

ASSISTANCE

Adapted situation awareneSS tools and tailored training curricula for increaSiNg capabiliTies and enhANcing the proteCtion of first respondErs



European Commission

Project co-funded by the European Union within the Horizon 2020 Programme



Project Ref. N°	ASSISTANCE H2020 - 832576
Start Date / Duration	May 1, 2019 (36 months)
Dissemination Level¹	CO (Confidential)
Author / Organisation	UPVLC

Deliverable D4.2

UAVs integrated into the system

31/01/2021

¹ PU: Public; PP: Restricted to other programme participants (including the EC services); RE: Restricted to a group specified by the Consortium (including the EC services); CO: Confidential, only for members of the Consortium (including the EC services).

ASSISTANCE

Nowadays different first responder (FR) organizations cooperate together to face large and complex disasters that in some cases can be amplified due to new threats such as climate change in case of natural disasters (e.g. larger and more frequent floods and wildfires, etc.) or the increase of radicalization in case of man-made disasters (e.g. arsonists that burn European forests, terrorist attacks coordinated across multiple European cities).

The impact of large disasters like these could have disastrous consequences for the European Member States and affect social well-being on a global level. Each type of FR organization (e.g. medical emergency services, fire and rescue services, law enforcement teams, civil protection professionals, etc.) that mitigate these kinds of events are exposed to unexpected dangers and new threats that can severely affect their personal safety.

ASSISTANCE proposes a holistic solution that will adapt a well-tested situation awareness (SA) application as the core of a wider SA platform. The new ASSISTANCE platform is capable of offering different configuration modes for providing the tailored information needed by each FR organization while they work together to mitigate the disaster (e.g. real-time video and resources location for firefighters, evacuation route status for emergency health services and so on).

With this solution, ASSISTANCE will enhance the SA of the responding organizations during their mitigation activities through the integration of new paradigms, tools and technologies (e.g. drones/robots equipped with a range of sensors, robust communications capabilities, etc.) with the main objective of increasing both their protection and their efficiency.

ASSISTANCE will also improve the skills and capabilities of the FRs through the establishment of a European advanced training network that will provide tailored training based on new learning approaches (e.g. virtual, mixed and/or augmented reality) adapted to each type of FR organizational need and the possibility of sharing virtual training environments, exchanging experiences and actuation procedures.

ASSISTANCE is funded by the Horizon 2020 Programme of the European Commission, in the topic of Critical Infrastructure Protection, grant agreement 832576.

Disclaimer

This document contains material, which is the copyright of certain ASSISTANCE consortium parties, and may not be reproduced or copied without permission.

The information contained in this document is the proprietary confidential information of the ASSISTANCE consortium (including the Commission Services) and may not be disclosed except in accordance with the consortium agreement.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the project consortium as a whole nor a certain party of the consortium warrant that the information contained in this document is capable of use, nor that use of the information is free from risk and accepts no liability for loss or damage suffered by any person using this information.

The information in this document is subject to change without notice.

Executive Summary

This document gives a summary of the work carried out in the task T4.2: UAV Management and Sensors Integration. This deliverable provides a description of the integration of the UAVs and sensors integrated in them with the rest of ASSISTANCE system through the Sensor Abstraction Service (SAS). The UAVs' main goal is to support first responders by providing them with information, gathered through onboard sensor systems, about the current state of the situation and environment impacted by a disaster. For this purpose, these equipment and sensors must be integrated with the rest of modules of the ASSISTANCE system so that the information can reach the users correctly. This document gives a clear description of this integration according to the following tasks:

- integration of drone telemetry data to the system through the SAS
- integration of onboard sensors (gas sensor) to the system through the SAS
- integration of video frames and video streaming taken from onboard cameras through the SAS
- reception of mission plans through the SAS

Apart from that, this document will also describe the ground segment of the UAVs. This type of aircraft usually has a ground equipment called Ground Control Station (GCS) from which the pilot can interact remotely with the air platform. From this component, the mission received is transmitted to the drone and monitored by the drone pilot. Apart from that, in the GCS all the telemetry data from the drone and onboard sensors can be monitored by the drone pilot.

List of Authors

Organisation	Authors
FADA-CATEC	Manuel García
FADA-CATEC	Leandro Candau

Change control datasheet

Version	Changes	Chapters	Pages	Date
0.1	First draft	All	20	30/11/20
1.0	First version for review	All	41	20/01/21
2.0	Second version after revision	All	42	28/01/21

Table of contents

- Executive Summary4
- 1 Introduction10
 - 1.1 Purpose of the document 10
 - 1.2 Scope..... 11
 - 1.3 Relationship with other work packages..... 11
- 2 ASSISTANCE General architecture.....12
- 3 Scenarios implementation14
 - 3.1 Scenario 1..... 14
 - 3.2 Scenario 2..... 15
 - 3.3 Scenario 3..... 16
- 4 GCSs.....17
 - 4.1 DJI GCS 17
 - 4.2 CATEC GCS..... 22
- 5 Communications interface with the SAS.....31
 - 5.1 Description of the interfaces 31
 - 5.2 Downloading the mission 32
 - 5.3 Telemetry data publisher..... 35
 - 5.4 Sensor data publisher 37
 - 5.5 Streaming interface 39
- 6 Conclusions41

List of Figures

Figure 1: Platform high level design schema	12
Figure 2: ASSISTANCE Architecture Scheme	13
Figure 3: Streaming of the video to the ASSISTANCE system	15
Figure 4: Tablet connected to the DJI radio transmitter	17
Figure 5 DJI GS Pro mission page.....	18
Figure 6: Waypoint setting interface.....	19
Figure 7: Parameters setting screen.....	20
Figure 8: Waypoint edit bottoms	21
Figure 9: DJI GS Pro during the operation	21
Figure 10: Main interface of the GCS application in mission mode	22
Figure 11: Waypoints editor	23
Figure 12: GCS toolbar	24
Figure 13: Alarms bar	24
Figure 14: Primary Flight Display	25
Figure 15: Actions tab.....	25
Figure 16: Telemetry tab	26
Figure 17: Parameters tab	26
Figure 18: Flight Control console.....	27
Figure 19: MAVLink protocol v1.0 format.	28
Figure 20: Return To Launch MAVLink command.	29
Figure 21: Mission Start MAVLink command.	29
Figure 22: Change Speed MAVLink command.	29
Figure 23: Attitude MAVLink message.	29
Figure 24: Heartbeat MAVLink message.	30
Figure 19: Data flow diagram.	31
Figure 20: Sending data to the SAS.	32
Figure 21: Downloading a mission from the SAS.	33
Figure 22: GCS application with a loaded mission.	34
Figure 29: Sample mission JSON downloaded from SAS.....	35
Figure 24: Configuring onboard computer and sending telemetry.	36
Figure 31: Sample telemetry data JSON sent to SAS.....	37
Figure 32: Sample gas sensor data JSON sent to SAS.....	38
Figure 33: Sample camera frame sent to SAS.	39
Figure 34: Streaming video to the SAS.	40

Acronyms

API	Application Program Interface
ASSISTANCE	Adapted situation awareneSS tools and tallored training curricula for increaSing capabiliTie and enhANcing the proteCtion of first respondErs
COTS	Commercial Off-The-Shelf
D#.#	Deliverable number #.# (D1.1 deliverable 1 of work package 1)
GCS	Ground Control Station
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HMI	Human Machine Interface
IP	Internet Protocol
JSON	JavaScript Object Notation
MQTT	Message Queue Telemetry Transport
MMM	Mission Management Module
NED	North East Down
OSDK	Onboard Software Development Kit
PC	Computer
RC	Radio Control
REST	Representational State Transfer
ROS	Robot Operating System
RTSP	Real Time Streaming Protocol
RPA	Remotely Piloted Aircraft
SA	Situation Awareness
SAS	Sensors Abstraction Service
SDI	Serial Digital Interface
SDK	Software Development Kit
UAV	Unmanned Aerial Vehicle
USB	Universal Serial Bus
WPX	Work Package X

1 Introduction

This document describes the integration of the UAVs and its sensors with the rest of ASSISTANCE modules through the Sensor Abstraction Service (SAS), according to the architecture designed in deliverable D2.4 ASSISTANCE System and Network Architecture Design. This task is essential so that First Responders (FRs) can access the information collected by UAVs about the situation in the disaster area. Nowadays, many FRs companies are acquiring drones to support their daily tasks, but they lack the integration of specific sensors that is carried out in this project that would allow them to obtain the maximum amount of information. Taking into account the selection of platforms made in task T4.1, as well as the interfaces for the exchange of messages between the different modules defined in task T3.1, and their implementation in task T3.2, it is possible to address the integration of the sensors and equipment on board the UAVs with the SAS. The main information that will be exchanged among the UAVs and the SAS is the following:

- Telemetry data: position, velocity, attitude of the UAV
- Sensor data: mainly data of presence of CO₂ by means of a gas sensor
- Video data: the UAVs have cameras onboard and the images must be sent to the rest of the ASSISTANCE system
- Mission commands: the UAV must accept and execute a list of waypoints defined by the user, which must be received in the UAV through the SAS

Furthermore, in this document, the description of the Ground Control Station (GCS) of each aerial platform is provided. The main specifications and functionalities of these systems that allow the remote control of the aerial platforms are given. This document will also help to understand the operation of drones in the context of ASSISTANCE project.

As a general remark, this document shows graphs and pictures which will help the reader to understand this integration and operation from the GCS.

1.1 Purpose of the document

The main purpose of this deliverable is to describe the integration of the UAVs and its sensors with the SAS according to the interfaces defined in T3.1. In addition to this, the ground equipment used to operate the aircraft by a pilot is described. This ground equipment is also used as an interface by the UAV pilot to check aircraft parameters during the flight. These GCSs are connected to the UAVs through communication links which must be bidirectional: downlink to get telemetry data from the aircraft, and uplink to send pilot commands in case of necessity.

