility simulator for drones, which we called the UAVs Mobility Simulator (UAVMS).
The goal of this simulator is to simulate the movement of a single drone or fleet on a specific path or in a specific area.

Our simulator is for quadcopter type drones. It offers features such as ease of use, variety of scenarios, battery model, which calculates the percentage of remaining power, in addition to some dynamics of the drone such as roll, pitch and yaw angles, torque, thrust and drag...etc.
Besides the trajectory tracking and random walking modes, our simulator offers strategies for drones to navigate as a fleet and that is to navigate in a sequence, as a circle, or individually.
Our simulator is free and open source, so researchers and developers can add their contributions to our simulator such as communication model, more mobility models, climatic phenomena...etc.

Bibliography

[1] J.-N. Lehec, "Dji smart controller, la radiocommande avec écran intégré pour contrôler son drone," 2019. url = https://phototrend.fr/2019/01/dji-smart-controller-telecommande-ecran-integre-drone/, Accessed: 06.06.2022. 4

[2] "Px4 development guide, sensors," 12.3.2021. url = https://docs.px4.io/v1.12/en/getting$_s$*tarted/sensor*$_s$*election.html*, *Accessed* : 06.06.2022. 4, 5

[3] "Px4 user guide, gps compass," 6.13.2021. url = https://docs.px4.io/master/en/gps$_c$*ompass/*, *Accessed* : 06.06.2022. 4

[4] "Px4 user guide, distance sensors (rangefinders)," 6.13.2021. url = https://docs.px4.io/master/en/sensor/rangefinders.html, Accessed: 06.06.2022. 5

[5] S. S. Marcello Chiaberge, Gianluca Dara, "Px4 autopilot customization for non-standard gimbal and uwb peripherals." 5, 6, 7, 8

[6] "Px4 development guide ,payloads and cameras," 6.3.2021. url = https://docs.px4.io/v1.12/en/payloads/ , Accessed: 06.06.2022. 5

[7] J. Wu, J. Ma, Y. Rou, L. Zhao, and R. Ahmad, "An energy-aware transmission target selection mechanism for uav networking," *IEEE Access*, vol. 7, pp. 67367–67379, 2019. 6

[8] M. Tropea, P. Fazio, F. De Rango, and N. Cordeschi, "A new fanet simulator for managing drone networks and providing dynamic connectivity," *Electronics*, vol. 9, p. 543, 03 2020. 6

[9] "Px4 development guide, hardware(escs motors)," 12.3.2021. url = https://docs.px4.io/v1.12/en/peripherals/esc$_m$*otors.html*, *Accessed* : 06.06.2022. 6

[10] E. S. M. Ebeid, M. Skriver, K. Laursen, K. Jensen, and U. Schultz, "A survey of open-source uav flight controllers and flight simulators," *Microprocessors and Microsystems*, vol. 61, 05 2018. 6, 19, 20

[11] "Px4 development guide, px4 architectural overview," 6.13.2021. url = https://docs.px4.io/v1.12/en/concept/architecture.html , Accessed: 06.06.2022. 7, 8

[12] "Px4 development guide ,modules and commands (estimators)," 11.18.2020. Accessed: 06.06.2022. 8

[13] Z. Cui, K. Guan, C. Briso-Rodríguez, B. Ai, Z. Zhong, and C. Oestges, "Channel modeling for uav communications: State of the art, case studies, and future directions," 2021. 9

[14] B. Crumley, "Blockchain tech may thwart hacks of automated drone fleets," 2021. url = https://dronedj.com/2021/10/07/blockchain-tech-may-thwart-hacks-of-automated-drone-fleets/, Accessed: 06.06.2022. 9

[15] J. A. Guerrero and R. Lozano, *Flight Formation Control*. John Wiley & Sons, 2012. 11

[16] C. Kownacki, "Multi-uav flight using virtual structure combined with behavioral approach," *Acta Mechanica et Automatica*, vol. 10, pp. 92–99, 06 2016. 11

[17] C. Rice, Y. Gu, H. Chao, T. Larrabee, S. Gururajan, M. Napolitano, T. Mandal, and M. Rhudy, "Autonomous close formation flight control with fixed wing and quadrotor test beds," *International Journal of Aerospace Engineering*, vol. 2016, pp. 1–15, 07 2016. 11

[18] Q. Zhang and H. H. T. Liu, "Robust nonlinear control of close formation flight," 2019.

