Multi-drone Control with Autonomous Mission Support

Nuno Paula*, Bruno Areias*, André Braga Reis*, Susana Sargento*
*Instituto de Telecomunicações, Universidade de Aveiro, 3810-193 Aveiro, Portugal

Abstract—**Recent advancements on miniaturization and cost of sensors and instruments have promoted a growth in the usage of drones in an increasingly wide range of scenarios such as search and rescue, agriculture and environmental monitoring. However, most mechanisms for drone control still require an active pilot, limiting the ability to execute complex missions, especially when multiple drones are involved. Leveraging recent advances in the autonomous capabilities of commercially-available drone equipment, we introduce a complete and modular solution for controlling multiple drones, implementing the functionality necessary for inexperienced users to plan, execute and monitor complex missions that require drone cooperation.**

Index Terms—**drones, remote control, autonomous flight, multi-drone, collaborative missions**

I. Introduction

Recent research efforts, miniaturization improvements and cost reductions have since promoted the usage of Unmanned Aerial Vehicles, or simply drones, in an increasingly wide range of scenarios such as search and rescue, building inspection, environmental monitoring and agriculture [1].

Most commercially-available drones still depend on manual remote control from a limited distance, requiring a constant aware pilot dedicated to each drone, and thus narrowing the ability of executing complex or repetitive missions, particularly when they require the participation of more than one drone. Basic navigation functionality, however, is increasingly being supported by major flight controller manufacturers. This, along with the increasing availability of Application Programming Interfaces (APIs) and Software Development Kits (SDKs) offered by such manufacturers, enables the integration of drones in platforms that abstract and simplify their control.

In order to overcome the current limitation and lack of abstraction in the control of one or more drones, this paper proposes a complete and modular control solution which enables an inexperienced user to plan, execute and monitor both simple and complex missions which may involve one or more participating drones, while also implementing the functionality required for collaborative scenarios.

We seek to combine the advantages of a single Unmanned Aerial Vehicle (UAV) with the advantages of a swarm, as proposed in [2], including the possibility of continuously executing a mission even when an unexpected event causes one drone to land or cancel its execution. Here, the potential for increased mission flexibility is also suggested, in which a set of drones dynamically adapts to a mission, for example, to increase the area coverage capacity. To achieve a complete drone control solution, a set of objectives must be accomplished:

- Build a drone system which can easily be integrated in a drone control platform.
- Abstract drone control through the use of high-level commands.
- Set up the communication mechanisms required for monitoring drone status and delivering control commands.
- Develop mechanisms for mission execution which are not reliant on user interaction.
- Create collaboration methods which enable more than one drone to participate in a mission, either simultaneously or in drone replacement scenarios.
- Provide a set of tools and applications which enable the monitoring and control of connected drones through a user-friendly graphical interface or a Representational State Transfer (REST) API.

The remainder of this paper is organized as follows. Section II describes actual advancements regarding platforms for drone control. Section III contains a description of the proposed architecture and a brief enumeration of its scenarios. Section IV presents three different experiments that evaluate the performance and functionality of the platform, along with a description of the used drone system. Finally, section V provides general conclusions and suggestions for future work.

II. Related Work

By analyzing the recent research efforts towards drone control, we can observe that an increasing amount of solutions are becoming available. The ROLFER project [3] proposes a drone control solution which enables swimmers in distress to call a drone by tapping a button on a smart watch. Its applications are limited to rescue support scenarios, providing no multi-drone support and no mission planning capabilities.

A system supporting complex mission planning and execution is proposed in [4]. This project provides a platform for infrastructure inspection using multi-rotor drones. It enables a user to define complex missions which can be autonomously executed, but provides no support for multi-drone interaction and has a limited communication range using Wi-Fi.

Recent research in [5] resulted in the development of a platform which enables persistent mobile aerial surveillance, using intelligent battery health management and drone swapping to maximize the time during which a zone may be covered without interruption. This mechanism enables a set of four quadcopters to take turns hovering above a location, achieving a total flight time of 54 minutes. Its communication range, however, is limited to half a mile.

The work in [6] proposes an architecture for controlling a drone remotely. Through the usage of multiple cellular carriers, the authors maintain drone operation even outside the communication area of one cellular carrier. No support for multi-drone interaction or collaborative missions is provided.