Depending on the scenarios described in D2.3, the requirements for each integration are different, and the solutions proposed for these integrations have specific particularities. It is the purpose of this document to detail the levels of integration for the different scenarios. In this way, the drones and equipment selected in deliverable D4.1 will be taken as inputs.

Hence, this deliverable covers the output of the Task 4.2 UAV Management and Sensors Integration. In the following sections, this deliverable's scope and structure are described, together with its relationship with other ASSISTANCE work packages.

1.2 Scope

This deliverable includes the description of the ground segment of the UAVs and the integration of their equipment with the rest of ASSISTANCE system, being one of the fundamental tasks of WP4. Firstly, a description of the ASSISTANCE architecture and how this architecture relates to this integration is shown, according to what was explained in D2.4. Next, the implementation of the different scenarios is described, in order to put in context the integration explained in the next sections. Section 4 shows the description of the GCSs which are used in the different scenarios of the project. Section 5 defines the integration of the different components of the drone part with the SAS according to the architecture previously defined. Finally, in Section 6, some conclusions are given.

1.3 Relationship with other work packages

This deliverable gets information from the following tasks:

- Task 2.2 User requirements gathering analysis and tracking
- Task 2.3 Reference scenarios, pilot operations specifications and KPIs.
- Task 3.1 Sensor Abstraction Service Adapted Interfaces Definition
- Task 3.2 Sensor Abstraction Service Adapted Interfaces Implementation
- Task 4.1 Unmanned Platforms Selection & Adaptation

The output from the task 4.2 and consequently this deliverable D4.2 contributes to the following tasks:

- Task 4.6 Mission management
- Task 5.1 ASSISTANCE SA platform adaptation
- Task 5.2 SA advanced modules development
- Task 5.3 Robust Land Mobile Communications Infrastructure Development
- Task 5.4 Advanced Modules, SAS & Communications Infrastructure Integration in ASSISTANCE SA Platform
- Task 7.1 Validation Plan
- Task 7.2 Integrated system test bed
- Task 7.3 Pilot Demonstration

2 ASSISTANCE General architecture

Deliverable D2.4 offers a high-level description of the ASSISTANCE system, with a representation of the role that unmanned platforms are going to play (see Figure 1). In this scheme, it is possible to see how the unmanned platforms (green shaded), together with the sensors they will have to mount, are a tool that through the Sensor Abstraction Service (SAS) will provide the necessary data (telemetry, images, sensor data, etc.) to the end-users through the Human Machine Interface (HMI).

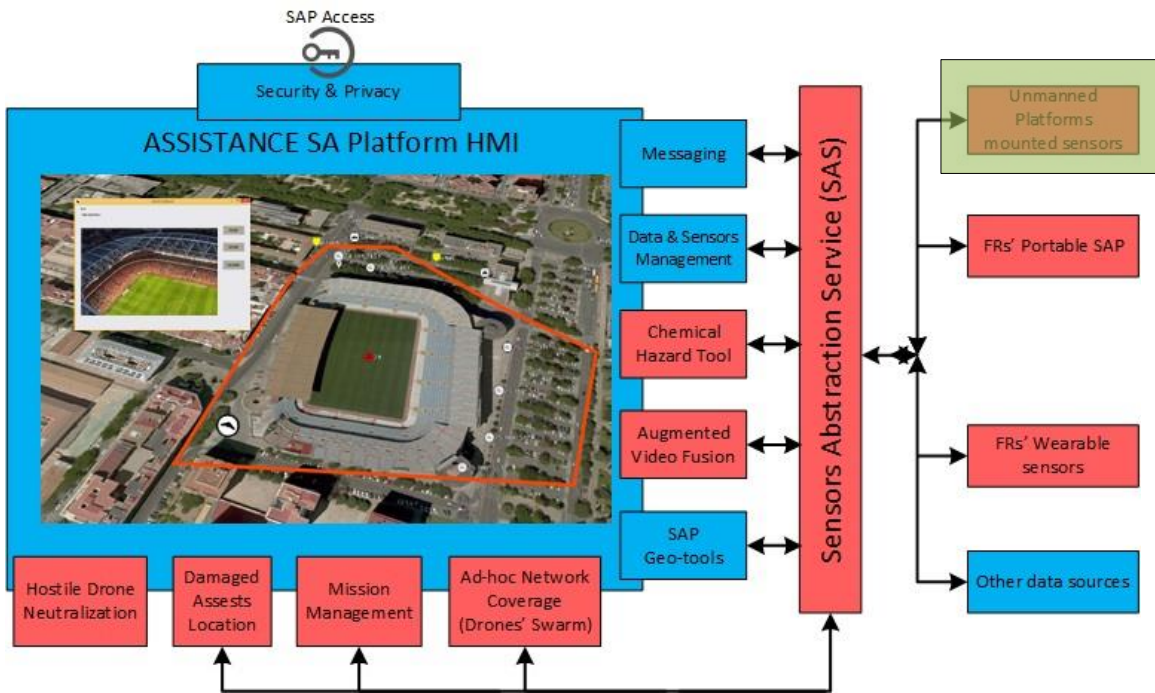


Figure 1: Platform high level design schema

As for the definition of the ASSISTANCE system architecture, Figure 2 shows the integration of all modules and components, with drones at the top (again green shaded). In this scheme, extracted from the D2.4 deliverable, it is possible to observe the general interfaces that these unmanned platforms must have in order to exchange with the SAS, which will be the communication link with the rest of the ASSISTANCE system modules. The main interfaces are:

- Sensor data: the main sensors that the aerial platforms will mount are:
 - o Cameras: video images of the situation to the end users.
 - o Thermal cameras: thermal sensor to detect high temperature areas.
 - o Gas sensor: sensor to detect CO & CO2 concentration.
- Mission data: it is necessary to exchange the flight path that the platform must go through and receive telemetry data:
 - o Flight path: list of waypoints that the robot must go through. This flight path is received in the robots through the SAS, from the Mission Management Module (MMM).
 - o Telemetry data: status data of the robot, such as position, velocity, attitude, etc. This data will be sent to the MMM through the SAS.

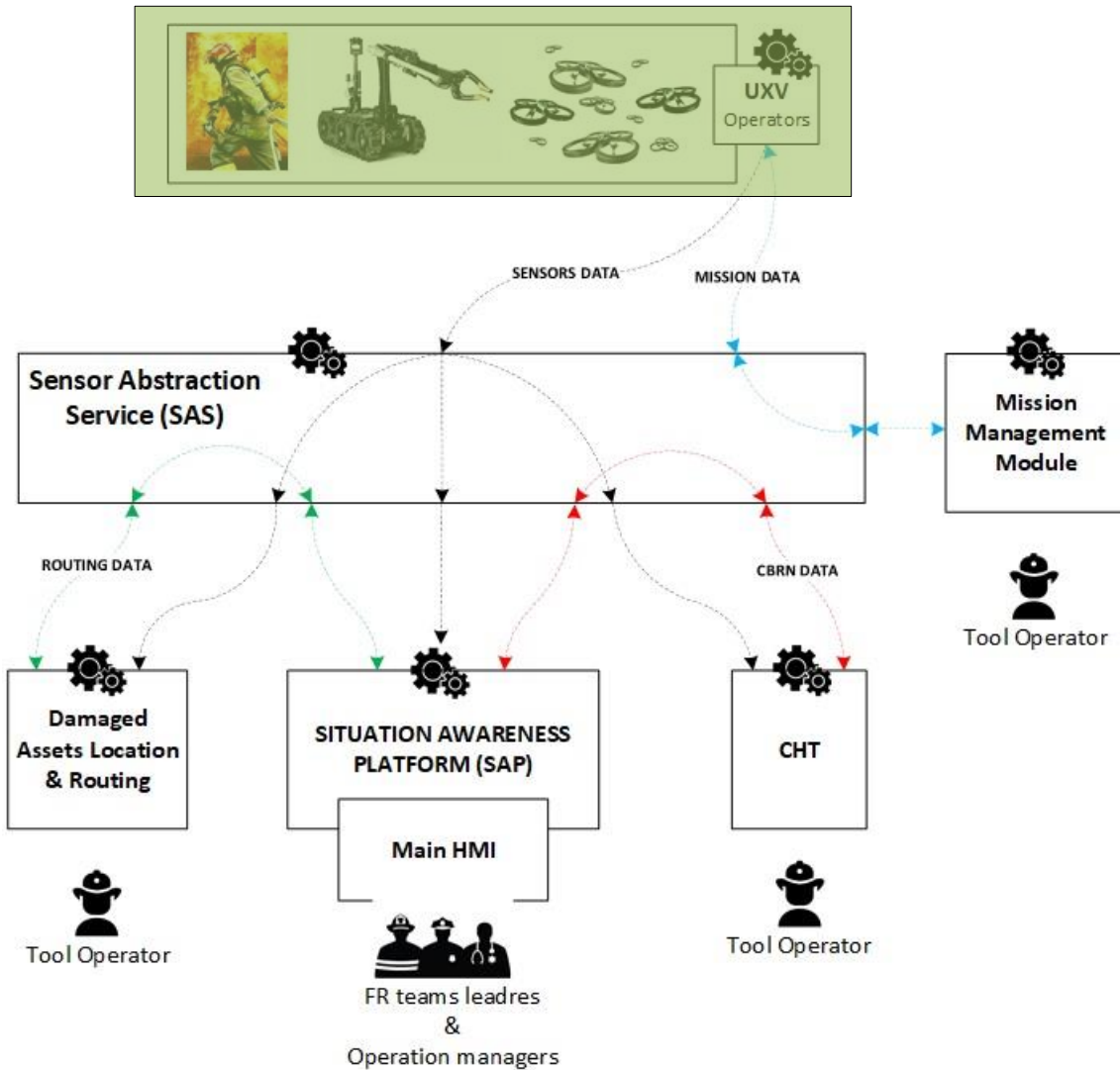


Figure 2: ASSISTANCE Architecture Scheme

Beyond content from D2.4, it is also important to highlight, as inputs for this deliverable, the use cases and functionalities described for each of the robots being considered.