[19] R. Abdallah, *Reliability approaches in networked systems : Application on Unmanned Aerial Vehicles*. Theses, Université Bourgogne Franche-Comté, May 2019. 11, 12, 13, 16

[20] Q. Zhang and H. H. T. Liu, "Robust cooperative formation control of fixed-wing unmanned aerial vehicles," 2019. 11

[21] G. Antonelli, S. Chiaverini, and A. Marino, "Decentralized deployment with obstacle avoidance for auvs*," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 12807–12812, 2011. 18th IFAC World Congress. 11

[22] E. Ferrera, A. Alcántara, J. Capitán, A. R. Castaño, P. J. Marrón, and A. Ollero, "Decentralized 3d collision avoidance for multiple uavs in outdoor environments," *Sensors*, vol. 18, no. 12, 2018. 11

[23] J. Li, Y. Zhou, and L. Lamont, "Communication architectures and protocols for networking unmanned aerial vehicles," pp. 1415–1420, 12 2013. 11

[24] L. Zhou, H. Ma, Z. Yang, S. Zhou, and W. Zhang, "Unmanned aerial vehicle communications: Path-loss modeling and evaluation," *IEEE Vehicular Technology Magazine*, vol. 15, no. 2, pp. 121–128, 2020. 12

[25] L. Chiaraviglio, L. Amorosi, N. Blefari-Melazzi, P. Dell'Olmo, A. L. Mastro, C. Natalino, and P. M. 0001, "Minimum cost design of cellular networks in rural areas with uavs, optical rings, solar panels, and batteries," *TGCN*, vol. 3, no. 4, pp. 901–918, 2019. 12

[26] J.-A. Maxa, M. S. B. Mahmoud, and N. Larrieu, "Secure routing protocol design for uav ad hoc networks," *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, pp. 4A5–1–4A5–15, 2015. 13

[27] P. Gabrlik, "The use of direct georeferencing in aerial photogrammetry with micro uav," *IFAC-PapersOnLine*, vol. 48, no. 4, pp. 380–385, 2015. 13th IFAC and IEEE Conference on Programmable Devices and Embedded Systems. 14

[28] R. D. Ballesteros Ruiz, A. C. Lordsleem Júnior, and J. H. Aquino Rocha, "Inspection of facades with unmanned aerial vehicles (uav): an exploratory study," *Revista ALCONPAT*,

vol. 11, pp. 88 – 104, Jan. 2021. 14

[29] A. Aabid, B. Parveez, N. Parveen, S. Khan, M. A. Raheman, M. Zayan, and O. Shabbir, "Reviews on design and development of unmanned aerial vehicle (drone) for different applications," *Journal of Mechanical Engineering Research and Developments*, vol. 45, pp. 53–69, 04 2022. 14

[30] M. D. F. Bento, "Unmanned aerial vehicle : An overiew," *insidegnss*, pp. 54–61, 2008. 15

[31] G. Singhal, B. Bansod, and L. Mathew, "Unmanned aerial vehicle classification, applications and challenges: A review," 11 2018. 15

[32] "Px4 autopilot dev guide (master) :simulation." https://docs.px4.io/master/en/simulation/ , Accessed: 06.06.2022. 18

[33] "Px4 autopilot user guide (master)." https://docs.px4.io/master/en/ , Accessed: 06.06.2022. 19, 21

[34] "Ardupilot documentation." https://ardupilot.org/ardupilot/ , Accessed: 06.06.2022. 20

[35] "Px4-autopilot github page license." https://github.com/PX4/PX4-Autopilot/blob/master/LICENSE , Accessed: 06.06.2022.

[36] "Ardupilot documentation license (gplv3)." https://ardupilot.org/dev/docs/license-gplv3.html , Accessed: 06.06.2022.

[37] "Px4 autopilot user guide airframe/vehicle builds." https://docs.px4.io/v1.9.0/en/airframes/airframe$_{r}ef erence.html, Accessed$: 06.06.2022.