These solutions are often highly specialized in unique tasks, and lack support for generic drone control, mission planning and multi-drone control. The proposed platform in this work features, simultaneously, long communication range, real-time high-level control, complex mission planning, multi-drone support and multi-drone task collaboration.

III. Architecture

When aiming for a platform which supports the monitoring and control of an arbitrary number of drones in a decoupled and abstracted manner, a series of sample scenarios should be taken into consideration, such as the acquisition of both **internal telemetry** and **readings from externally connected sensors**, geographic **coordinate based drone control**, **mission execution**, drone **collaboration** and event **logging**. These are scenarios that, when fully supported by the platform, will show its ability to execute relatively complex tasks, which require the combined use of one or more components designed for such scenarios. These components are divided in two major groups, drone side and ground side.

A. Overall Architecture

The overall architecture, with both drone and ground sides, is presented in Figure 1, which features multiple drone systems that are connected to the ground side through broker message relaying.

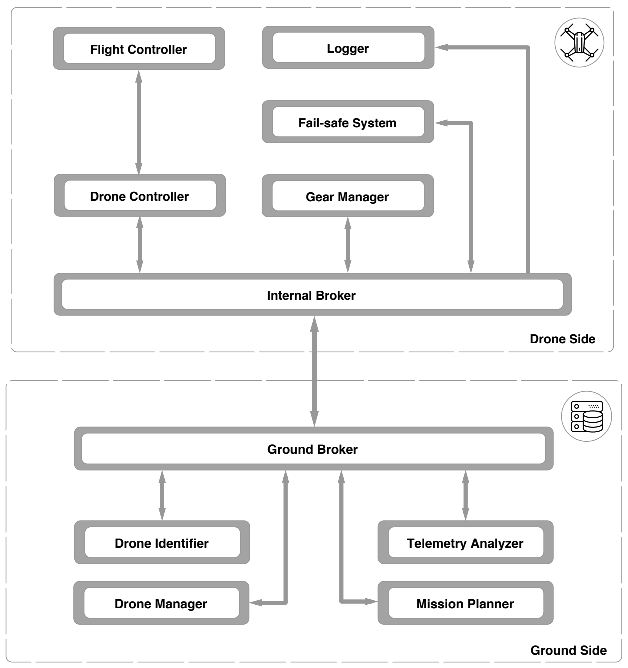

Figure 1: Overall platform architecture

B. Drone Side

1) Flight Controller: The **Flight Controller** holds a crucial place in the drone side architecture of the platform and is designed to allow for internal telemetry information to be easily exported, providing architecture components which are external to the flight controller with a means of obtaining this information. Sensors such as a Global Positioning System (GPS) receiver, external magnetometers or even battery voltage and current sensors are typically supported out-of-the-box by flight controllers and – since they can provide important information for flight operations – can be directly connected to them using commonly accepted communication interfaces.

In order to implement high-level control mechanisms, the **Flight Controller** will also receive control commands that may either be relative to the actual position of the drone, based on geographic coordinates or even arming switch commands, which originate from the **Drone Controller**. The **Flight Controller** also puts the burden of receiving and processing flight data on the **Drone Controller**.

2) Drone Controller: This component keeps a permanent connection to the **Flight Controller** and is responsible for making drone control functionality available in a high-level basis. It is composed of several submodules which enable the functionality requested in the described scenarios.

- **Flight Controller Interface**: since **Flight Controllers** typically rely on hardware implementations using diversified or even proprietary communication protocols, the **Drone Controller** must feature a layer which enables the abstraction of communication details that are specific to these implementations. This task is accomplished by the **Flight Controller Interface**, which is responsible for keeping direct, bi-directional communication with the **Flight Controller**, handling incoming flight data and crafting control commands.
- **Navigation Processor**: it is responsible for processing the desired high-level control actions into commands that can be understood by the **Flight Controller**.
- **Mission Worker**: it is responsible for parsing received mission requests, storing each of the mission steps in memory and sequentially feed each of them to the **Flight Controller**, while also keeping track of the progress of the current mission.

The **Drone Controller** will also periodically transmit heartbeats in order to announce its availability to the **Drone Identifier**, a ground-side module.