Deliverable D4.1 defines the components that are involved in a generic operation of a UAV and that make up the complete system. To summarize, the three main elements in this type of system are:

- the aircraft or drone that executes the mission and has on board the sensors and equipment necessary to remotely interact with it,
- the ground control station (GCS) that serves as an interface for the pilot to interact with the aircraft,
- the communications link between the aircraft and the GCS.

The following sections will describe, for each scenario, how the integration of the drones and their sensors with the SAS will be performed, according to the architecture shown in Figure 2 and according to the interfaces described in deliverable D3.1. In addition, the GCSs to be used in each scenario will be described, which depends on the autopilot on board the aircraft.

3 Scenarios implementation

3.1 Scenario 1

Deliverable D4.1 proposes the use of a drone DJI Phantom 4 with an integrated telemetry module based on a Pixhawk autopilot (see section 3.2.2 of D4.1). Since the proposed DJI Phantom 4 drone, which is the one the Turkish partner AAHD has for the first scenario, is a COTS drone that does not allow data exchange with it, the PIXHAWK telemetry module will be in charge of sending the flight data to the SAS. Furthermore, the execution of missions by this drone from lists of waypoints received from the SAS will be carried out through the mediation of the drone's safety pilot, who will be responsible for entering the mission into the drone's GCS after validating the mission received from the SAS. The telemetry data will be given by the telemetry module based on PIXHAWK autopilot (see Section 3.2.2 in D4.1). This autopilot uses MAVLINK protocol for communications. The autopilot is connected to a small onboard PC from which this telemetry data is sent to the SAS. This integration is explained in sections 5.2 and 5.3.

As for the reception of the mission commands, these will come in the form of a list of waypoints through which the drone must pass autonomously. This list of waypoints will be provided to the drone through the GCS of the drone, after the proper validation by the safety pilot. The GCS is described in section 4.1, where the introduction of waypoints is explained.

In this scenario the interest was centered on obtaining images using the camera available on board the aircraft. In this case, only the streaming of the images taken by the drone camera is required. For this purpose, a framegrabber is used, which is a small device that captures video from nearly any source. Once captured, it is possible to send that video to a computer, stream it using VLC, Webex, or other software. The proposed framegrabber is Epiphan SDI2USB 3.0, which is connected via SDI to the GCS, and via USB to the computer. The integration of the video stream into the SAS is explained in Section 5.5.

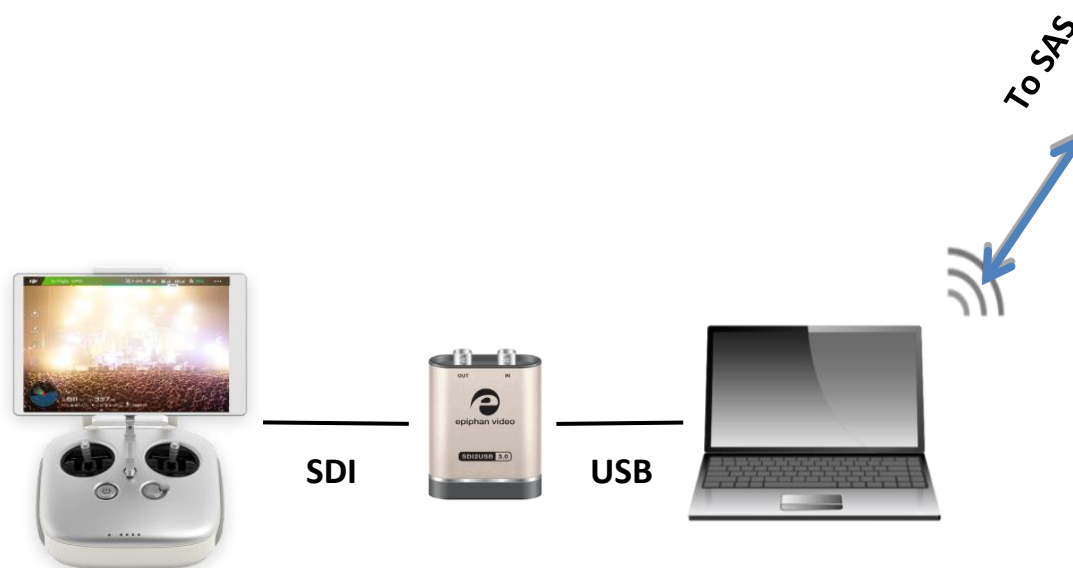


Figure 3: Streaming of the video to the ASSISTANCE system

3.2 Scenario 2

Deliverable D4.1 proposes the use of a drone DJI Matrice 200 with a DJI autopilot onboard from which telemetry can be obtained. The onboard PC will be in charge of receiving onboard telemetry using the SDK from DJI for developers in Robot Operating System (ROS). The DJI SDK is a software development kit that allows the creation of new applications / developments that interact with DJI drones or autopilots. DJI offers different SDKs for the development of applications on mobile platforms, with Windows, for new payloads, etc. In this case the Onboard SDK will be used, which allows the connection of the onboard computer to a supported DJI vehicle or flight controller using a serial port. More information can be found in: <https://developer.dji.com/onboard-sdk/>

In this case, sending the telemetry, the data from the sensors and video frames will be carried out as indicated in the section 5.3 and 5.4.

The GCS in this case will be CATEC's proprietary one, since the integration with the SAS will be easier in this case. This GCS is described in section 4.2. Since this GCS is developed using MAVLINK protocol, a module to convert from MAVLINK to DJI messages and vice versa will be developed, which is explained in section 4.2.10.

The integration of mission commands and sending of telemetry data and sensors is explained in section 5. The streaming of video is performed in an analogous way to the scenario way, and following the integration explained in section 5.5.

3.3 Scenario 3

The D4.1 deliverable proposes the use of a drone DJI Matrice 600 with a DJI autopilot onboard from which telemetry can be obtained. The configuration is the same as in Scenario 2, so the same GCS and integration strategies are followed.

4 GCSs

Two different GCSs will be used for the drones. For the first scenarios in which a more relaxed integration is performed, the GCS that will be used is DJI's proprietary one that interacts with the drone and with the rest of the system. Instead, for the other two scenarios, the GCS developed by CATEC will be used to exchange the required messages with the SAS and interact with the drones, both from DJI on stage 2 and Pixhawk on stage 3. In this section, both GCSs are explained and defined.

4.1 DJI GCS

The GCS of DJI Phantom 4 consists of a radio transmitter which can be connected to a tablet or mobile phone. In the tablet in Figure 4, the application DJI GS PRO is installed. As can be seen, the GCS is portable, easy to transport, and easy to be deployed in any environment, since it an iPad and the DJI transmitter is all that is needed.



Figure 4: Tablet connected to the DJI radio transmitter

The application has several features, of which only those most of interest to the scenario are presented. The mission page of the application is as shown in Figure 5.

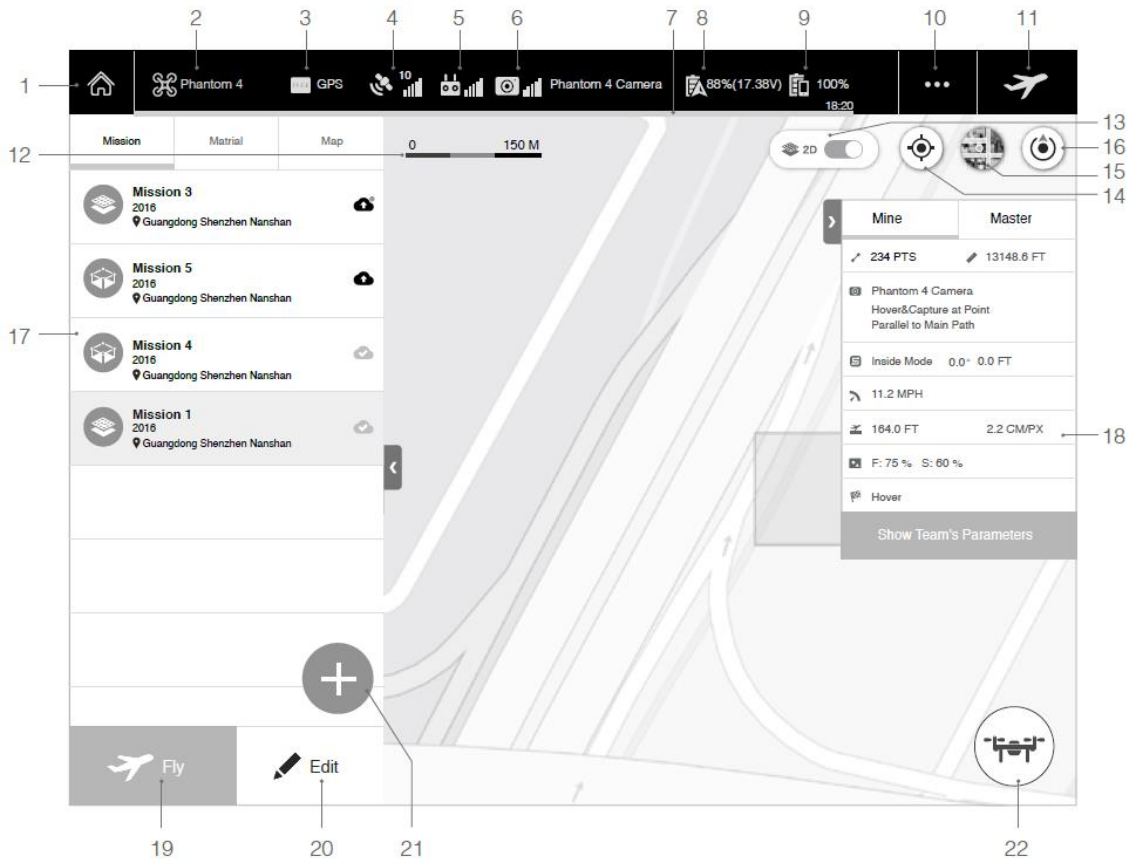






Figure 5 DJI GS Pro mission page.