[38] "Px4 autopilot user guide flight modes." https://docs.px4.io/v1.9.0/en/flight$_{m}odes/, Accessed$: 06.06.2022. 21

[39] "Mavlink developer guide." url = https://mavlink.io/en/ , Accessed: 06.06.2022. 22

[40] "Qgroundcontrol user guide." url = https://docs.qgroundcontrol.com/master/en/index.html , Accessed: 06.06.2022. 23

[41] A. I. Hentati, L. Krichen, M. A. Fourati, and L. Chaari, "Simulation tools, environments and frameworks for uav systems performance analysis," *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pp. 1495–1500, 2018. 23, 24, 27, 28, 29, 30, 31, 32

[42] "Ardupilot mission planner documentation." url = https://ardupilot.org/planner/docs/mission-planner-overview.html , Accessed: 06.06.2022. 24

[43] "Ardupilot mavproxy documentation." url =https://ardupilot.org/mavproxy/ , Accessed: 06.06.2022. 25

[44] "Gazebo official website ,documentation." url =https://gazebosim.org/docs , Accessed: 06.06.2022. 27

[45] "Ros documentation." http://wiki.ros.org/Documentation, Accessed: 06.06.2022. 27

[46] "Airsim official website." url = https://microsoft.github.io/AirSim/ , Accessed: 06.06.2022. 28

[47] "jmavsim official github page." url = https://github.com/PX4/jMAVSim , Accessed: 06.06.2022. 29

[48] "Px4 developer documentation :jmavsim with sitl." url = https://dev.px4.io/v1.11$_n$*oredirect/en/simulation/jmavsim.html*,*Accessed* : 06.06.2022. 29

[49] "X-plane developer documentation." url = https://developer.x-plane.com/docs/ , Accessed: 06.06.2022. 30

[50] "Flightgear flight simulator official website." url = https://www.flightgear.org/about/ , Accessed: 06.06.2022. 31

[51] "Px4 dev guide simulation : Flightgear." url = https://docs.px4.io/master/en/simulation/flightgear$_v$*ehicles.html*,*Accessed* : 06.06.2022. 31

[52] "Flightgear flight simulator official website:features." url = https://www.flightgear.org/about/features/ , Accessed: 06.06.2022. 31

[53] "Matplotlib 3.5.0 documentation." url = https://matplotlib.org/3.5.0/api/_as_gen/matplotlib.pyplot.html , Accessed: 06.06.2022. 34, 40

[54] "Numpy documentation." url = https://numpy.org/doc/stable/ , Accessed: 06.06.2022. 34

[55] "pandas documentation." url = https://pandas.pydata.org/docs/ , Accessed: 06.06.2022. 35

[56] A. Sakai, D. Ingram, J. Dinius, K. Chawla, A. Raffin, and A. Paques, "Pythonrobotics: a python code collection of robotics algorithms," 2018. 35

[57] "jxmapviewer2 official github page." url = https://github.com/msteiger/jxmapviewer2 , Accessed: 06.06.2022. 35

[58] "Java api for json processing: An introduction to json." url = https://www.oracle.com/technical-resources/articles/java/json.html , Accessed: 06.06.2022. 35

[59] J. Wu, J. Ma, Y. Rou, L. Zhao, and R. Ahmad, "An energy-aware transmission target selection mechanism for uav networking," *IEEE Access*, vol. 7, pp. 67367–67379, 2019. 37

[60] M.-H. Hwang, H.-R. Cha, and S. Jung, "Practical endurance estimation for minimizing energy consumption of multirotor unmanned aerial vehicles," *Energies*, vol. 11, p. 2221, 08 2018. 37