3) Gear Manager: The acquisition of external sensor data requires interfacing with devices that provide no compatibility with the flight controller. These include sensors such as a CO_2 module or a camera. The **Gear Manager** tracks the physical details of the drone (type, frame size, battery capacity) and external sensors, and publishes sensor readings which may allow other modules to trigger events or for logging purposes.

4) Fail-safe System: The **Fail-safe System** is a component that continuously analyzes drone telemetry with the goal of detecting behavior anomalies or sensor readings that indicate dangerous situations, such as a low remaining battery capacity. This gives in-flight drones the ability to automatically request a replacement or, upon critical situations such as an imminent crash due to a physical failure, to deploy fail-safe mechanisms such as a parachute to slow down the descent of the drone, or a loud audible alarm to alert nearby people.

5) Logger: The **Logger** keeps a connection to the internal broker and records every telemetry reading, control command or component interaction.

C. Ground Side

This subsection describes the platform architecture on the ground side, shown earlier in Figure 1.

1) Drone Identifier: This component keeps track of which drones are connected to the platform, by listening for heartbeat messages, which originate from each connected drone. These messages contain a unique drone identifier, current geographic coordinates and a timestamp of when a drone was last seen.

2) Drone Manager: The **Drone Manager** is a component which serves a REST endpoint for high-level drone control. The existence of this endpoint allows for a high degree of extensibility, since it enables any device with Internet access to achieve drone control through text-based commands. It also features a simple web application which provides the user with a graphical interface for interacting with the platform.

3) Telemetry Analyzer: Telemetry acquisition scenarios require architecture components which are capable of storing telemetry data. The storage of this data enables a user to have access to flight details, which may also be important for debugging purposes or in case of a crash. This component is backed by a database to serve as storage for the time-series data. Ideally, this storage backend should be optimized for time-series data. For this purpose, InfluxDB [7] is used. A web dashboard is bundled with this module, to provide the user with a graphical means of viewing the obtained drone telemetry.

4) Mission Planner: Mission requests are normally generated on the ground side and are planned by a user. Replacement and collaboration requests are exceptions to this behavior, since these happen with no user interaction. The main purpose of the **Mission Planner** is the generation of mission requests that can be effectively parsed and executed by the drone-side **Mission Worker**. The **Mission Planner** interacts with the **Drone Identifier** to obtain the list of drones which are connected to the platform, and is also in charge of keeping track of the mission progress of each drone. Knowing which drones are connected at a given time proves useful when a drone needs to be replaced or a collaborative mission should occur. This module also serves a web application which enables users to graphically prepare and send the chain of commands.

IV. Experimental Verification

To validate the proposed architecture, we plan and execute three distinct experiments, ranging from simple tests to the baseline capabilities of the platform to more sophisticated tasks which require the automation of full paths and multiple drone collaborative missions.

A. Drone System Setup

To implement a multi-rotor that can be integrated in the platform, the following physical components were used:

- **Flight Controller** - OpenPilot Revolution
- **Frame** - DJI Flamewheel F550 (hexacopter)
- **Electronic Speed Controllers** - DJI 420S (6)
- **Motors** - DJI 2313 (6)
- **GPS** - u-blox NEO-M8N
- **Battery** - Four-cell 5500mAh LiPo

Each drone contains a Raspberry Pi 2 in which all drone side modules are deployed. A physical Universal Serial Bus (USB) connection is kept at all times between the Raspberry Pi and the **Flight Controller**. Internet connection is provided to the Raspberry Pi through a USB 3G modem. The resulting drone system is shown in Figure 2.

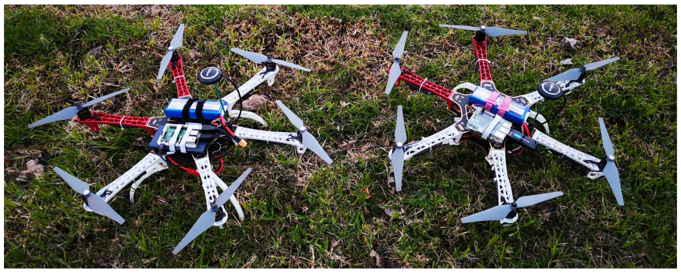
Figure 2: Two drones in a disarmed state.