As can be seen in the image above, this interface gives enough information about the status of both the vehicle and the mission. From the UAV perspective, it shows information about battery level (8), the flight mode (3), the GNSS strength signal (4), camera model (6), etc. As can be seen in the picture (17), the application allows selecting among several missions previously defined and even to add more missions (21). Once a mission is selected, its parameters can be previewed and set (18), a checklist can be entered (11), and then it will be confirmed (19). Once the mission has started, it can be paused, resumed, and finished (11).

The application allows creating four different types of mission, which are:

Mission Type	Icon	Description
PhotoMap		The flight path is automatically generated to complete the photography task. After that, the photomap can be composited automatically and the user can manually calibrate it.

Virtual Fence		To define a specific area of flight, if the flight must be confined within this area or no fly zones are near the flight area. If the UAV approaches the boundaries of the area, it will hover.
3D Map	 	The images captured during the flight could be used to build a 3D reconstruction to generate a 3D map. Another option is to use the 3D Map Point Of Interest, which provides complete accuracy for critical structural management.
Waypoint flight		The user can set a flight plan defining several waypoints (up to 99) with its own parameters and actions, and start flying.

Among all the possibilities that the application provides to define a mission, the one that will be used in this scenario shall be the last one: a mission defined by waypoints. In this type of mission there are several parameters that can be configured. The waypoints setting interface is the one shown in Figure 6. As can be seen, several parameters can be configured for each waypoint.

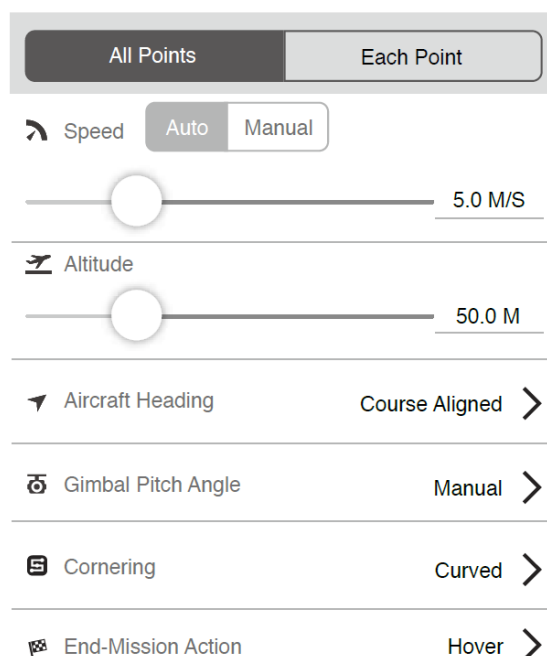


Figure 6: Waypoint setting interface

The parameters that can be set for each waypoint are explained in the following paragraphs.

- **Speed:** either automatic or manually controlled, from 1 to 15m/s.

- **Altitude:** defined as relative to the take-off point, and can be set from 1 to 500m.
- **Aircraft Heading:** can refer to next waypoint (course aligned), be defined per each point, or controlled manually.
- **Gimbal pitch angle:** the pointing angle of the camera, which can be controlled either manually or can be defined per each point.
- **Cornering:** the parameter to control how the aircraft changes direction before flying towards the next point. It can be set as straight or curved. The first one will fly forwards towards the next waypoint, and once it is reached, it will change its heading to face the following waypoint. On the other hand, if the curved option is selected, the aircraft will fly on a smooth curve when passing by a waypoint. In that case, the cornering radius can be determined for each waypoint.
- **End-Mission action:** determines what the aircraft should do once the mission is completed. The different options are Return To Home, Hover, and Land.

Moreover, clicking in the *Each Point* tab, some more options can be set, such as Aircraft rotation direction (clockwise or anticlockwise), gimbal pitch angle, or waypoint action.

Apart from the parameters described, there are other possibilities to be set.

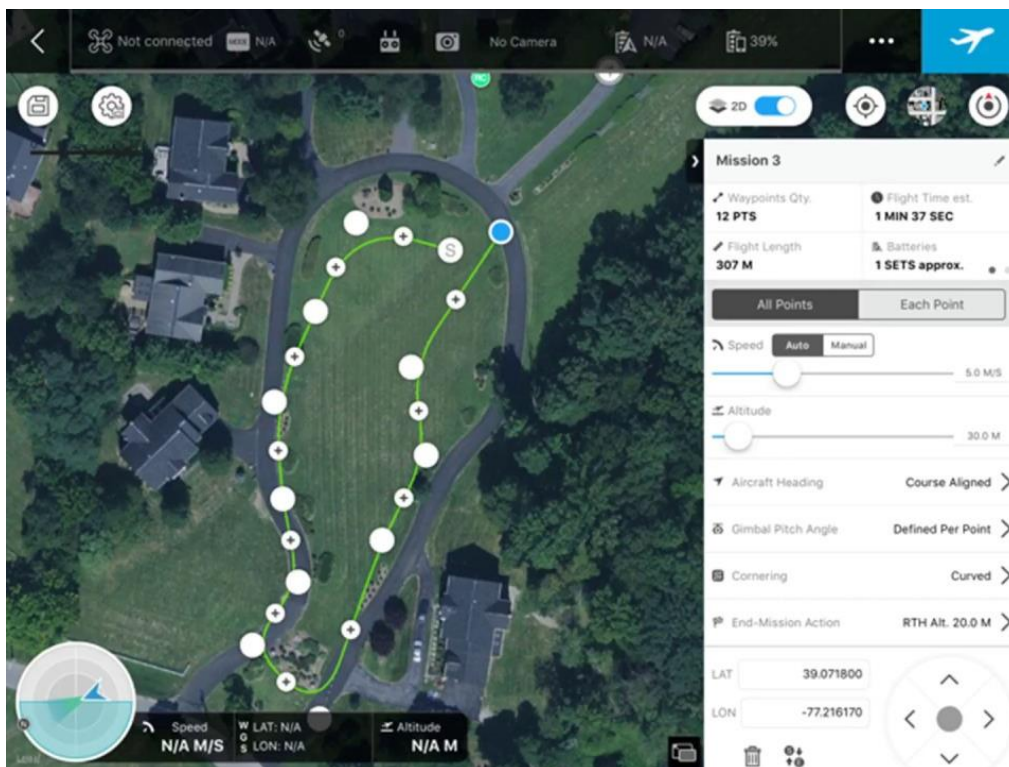


Figure 7: Parameters setting screen

In Figure 7 some of these parameters are shown. At the bottom right of the figure, Waypoint edit bottoms are visible. These are shown in greater detail in Figure 8.

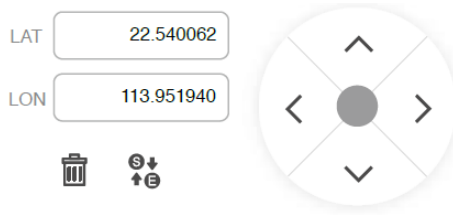


Figure 8: Waypoint edit bottoms

Once the waypoint list has been defined, the operation could be started and performed. The first step will be to adjust the aircraft according to the checklist that appears until all items are green, and after that, the user could tap the *Start to fly* bottom. After that, the procedure of the UAV will vary depending on the definition of the mission.

During flight, it is possible to check the main flight data of the aircraft. Apart from the information which appears in the top black row previously explained, in the bottom part of the interface it is possible to see the waypoint number to which the aircraft is moving at the moment, also information about speed and altitude, and position information. Furthermore, a heading indicator is shown to increase the situational awareness of the pilot.

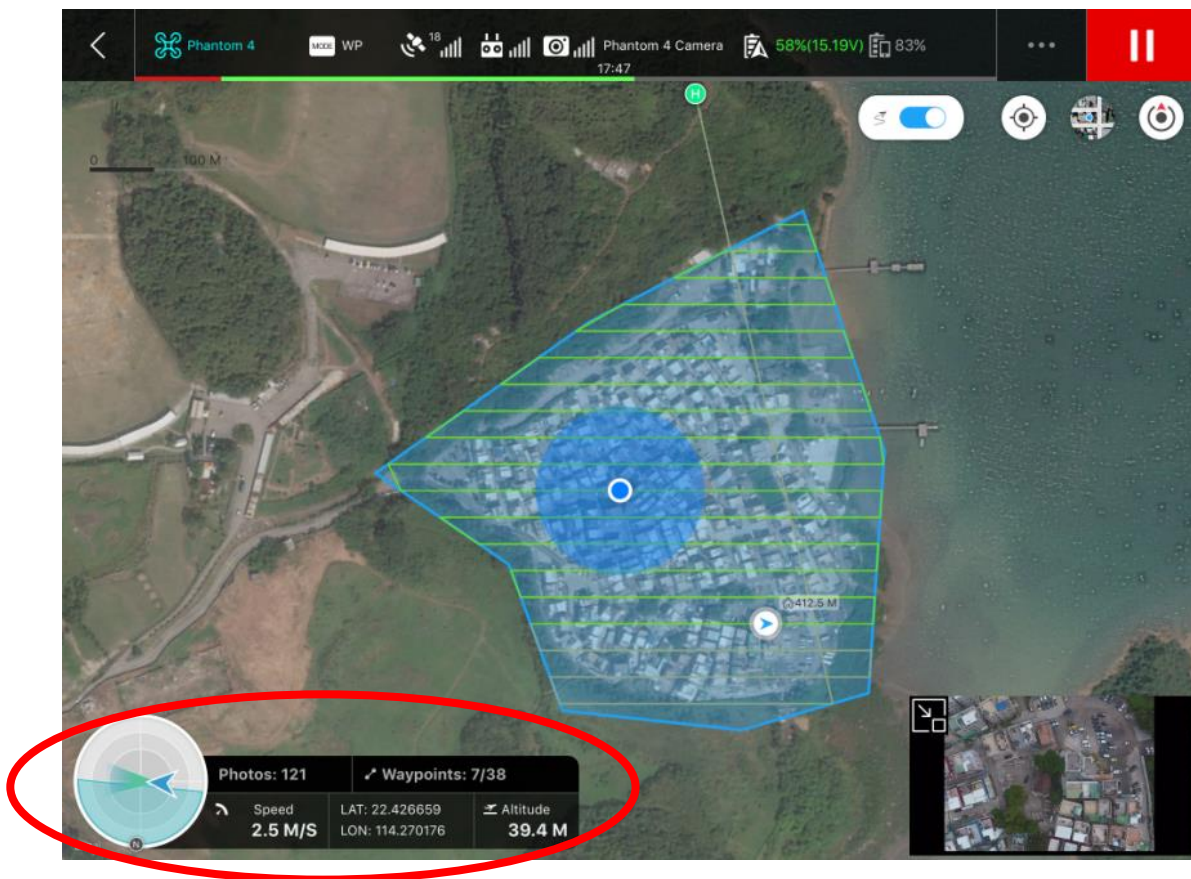


Figure 9: DJI GS Pro during the operation

To sum up, this is a very versatile GCS for piloting DJI aircrafts, with a very user-friendly interface for the operator.