Use case diagram of UAVs Mobility Simulator

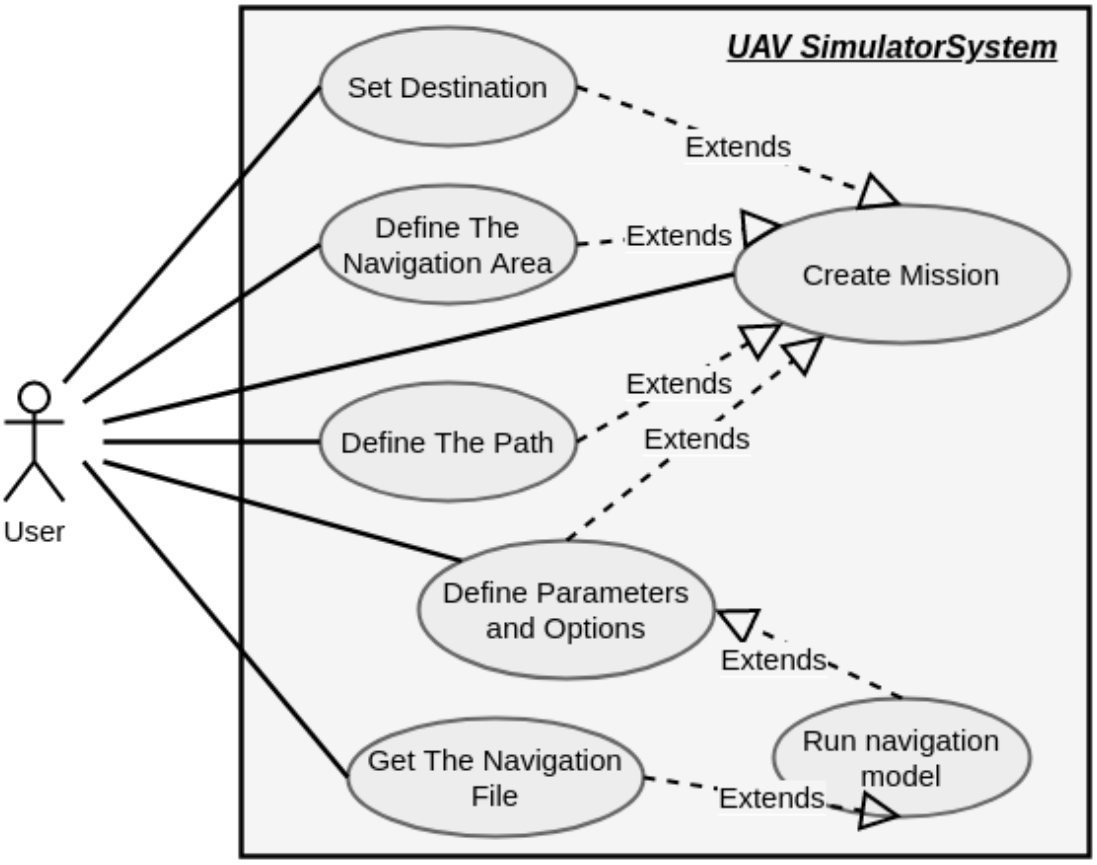

Figure 24: Use case diagram

Class diagram of UAVs Mobility Simulator

.1 First section

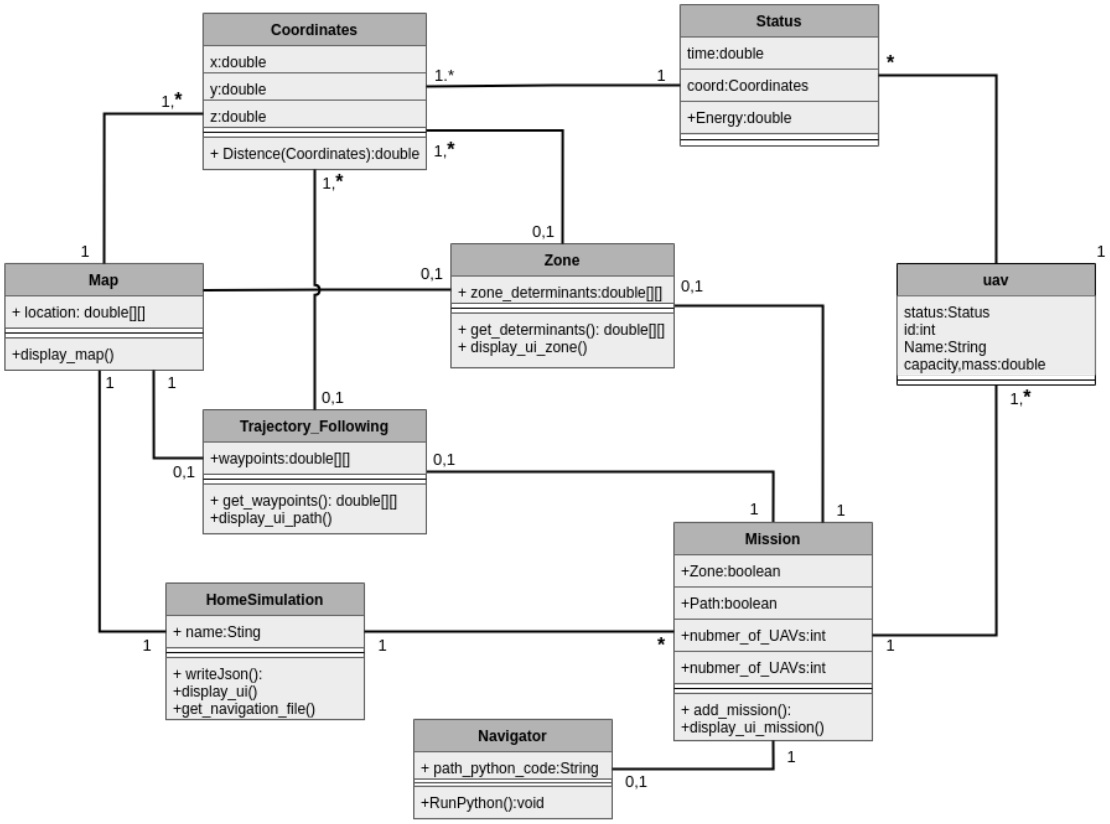