B. Experiment 1: Panic Button

1) Objectives: A panic button allows testing the platform for the correct implementation of basic navigation capabilities. In this scenario, a user carries a device with geolocation capabilities which presents a simple button that, when pressed, calls an available drone to travel to the location of the user and hover above him at a fixed altitude. If successful, the experiment shows that the drone-side control components, server-side mission assignment components and communication mechanisms as working as expected.

2) Method: This experiment makes use of a simple iOS application with a panic button that, when pressed, sends a panic request with the GPS coordinates of the user.

3) Evaluation Procedure: For this experiment, the drone is placed approximately 100 meters away from the user. The drone starts at the ground level, in a disarmed state. At this point, the user makes use of the developed application to initiate the experiment. A drone is first placed in the armed state, and must then climb to an altitude of 10 meters and

Stage	Duration (s)
Arming	3.008
Vertical flight	5.005
Horizontal flight	16.539
Total	24.552

Table I: Duration of each stage of the panic button experiment.

initiate an entirely horizontal flight towards the position of the calling user. Timing metrics for message reception and processing, arming, vertical and horizontal flight, as well as the total time elapsed since the initial request up to its completion, are obtained at the end.

4) Results: After executing this experiment, the **Mission Planner** produces a mission log file with the path followed by the drone. This file was imported into the viewing front-end of the **Mission Planner**, showing the path displayed in Figure 3.

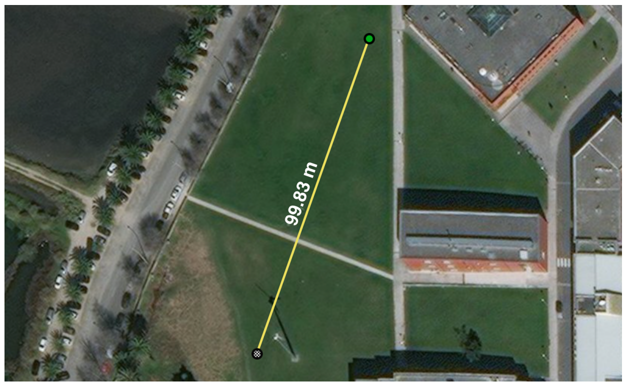

Figure 3: Flight path on the panic button experiment.

By observing the path, it is possible to conclude that the drone followed the expected behavior, traveling along a horizontal line of approximately 100 meters. For each stage, timing metrics were acquired and are shown in Table I.

The arming time is a configurable value at the **Flight Controller** level and defaults to 3 seconds, and so the time spent in the first stage represents an expected value. As a safety measure, the maximum climb rate is limited to $2m/s$. Since the drone must climb 10 meters during the vertical flight stage, this stage should never complete in less than 5 seconds. However, flight speed during vertical or horizontal flights may be affected by external factors such as gusts of wind. The duration of stages 2 and 3 show that the drone executed its vertical flight with a speed of approximately $1.99m/s$ and its horizontal flight with a speed of approximately $6.04m/s$, which both match the expected values. Finally, the obtained timing metrics for each stage show that the platform is able to place an initially disarmed drone above a user which is 100 meters away in under 25 seconds, even when including the negligible delay of 3.5 ms required to decode and process the steps of the mission. Additional timing metrics are shown in Figure 4 and include the average round-trip time, measured between the drone and the user device which initiates the mission, along with the average time required for the drone to communicate its progress to the **Mission Planner**. These values are strongly correlated to the delay of the cellular link of the drone.

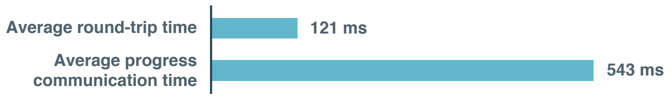

Figure 4: Average network round-trip time between the drone and the user device and average time taken for the drone to communicate its progress to the ground.

C. Experiment 2: Drone Self-replacement

1) Objectives: This experiment tests if the platform is able to quickly replace an in-flight drone and have a second drone resume its execution, showing a basic collaboration functionality of the platform. Here, a single drone participates in a mission, during which self-replacement is triggered. When this happens, the remaining steps of the mission are assigned automatically to a second drone, which executes them sequentially until reaching the end of the mission.