4.2 CATEC GCS

4.2.1 Introduction and installation requirements

The ground control station, GCS, is a tool designed by FADA-CATEC to provide the necessary functionalities to allow human control of UAVs. The system requirements for the installation of this application are

- Operating system Ubuntu 12.04 of 64 GB.
- Intel i5 2.60Ghz processor.
- 8GB of RAM memory.
- Qt Framework 5.4.2 for 64 bits Linux

This GCS is compatible with the MAVLINK communication protocol (<https://mavlink.io/en/>), whose data messages are defined on the following website: <https://mavlink.io/en/messages/common.html>

4.2.2 User interface

The following describes the main elements that make up the user interface and the interactions that can be made on them. Figure 10 shows the general view of the GCS in its two operating modes: flight mode and vision mode, respectively.

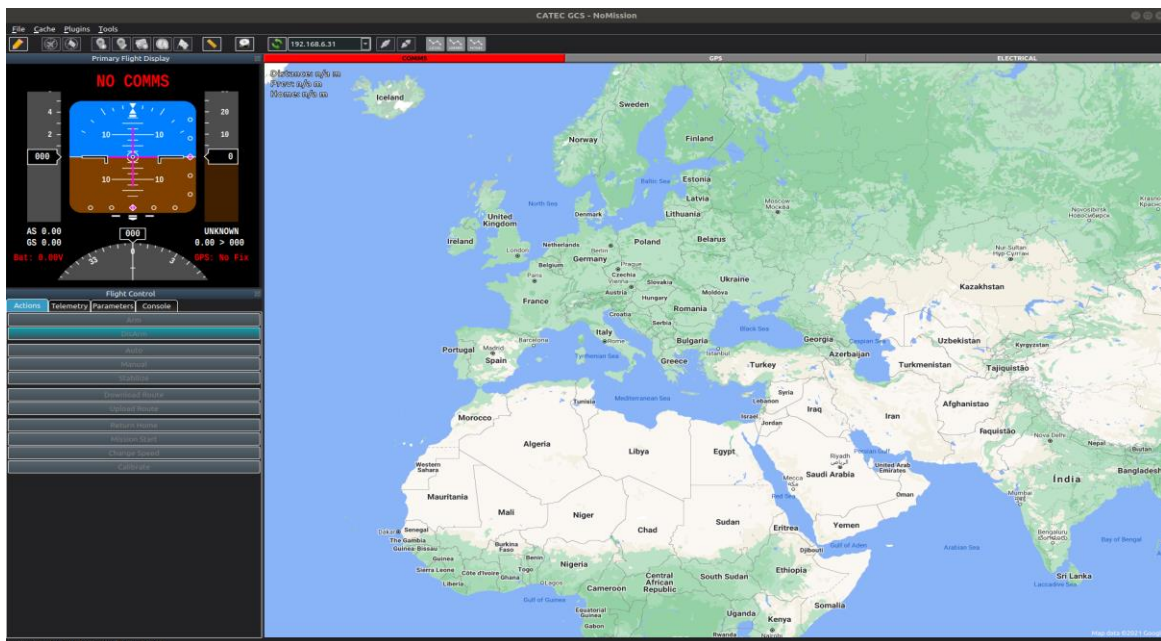


Figure 10: Main interface of the GCS application in mission mode

The main functionalities and capacities of the station are detailed below.

4.2.3 Map caching

As can be seen, most of the display consists of the map on which the operation is to be performed. This map is loaded through Google's servers. Therefore, in order to load these backgrounds, an internet connection is required. In case an internet connection is not available in the flight zone, GCS allows the caching of the ground map of the desired zone.

The map has a legend showing three parameters, visible in its upper left corner, which are:

- Distance: total distance of the selected route.
- Prev: distance between the selected route waypoint and the course position on the map.
- Home: distance between the position the aircraft has set as Home and the position of the cursor on the map.

4.2.4 Flight plan generation

To generate a flight plan it is necessary to place on the map the waypoints through which you want the aircraft to fly. The GCS has an editor of waypoints (see Figure 11) that allows the modification of certain parameters of them, such as height, type, identifier, etc. Additionally, it allows the storage of certain routes to be used later.

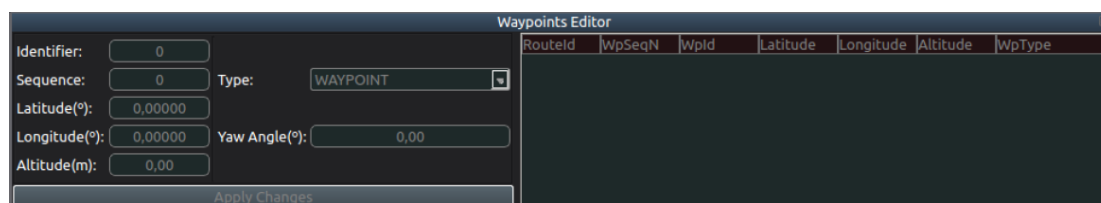


Figure 11: Waypoints editor

It is of interest to understand the different types of waypoints available. These are:

- Waypoint: used when you want to take the aircraft to a specific position. Here, the yaw angle parameter tells the aircraft the orientation it has to maintain until it reaches the point.
- Loiter: used when you want to keep the aircraft in an orbit (defined by the radius parameter) over a specific position for a defined time.
- Loiter time: used to keep the aircraft waiting over a specific position, defined by the delay parameter.
- Land/Takeoff: used to define the landing / takeoff actions of the aircraft.
- Speed: used to modify the speed of the aircraft. The speed parameter indicates the new speed, while the speed type parameter indicates if the speed change is with respect to the air or with respect to the ground.
- Yaw: used to change the orientation of the aircraft when arriving at a certain position. The Angle type parameter indicates if the change will be relative to the

current or absolute orientation; the speed parameter, the speed at which the aircraft should turn; the angle parameter, the new orientation to take; and the direction parameter in which to make the turn, clockwise or counterclockwise.

- Picture: to capture a photo when you reach a certain position.
- Return to launch: to make a return trip home when arriving to a certain position.

4.2.5 Toolbar

The GCS has a useful toolbar (Figure 12) to be used. They are divided into sections formed by sets of buttons, which are described below:



Figure 12: GCS toolbar

- On/Off button and IP selection.
- Waypoint editing buttons.
- Ruler instrument to measure distances on the map
- UAV centering button: it centres the map on the position of the UAV
- Home button: edition of the safety waypoint to which the aircraft will return in case of emergency.

4.2.6 Alarms

The alarm bar has indicators in case of loss of communications, loss or degradation of GNSS reception and electrical alarm, which will be activated in case of low or critical voltage of the autopilot power system. Each indicator shows its current status through colors: green (OK), yellow (WARNING) and red (ERROR).



Figure 13: Alarms bar

4.2.7 Primary flight display

When the GCS is used in flight mode, on the left of the screen the operator will receive the status information of the aircraft through this display. As shown in Figure 14, the operator will receive data on altitude, speed, aircraft attitude, waypoint to which it is heading and distance to that waypoint, flight mode, GNSS quality, battery voltage, etc.

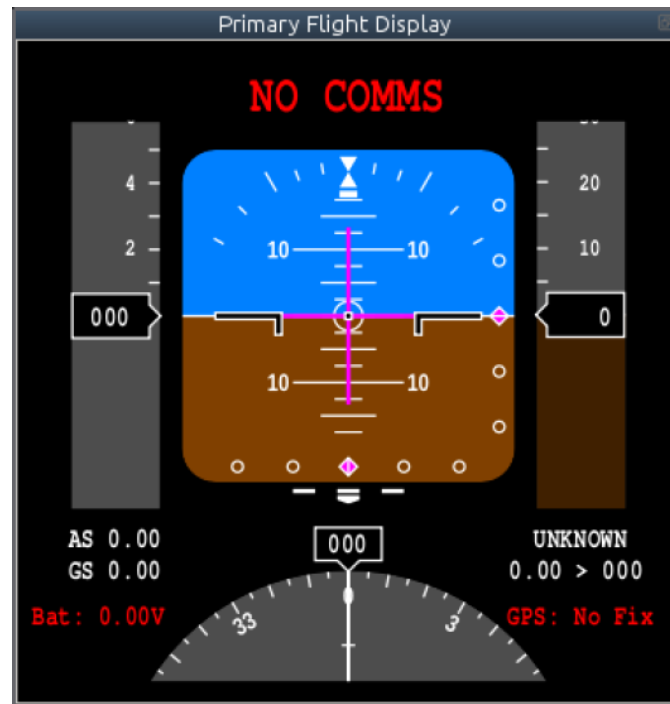


Figure 14: Primary Flight Display

4.2.8 Flight control

The "Flight Control" window is composed of four tabs, through which the user can command actions to the aircraft, visualize the telemetry, configure the parameters of the aircraft and obtain a list of all the messages obtained with each action performed in the GCS. These tabs are:

- Actions (Figure 15): in which there are buttons like start mission, return home or change speed.



Figure 15: Actions tab

- Telemetry (Figure 16): which shows a list with the telemetry received in the GCS.