Figure 25: Class diagram of UAVs Mobility Simulator GUI section

.2 Second section

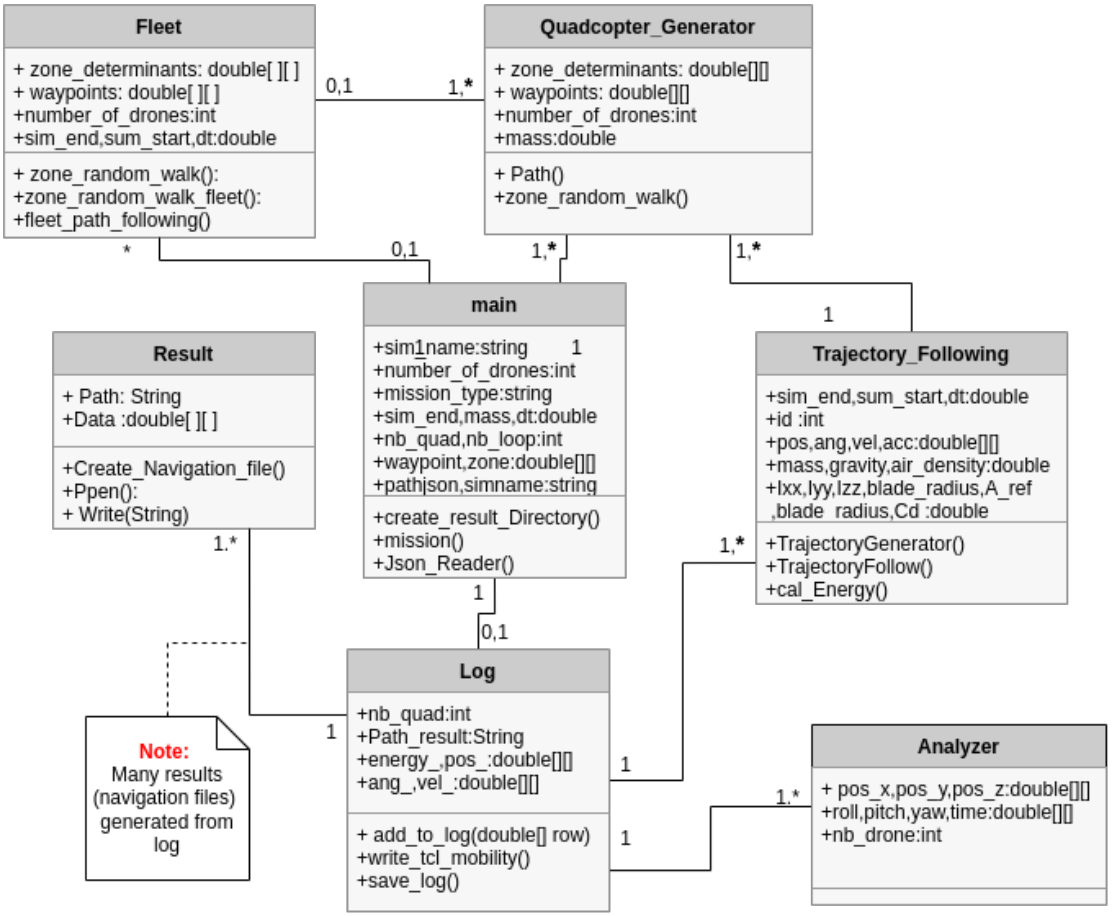

Figure 26: Class diagram of UAVs Mobility Simulator mobility section