2) Method: The **Mission Planner** front-end is used to generate a sequence of waypoints that cover a quadrilateral area. For delimiting the desired area, the user must select three vertexes of the area by clicking on a map element. Upon pressing the Map button, a distribution of waypoints is automatically calculated for the user.

Figure 5 displays the path that is sent to the drone, along with the waypoint that will trigger self-replacement.

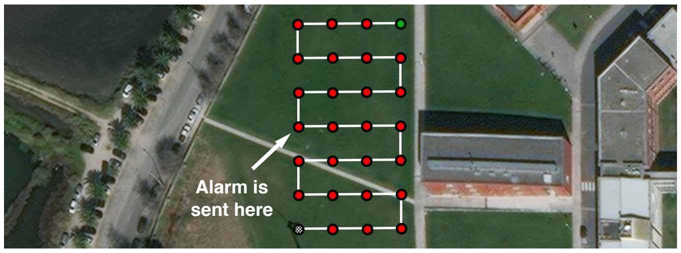

Figure 5: Path planned for the self-replacement experiment.

3) Evaluation Procedure: Before executing the experiment, the mission path is selected using the front-end of the **Mission Planner**. A total of 28 waypoints are generated. The drone which is initially assigned to execute the mission is already hovering nearby, awaiting orders from the user, while the drone which will be used to replace the first drone is located at ground level, in a disarmed state. Upon initiating the mission, the first drone is expected to travel to the first waypoint of the delimited area, successively flying towards the following waypoints until it reaches the waypoint in which the alarm is triggered. When this alarm is triggered, the **Mission Planner** should instruct the in-flight drone to cancel its current mission and perform a landing. Simultaneously, the **Mission Planner** assigns a new mission to the second drone, which contains only the remaining steps of the mission that was being

t (s)	**Drone**	**Stage**	**Duration** (s)
0	A	Complete waypoint 1	8.906
8.91	A	Complete waypoints 2-13	81.453
90.36	A	Replacement alarm	-
90.36	A	Landing	21.624
90.36	B	Preparation	10.336
100.69	B	Complete waypoint 14	11.029
111.72	B	Complete waypoints 15-28	101.647
-	-	Total	213.37

Table II: Duration of each stage of the self-replacement.

executed by the first drone. Preparation steps which include arming and vertical ascension are required before resuming the mission of the first drone. The viewing interface of the **Mission Planner** can be used to evaluate the path executed by both drones. Timing metrics are acquired for both mission execution and mission handover.

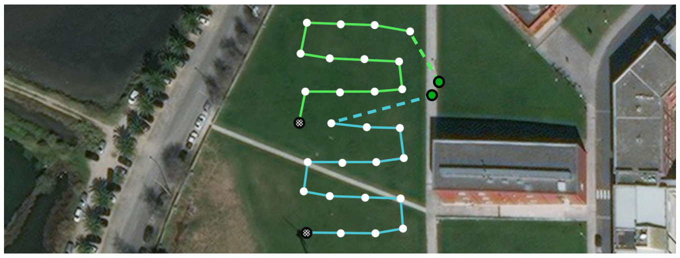

Figure 6: Path followed by both drones during the execution of the self-replacement experiment.

4) Results: Figure 6 shows the path of both drones: green line of the first drone (A), blue line of the second drone (B), in a successful collaboration. The first drone achieved 13 mission waypoints, while the second drone completed 15, totalling the 28 original waypoints. Table II contains the time required for each stage of this experiment, including the waypoint flight and landing of the first drone, along with the preparation (arming and vertical flight) and waypoint flight of the second drone. Since both the first and the second drones are, respectively, 17 meters and 40 meters away from their first waypoint when executing their part of the mission, its execution time is expected to be greater than the remaining waypoints. Figure 7 shows the difference between the time taken to execute the first waypoint of each drone and the average time required to execute a waypoint.

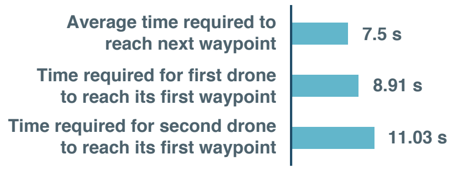

Figure 7: Time required for the execution of the first waypoint of each drone in comparison to the average time required for the execution of waypoints in the self-replacement experiment.