Figure 16: Telemetry tab

- Parameters (Figure 17): for the configuration of aircraft parameters such as geofencing, return to home, flight speed, etc.

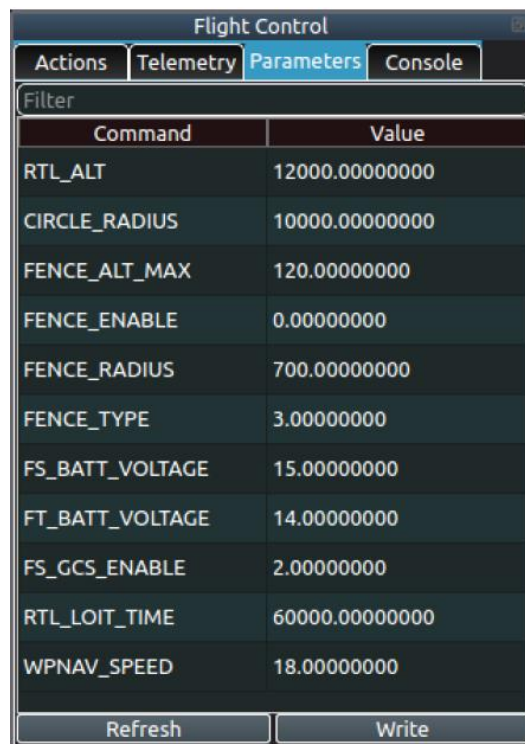


Figure 17: Parameters tab

- Console (Figure 18): where you can view the messages obtained after each action performed by the GCS.

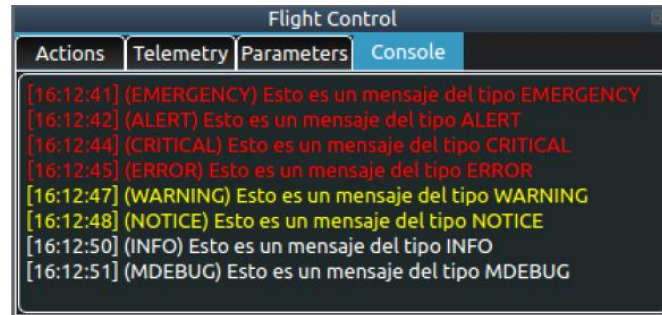
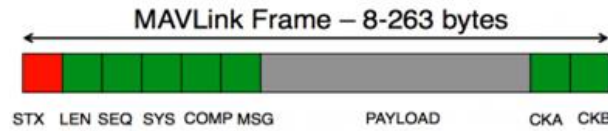


Figure 18: Flight Control console

4.2.9 Integration with MAVLink

MAVLink is a lightweight messaging protocol designed to communicate a GCS with a UAV. It supports common flight commands and status messaging useful for most ground control stations and autopilots, while also supporting the development of custom messages for specific needs. Messages are defined with a simple header of 6 bytes, 9 bytes for the second version of the protocol, followed by the payload of the message, that can take up to 255 bytes, hence it is a lightweight protocol, and a confirmation checksum at the end, Figure 19. There are common messages defined by the MAVLink project on <https://mavlink.io/en/messages/common.html>, that include GPS info, telemetry from the UAV, commands for the drone, mission items and many more.



Byte Index	Content	Value	Explanation
0	Packet start sign	v1.0: 0xFE (v0.9: 0x55)	Indicates the start of a new packet.
1	Payload length	0 - 255	Indicates length of the following payload.
2	Packet sequence	0 - 255	Each component counts up his send sequence. Allows to detect packet loss
3	System ID	1 - 255	ID of the SENDING system. Allows to differentiate different MAVs on the same network.
4	Component ID	0 - 255	ID of the SENDING component. Allows to differentiate different components of the same system, e.g. the IMU and the autopilot.
5	Message ID	0 - 255	ID of the message - the id defines what the payload "means" and how it should be correctly decoded.
6 to (n+6)	Data	(0 - 255) bytes	Data of the message, depends on the message id.
(n+7) to (n+8)	Checksum (low byte, high byte)	ITU X.25/SAE AS-4 hash, excluding packet start sign, so bytes 1..(n+6) Note: The checksum also includes MAVLINK_CRC_EXTRA (Number computed from message fields. Protects the packet from decoding a different version of the same packet but with different variables).	

Figure 19: MAVLink protocol v1.0 format.

CATEC GCS has a modular communication plugin that supports sending and receiving MAVLink messages to and from the autopilot. These messages are used to update the main graphic user interface and can be sent using the action buttons on it. MAVLink defines common messages that have implementation of the GCS commands such as “Return To Launch”, Figure 20, “Mission Start”, Figure 21, or “Change Speed”, Figure 22, found in the Actions widget. When a mission is loaded in the GCS, that can be loaded from the SAS or set manually, MAVLink messages are used to set the mission waypoints into the autopilot on the UAV, and a command is sent to start and control the mission. Going to a specific waypoint is also supported by selecting it on the map from the user interface, via a MAVLink command to the autopilot. Command messages are a particular message that supports up to seven parameters per command, “Command Long” MAVLink message where the command ID is set, defining the type of command sent and what parameters to expect.

MAV_CMD_NAV_RETURN_TO_LAUNCH (20)

[Command] Return to launch location

Param (:Label)	Description
1	Empty
2	Empty
3	Empty
4	Empty
5	Empty
6	Empty
7	Empty

Figure 20: Return To Launch MAVLink command.

MAV_CMD_MISSION_START (300)

[Command] start running a mission

Param (:Label)	Description	Values
1: First Item	first_item: the first mission item to run	min:0 increment:1
2: Last Item	last_item: the last mission item to run (after this item is run, the mission ends)	min:0 increment:1

Figure 21: Mission Start MAVLink command.

MAV_CMD_DO_CHANGE_SPEED (178)

[Command] Change speed and/or throttle set points.

Param (:Label)	Description	Values	Units
1: Speed Type	Speed type (0=Airspeed, 1=Ground Speed, 2=Climb Speed, 3=Descent Speed)	min:0 max:3 increment:1	
2: Speed	Speed (-1 indicates no change)	min: -1	m/s
3: Throttle	Throttle (-1 indicates no change)	min: -1	%
4: Relative	0: absolute, 1: relative	min:0 max:1 increment:1	
5	Empty		
6	Empty		
7	Empty		

Figure 22: Change Speed MAVLink command.

The UAV reports its status through the autopilot to the GCS, where MAVLink messages are received such as “Attitude”, or “GPS Raw Int”. These messages contain information of the status measured by the UAV sensors used to represent it on the user interface, the primary flight display and the telemetry widget. For instance, the UAV sends its telemetry periodically with the “Attitude” message, as seen on Figure 23, from where the roll, pitch and yaw are taken to update the primary flight display and telemetry widgets. To maintain the connection between the GCS and the autopilot a “Heartbeat” message is also sent at 1Hz, shown in Figure 24.

ATTITUDE (#30)

[Message] The attitude in the aeronautical frame (right-handed, Z-down, X-front, Y-right).

Field Name	Type	Units	Description
time_boot_ms	uint32_t	ms	Timestamp (time since system boot).
roll	float	rad	Roll angle (-pi..+pi)
pitch	float	rad	Pitch angle (-pi..+pi)
yaw	float	rad	Yaw angle (-pi..+pi)
rollspeed	float	rad/s	Roll angular speed
pitchspeed	float	rad/s	Pitch angular speed
yawspeed	float	rad/s	Yaw angular speed

Figure 23: Attitude MAVLink message.

HEARTBEAT (#0)

[Message] The heartbeat message shows that a system or component is present and responding. The type and autopilot fields (along with the message component id), allow the receiving system to treat further messages from this system appropriately (e.g. by laying out the user interface based on the autopilot). This microservice is documented at <https://mavlink.io/en/services/heartbeat.html>

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

Figure 24: Heartbeat MAVLink message.

4.2.10 Integration with DJI SDK

Our GCS supports integration with a DJI autopilot, such as the Matrice 210 from scenario 2 and the Matrice 600 from scenario 3, via a module on the onboard computer that converts MAVLink messages into messages understood by DJI’s Onboard Software Developing Kit for ROS package, OSDK-ROS for short. The package provides a ROS interface for OSDK and supports full control over platforms like DJI M100, M600, M210, or drones equipped with A3/N3 flight controllers, using standard ROS messages and services. More information can be found on the ROS wiki for DJI’s OSDK, http://wiki.ros.org/dji_sdk/. The module loaded into the on-board computer maintains the connection with our GCS sending the published telemetry to it in MAVLink messages, while also forwarding the commands received from the GCS to the autopilot through ROS. By subscribing through ROS to the topics on the on-board computer where the autopilot publishes its telemetry and even the camera publishes frames, we can integrate the whole system with the SAS. After obtaining the data from the UAV it can be sent to the SAS as explained on Section 5.3.

5 Communications interface with the SAS

5.1 Description of the interfaces



Figure 25: Data flow diagram.

In all three scenarios there are three interfaces to get or request data to and from the SAS:

- The incoming missions are downloaded via REST API from the SAS and processed by the GCS on the ground computer, from where they can be loaded into the UAV by the pilot.
- The video stream from the cameras on board is sent to a video receiver connected to a framegrabber or a video card and from there to the ground computer. The video is then sent to the SAS via RTSP where it can be consumed while its streaming.
- The UAV publishes to the SAS its telemetry, any sensory output measured as well as requested snapshots synchronized with the telemetry at the moment of its capture. Its packed in a JSON scheme and sent via MQTT protocol.

Sensory output, telemetry and synchronized frames with telemetry are processed by the onboard computer and sent through a MQTT client to a midway broker that then forwards the queues to the SAS's own MQTT broker (see Figure 20). The communication

between the onboard computer and the SAS is done through a radio data link connected to a modem on ground, thus allowing the client to see the SAS while on flight.

A midway broker is placed on the onboard computer so that in case the communication channel with the SAS is lost momentarily all messages are still sent, since it enqueues the received messages that have not been forwarded, while keeping the latest one on top. The communications channel robustness cannot be guaranteed during flying operations with UAVs, hence the need for this broker.

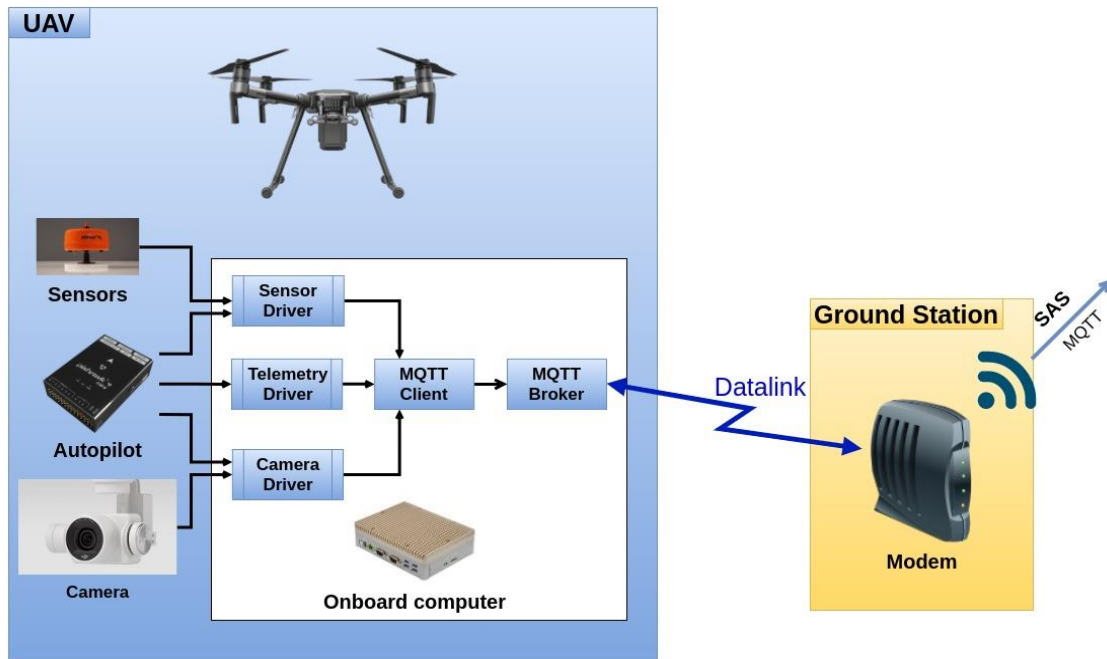


Figure 26: Sending data to the SAS.

5.2 Downloading the mission

Communications with the SAS are essential in order to obtain mission information. The operator of the UAV will receive from the SAS a list of available missions, selecting one through REST API protocol. The mission is downloaded into the ground computer and parsed in the GCS, where the operator can upload it into the UAV through the datalink. A diagram showing the flow of data when downloading a mission is represented in Figure 21.

The mission is received from the SAS as a JSON with a list of coordinates and actions, a sample of which can be seen in Figure 29. This list is parsed into a mission in the GCS, representing it in the map and through the datalink is uploaded to the autopilot. The pilot can then select to start, stop, or edit the mission and it can be safely executed.



Figure 27: Downloading a mission from the SAS.

The mission received is parsed and then represented into the GCS, where the pilot can review it and edit, if needed, any of the waypoints. The mission is then sent to the onboard computer and to the autopilot. The onboard computer must configure the mission name and context fields on the messages that are sent to the SAS by the sensors on the UAV, beginning the measure and transmission when the mission starts. On Figure 22 we have a mission updated into the GCS after being parsed.

The mission received includes a list of ordered waypoints as geographical coordinates and a list of actions. The list of actions must include on each action the index of the waypoint where the action must be done. The actions include common UAV commands as well as request for a specific sensor’s measurement or a camera’s snapshot synchronized with telemetry. The possible actions than can be requested are:

- Take-Off / Land
- Go to waypoint
- Loiter over point
- Return to launch
- Sensor measurement Start / Stop
- Camera frame request

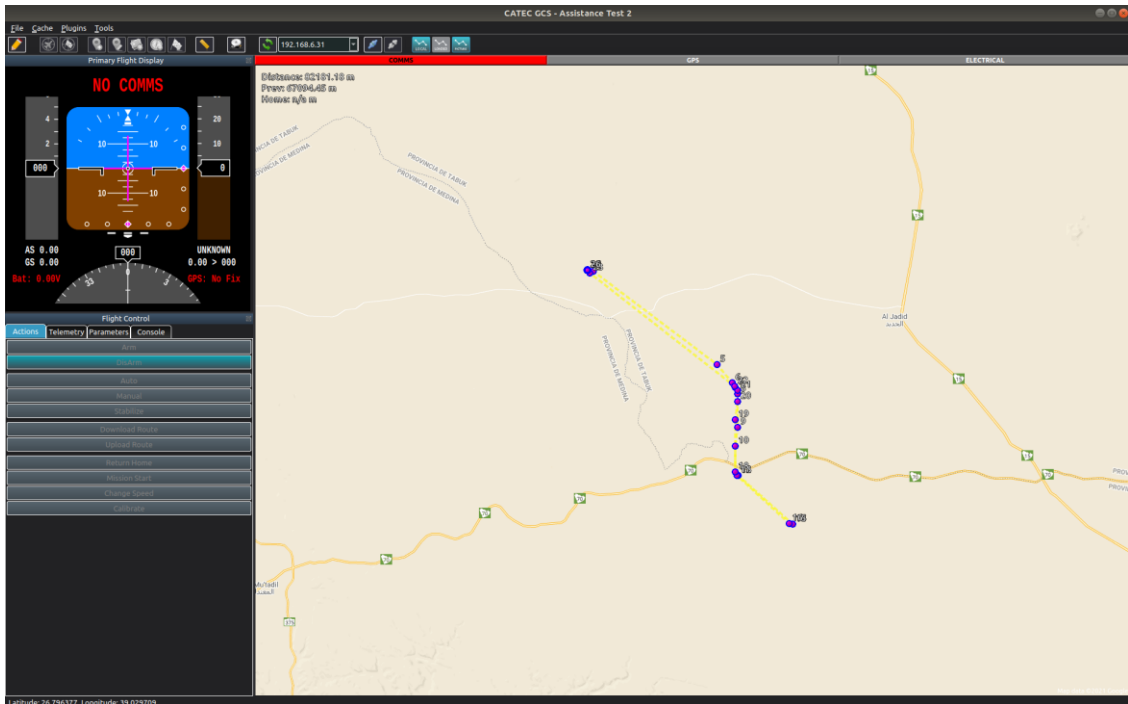


Figure 28: GCS application with a loaded mission.

```

{
  "id":"846d8595-aa1b-4283-af92-e62963e3a6e3--105-15-11-105015",
  "description":"Test 2",
  "context":"Assistance Test 2",
  "resource":"105-15.11.105015",
  "actions":[
    {"sensor":"10", "action":"START", "position":13},
    {"sensor":"10", "action":"STOP", "position":13}],
  "geometry":{"type":"MultiPoint", "coordinates":[
    [38.40625,27.029235,0],
    [38.40625,27.029235,400],
    [38.40914427925721,27.02771840870259,400],
    [38.41301310336403,27.027718407661258,400],
    [38.55772253210502,26.93026968386475,400],
    [38.57591578755577,26.911477484591618,400],
    [38.57894799679757,26.90665406527043,400],
    [38.581980206957105,26.899887034159207,400],
    [38.58195559474063,26.8650239337872,400],
    [38.57894799598244,26.845655544040287,400],
    [38.57894799679757,26.81853979735328,400],
    [38.581980206958775,26.814666119270665,400],
    [38.58280004858558,26.814666120445228,400],
    [38.64676713017165,26.76371854905318,400],
    [38.64676713017165,26.76371854905318,400],
    [38.642309198115996,26.76430830710326,400],
    [38.58150726717599,26.814666120445228,400],
    [38.57894799591683,26.818539799519765,400],
    [38.57897260809629,26.872771289685964,400],
    [38.58198020685448,26.892139679432876,400],
    [38.581980206039354,26.903760714322356,400],
    [38.578631084642986,26.908039249710537,400],
    [38.40914427925721,27.025730989253862,400],
    [38.40625,27.029235,400],[38.40625,27.029235,0],
    [38.40625,27.029235,0]]],
  "_updated":"2020-12-04T14:10:25.584Z",
  "_expired":false
}

```

Figure 29: Sample mission JSON downloaded from SAS.

5.3 Telemetry data publisher

The UAV system's telemetry data defines the latest situation of the aircraft reported by the sensors mounted on the aircraft, and it is composed by:

- Aircraft's attitude, usually given in Euler angles; roll, pitch, yaw, or in quaternions. The aircraft uses a NED-system (North-East-Down) as an external reference, the world frame.
- Aircraft's speed and acceleration vectors referenced to the NED world frame.

- Global position, using GPS standard, given as a latitude and longitude from the WGS84 ellipsoid and altitude over mean sea level, as well as the timestamp at measuring moment.
- Remaining autonomy reported as the remaining flight time.

When the pilot selects a mission on the GCS, and loads it into the UAV, telemetry data starts being sent to the SAS. Some of the telemetry parameters are set by the GCS at the time of loading a mission and are stored on the on-board computer throughout the duration of the mission, the resource's ID, the mission name and the context of that mission. Telemetry data is then reported by the UAV's autopilot to its on-board computer at a given rate. The computer gathers this information as well as the stored parameters and other relevant information such as the aircraft's number plate and some operation limits and packs it together on a JSON string to be published to the SAS, so as any other module may identify and consult the latest telemetry.

The on-board computer publishes the latest telemetry data available so all other modules may consult its current situation by checking the resource's topic on the SAS. The UAV's telemetry data is received from the autopilot, processed by the on-board computer and published to the SAS via MQTT protocol, as shown in Figure 24. The on-board computer packs the data on the JSON schema defined by the Data Model and sends it through a MQTT client to a MQTT midway server that will forward the messages to the SAS when connection is available. Connection to the SAS is done through the radio datalink connected on the ground to a modem from where the SAS can be accessed. This way, even if the drone loses connection with the server, it is ensured that all telemetry packets will still be received, albeit with some delay, maintaining the latest message on the top of that queue. This is done so that any applications wanting to view the route of the drone can download the full flight telemetry from the SAS, even if connection is lost at any point.