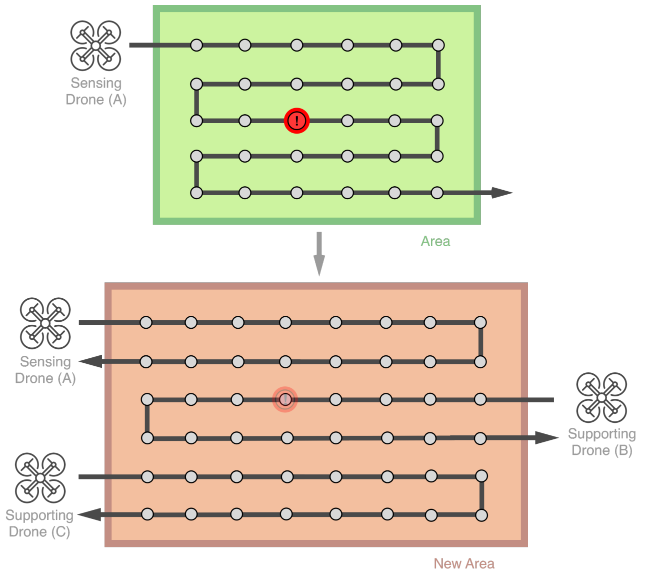

Figure 8: Collaborative sensing scenario with three drones after automatic area reconstruction, showing the location where the alarm occurs, which is the center of the new area.

D. *Experiment 3: Collaborative Sensing*

1) Objectives: This experiment involves the collaboration of multiple drones that are simultaneously connected to the platform. The goal is to have more than one drone working on the same task at a given time while each of them provides relevant progress to the mission.

2) Method: As an example of the specified interaction, a collaborative sensing task is proposed. In this scenario, a drone is actively acquiring data from a set of environmental sensors carried aboard. In order to accomplish this task, the **Mission Planner** calculates a series of evenly distributed waypoints inside the area of interest for drones to travel along, obtaining data from the connected sensors upon reaching each waypoint.

Because the flight time of a drone is a limited resource, this task makes use of multiple drones to cover wider areas. A single initial drone fires an alarm when a predefined sensor threshold is triggered, issuing a request for collaboration. As shown in Figure 8, upon receiving this request, the **Mission Planner** automatically selects available drones currently connected to the platform, prioritizing those with high remaining battery. A new wider area centered on the location in which the alarm was triggered is then calculated along with its set of interior waypoints, which are evenly distributed among the participating drones, initiating a new sensing task.

3) Evaluation Procedure: In the proposed area, the waypoints are distributed 7 meters apart from each other. This way, the entire area can be covered by a total of 30 waypoints, as shown in the green section of Figure 8. The request for collaboration is launched by issuing a sensor reading alarm once the initial drone reaches approximately half of the originally assigned waypoints. This simulates the occurrence of an abnormal environmental reading. During this experiment, three drones are connected to the platform, all of them on the ground and disarmed. The mission, including the command to simulate abnormal sensor readings, is prepared and executed using the **Mission Planner** front-end.

4) Results: The path followed by the first drone before a collaboration request has occurred, along with the paths taken by all drones when simultaneously collaborating in the mission can be seen in Figure 9.

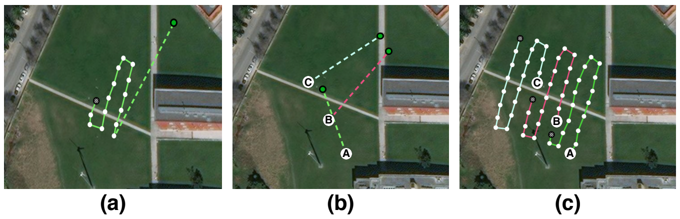

Figure 9: Path followed by all drones during the execution of the collaborative sensing experiment. In step (a), drone A executes the initially planned mission until the alarm waypoint is reached. Next, in step (b), a new mission is generated and drones A, B and C simultaneously reach the first waypoint of their part of the mission. Finally, in step (c), all drones collaboratively execute their parts of the mission.