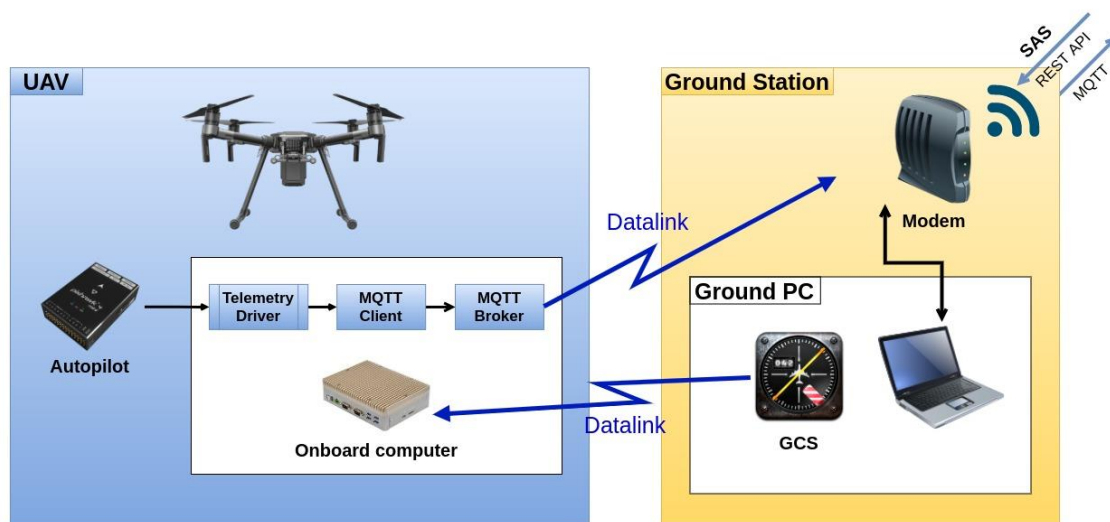


Figure 30: Configuring onboard computer and sending telemetry.

```
{
  "_id": "catec_uav",
  "attitude": {
    "pitch": -0.5,
    "roll": 0.5,
    "yaw": 1
  },
  "context": "Test Context",
  "geometry": {
    "coordinates": [
      36.99639892578125,
      -6
    ],
    "type": "Point"
  },
  "height": 50,
  "mission": "Test Mission",
  "operationSpeed": 120,
  "plate": "UAV1234",
  "remainingAutonomy": 0,
  "sensors": ["catec_sensor"],
  "speed": [ 1, 0, 0],
  "subtype": "uav",
  "timestamp": "2020-12-18T14:10:08.884Z",
  "type": "unmannedVehicle",
  "workingAltitudeMaximum": 120,
  "workingAltitudeMinimum": 0
}
```

Figure 31: Sample telemetry data JSON sent to SAS.

5.4 Sensor data publisher

5.4.1 Gas sensors

The gas sensors mounted on the drone, as well as some other optionally replaceable sensors, continuously measure the CO and CO₂ concentration in the atmosphere around the drone. Sensor information is processed by the onboard computer and packed into the expected JSON structure, alongside the drone's telemetry, its position, and its attitude, at the measuring time. This message is then sent through a MQTT client to a midway broker and forwarded to the SAS, as shown in Figure 20.

If the sensor measures more than a single variable, only one message is sent, described as an array with all measures. For the gas sensor, the measured CO, CO₂, NO₂... may be published in a single message, thus minimizing the necessary packets to be sent. A sample JSON structure is shown in Figure 32. Both the mission and context fields are filled with the current mission downloaded from the SAS. The measure field is an array of measures of the sensor, so if a sensor were to also give us any other gas readings, these would be placed following the CO₂.

```
{
  "_id": "catec_sensor",
  "context": "Test Context",
  "geometry": {
    "coordinates": [
      36.95880126953125,
      -6
    ],
    "type": "Point"
  },
  "measurement": {
    "confidence": 1,
    "type": "CO2",
    "unit": "ppm",
    "value": 568
  },
  "mission": "Test Mission",
  "timestamp": "2020-12-18T09:04:23.451Z",
  "type": "CO2 sensor"
}
```

Figure 32: Sample gas sensor data JSON sent to SAS.

5.4.2 Camera frames

It is possible to request the mounted visual cameras to take snapshots at a given time. These frames must be sent to the SAS with the UAV's telemetry at the moment of capture, both the GPS position and timestamp as well as the attitude of the camera. Similarly to information from the gas sensors, the frame is encoded into a base 64 string and sent to the SAS through a MQTT client and midway broker packed into a JSON structure. This JSON contains both the image encoded into a string as well as the associated telemetry. A sample JSON is shown in Figure 33. The codified image string is placed in the value field, simplified in the sample to fit since the resulting string is quite long.

To maintain proper time coherence the UAV's telemetry must be stored when capturing the frame, since the codification may incur in a slight delay which would be noticeable as the platform moves. Images are compressed in a different thread ensuring the system's running time and stability are not compromised and get sent to the SAS when available. The reason for this encoding is that the JSON format needed to transfer data to and from the SAS does not natively support binary data, hence encoding the data into a base 64 string becomes necessary.

```
{
  "_id": "catec_camera_id",
  "context": "Test Context",
  "geometry": {
    "coordinates": [
      35.77040100097656,
      -6
    ],
    "type": "Point"
  },
  "measurement": {
    "confidence": 1,
    "type": "Imagen visual codificada en base64",
    "unit": "",
    "value": "/9j/4AAQSkZJRgABAQAAQABAAD/2wBD ..."
  },
  "mission": "Test Mission",
  "timestamp": "2021-01-13T10:14:27.382Z",
  "type": "visibleCamera"
}
```

Figure 33: Sample camera frame sent to SAS.

5.5 Streaming interface

In all scenarios the videos from the cameras mounted on the UAV and the first-person view cameras from the drone itself are streamed into the SAS so it may be consumed or stored. The protocol used to transmit the video is (RTSP). This protocol is used for media streaming over a network, allowing real-time control of the visualized video at the endpoint. This allows the client and the server to issue commands to play/stop or record the stream, thus making it possible for the SAS to store the received videos or reproduce them for real-time visualization of the UAV's camera feed.

The video captured by the cameras is emitted via a wireless transmitter to either the pilot's RC controller in scenario 1 and 2 or a video receiver and monitor for visualizing it in scenario 3, as described in deliverable D4.1. Both the monitor or the visualizing device attached to the pilot's RC have either a capture card or a framegrabber attached, allowing the transmission of the stream to the ground computer, from where it is sent via RTSP to the SAS for consuming remotely. A diagram of the video emission to the SAS is represented on Figure 34. GStreamer framework is used to compress video stream in H.265 standard and then encode the payload into RTSP packets.

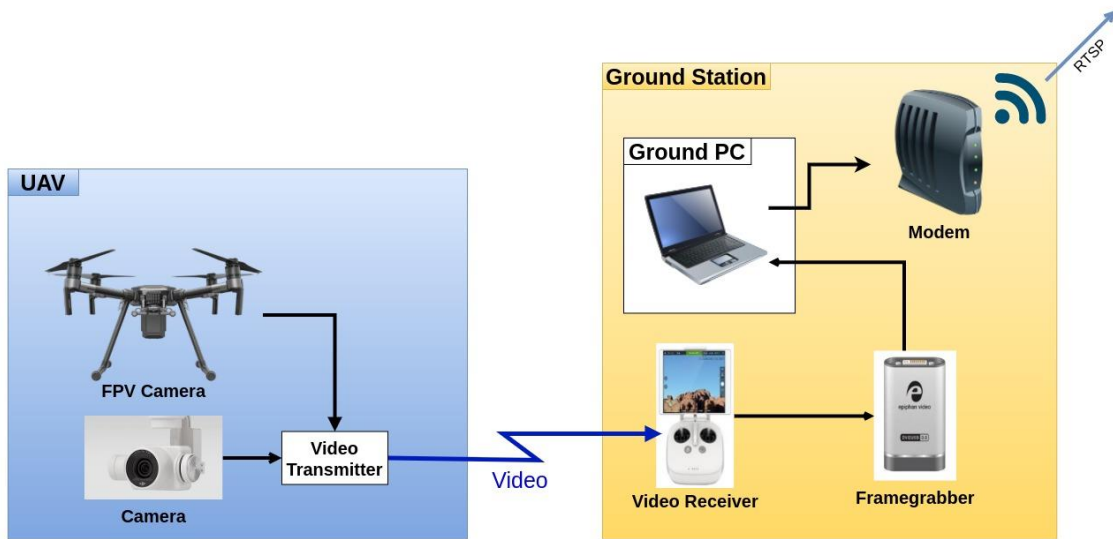


Figure 34: Streaming video to the SAS.

6 Conclusions

This document shows the integration of the drones with the rest of the ASSISTANCE project system, through the SAS module. The first part of the document addresses the general architecture and the scenarios to put the integration into context. The GCSs from which the operators will interact with the drones are defined below. Finally, in the last part of the document, the integration of the different components is detailed: flight data telemetry, video, video frames, sensor data, etc., with the SAS, also showing some results of integration tests that have been made during the development of it.

In this way, the drones in their different modalities that depend on the scenarios are already integrated with the rest of the system, a fundamental task to carry out the experiments in the last months of the project. This task is essential for drones and the data provided by them to reach the corresponding modules and be useful for the first responders.

The next steps in this field will consider the execution of this integration in experimental flight tests. For this purpose, test flights will be carried out where real telemetry of the drone, the gas sensor and the camera will be taken to test said integration with real data. It will be debugged in the laboratory but under the same conditions and with the same software that will be used in the field tests, so it is transparent to the environment of execution of said tests.