As can be seen in the step c) of Figure 9, the mission now includes 56 waypoints instead of 30, covering an area of $2058m^2$, which corresponds to an increase of 110%. This area increase can be pre-set in the **Mission Planner**. It is possible to observe that the new mission was successfully and collaboratively executed by the three drones, since every generated waypoint was achieved. Table III contains the time required for each stage of this experiment. The collaboration alarm was launched by drone A once it reached waypoint 15 of the initial mission, after approximately 2 minutes had passed, and initiated the arming, take-off and vertical flight of drones B and C. Since drone A was already in flight, no preparation stage was required when the alarm was launched, and thus it proceeded to execute the first waypoint of its part of the mission, waypoint 1. Drone B proceeded to execute waypoint 19, while drone C proceeded to execute waypoint 37. After approximately 2 minutes and 34 seconds had passed since the mission expansion, all drones successfully had completed their parts of the mission.

t (s)	**Drone**	**Stage**	**Duration** (s)
0	A	Preparation	10.185
10.19	A	Complete waypoint 1	12.08
22.27	A	Complete waypoints 2-15	97.517
119.78	A	Cooperation alarm	-
119.78	A	Complete waypoint 1 (new area)	10.069
119.78	B	Preparation	13.223
119.78	C	Preparation	12.912
129.85	A	Complete waypoints 2-18	128.727
132.7	C	Complete waypoint 37	9.162
133	B	Complete waypoint 19	15.054
141.86	C	Complete waypoints 38-56	131.449
148.05	B	Complete waypoints 20-36	120.243
-	-	Total	273.31

Table III: Duration of each stage of collaborative sensing.

V. Conclusion

This paper proposed a new modular solution for autonomous control of multiple aerial drones. This platform enables control details to be completely abstracted, allowing an inexperienced user to plan, execute and monitor complex missions with one or more participating drones, that can also collaborate in the execution of these missions. Through a reference set of real-life experiments, the features of the platform were thoroughly verified and profiled. The modular nature of the developed platform allows its expandability for new scenarios, for increased drone and flight controller compatibility and for its scalability. The resulting platform is ready to support drone monitoring and autonomous drone flight in tasks ranging from basic navigation commands to the automation of complex paths and collaboration of multiple drone systems. As future work, we aim to support drone-to-drone communication by implementing communication mechanisms which do not constantly rely on a connection to the ground. Mission planning and execution can also be improved to support the creation of conditional steps and fallbacks. Finally, devices such as parachutes or sirens could be added to drones, which would be triggered upon automatic detection of anomalous behavior.

VI. Acknowledgements

This work is funded by FCT/MEC through national funds under the project, S2MovingCity CMUP-ERI/TIC/0010/2014, and by the European Regional Development Fund (FEDER), through the Regional Operational Programme of Lisbon (POR LISBOA 2020) and the Competitiveness and Internationalization Operational Programme (COMPETE 2020) of the Portugal 2020 framework Project 5G with Nr. 024539 (POCI-01-0247-FEDER-024539).

References

[1] N. Mohd Noor, A. Abdullah, and M. Hashim, "Remote sensing UAV/drones and its applications for urban areas: a review," *IOP Conference Series: Earth and Environmental Science*, vol. 169, no. 1, jul 2018.

[2] S. Chaumette, "Collaboration Between Autonomous Drones and Swarming," in *UAV Networks and Communications*. Cambridge University Press, 2017, pp. 177–193.

[3] E. Lygouras, A. Gasteratos, K. Tarchanidis, and A. Mitropoulos, "ROLFER: A fully autonomous aerial rescue support system," *Microprocessors and Microsystems*, vol. 61, pp. 32–42, sep 2018.

[4] J. Besada *et al.*, "Drone Mission Definition and Implementation for Automated Infrastructure Inspection Using Airborne Sensors," *Sensors*, vol. 18, no. 4, p. 1170, apr 2018.

[5] A. Williams and O. Yakimenko, "Persistent mobile aerial surveillance platform using intelligent battery health management and drone swapping," in *Proceedings - 2018 4th International Conference on Control, Automation and Robotics, ICCAR 2018*. IEEE, apr 2018, pp. 237–246.

[6] N. Yamamoto and K. Naito, "Proposal of Continuous Remote Control Architecture for Drone Operations." Springer, Cham, jun 2019, pp. 64–73.

[7] InfluxData Inc., "InfluxDB — The Time Series Database in the TICK Stack — InfluxData," 2018. [Online]. Available: https://www.influxdata.com/time-series-platform/influxdb/