Towards an Automated Flying Drones Platform

Bruno Areias[1,2], Nuno Humberto[1,2], Lucas Guardalben[1], José Maria Fernandes[2] and Susana Sargento[1,2]

[1]*Instituto de Telecomunicações, Aveiro, Portugal*

[2]*DETI - University of Aveiro, Aveiro, Portugal*

Keywords: Automated, Flying Drones, Ground and Drones Communications.

Abstract: Nowadays, some *drone* Flight Controllers (FCs) support basic automation (e.g., GPS waypoint, return-to-home, path flight, take-off/landing), although it requires direct *drone* connectivity (e.g., radio, base station/-Ground Control Station (GCS)) and an extensive knowledge over technical details (e.g., assembly, configuration, battery maintenance, flight, etc.). This paper proposes a novel platform that offers an abstraction layer between the end-user and the *drone* itself, automating most of the *drone* flight requirements. The platform allows to perform high-level *drones* control (e.g., up, down, left, right, GPS go-to and stream follow, return to base, etc.) through end-user communications, contributing with a highly modular event-driven control platform, enabling development of more complex integrations between *drones* and other technologies. The obtained results show that the proposed automated flying drones' platform is able to properly abstract and decouple the direct control, handling up to 32 *drones* with no significant impact on the performance. The platform is also capable of displaying and correlating sensor metrics obtained in real-time during flight.

1 INTRODUCTION

Drones are agile, affordable and diverse, offering an excellent platform to carry devices around (e.g., sensor arrays, cameras, small computers, etc.). They also enable exploration and study of dangerous places (e.g., radiation zones, wild fires, hurricanes, catastrophe SAR, etc.)(Erdelj et al., 2017) without threat to human lives. Since the first appearance of a controllable Unmanned Aerial Vehicle (UAV) in the First World War (Keane and Carr, 2013), evolutions nowadays in technology improved range, communication range, network bandwidth and automation (e.g., autopilot, GPS go-to, etc.).

In terms of *drone* automation, in most cases it still requires humans to interact or overview their operation either directly (in field of view) or through telemetry data (GCS), and for configuration in advance of the flight parameters to the auto-pilot or GPS go-to. Therefore, this creates dependencies on human interaction and extra hardware (radio controller transmitters, computer radio adapters, etc.) for most of the use cases. Moreover, when entering in the field of multiple *drones*, each one requires at least one dedicated active radio, both on the ground and on the *drone*, which makes the integration with other external technologies or applications complex and unsustainable, specially in multi-drone control scenarios.

possible to conclude possibility ...

I would rather write: "Thus, delivering drones as a a service would allow a greater abstraction level to..."

Through analysis of related works, it is clear that several interesting technologies would become possible through integration of *drones* with other platforms. Thus, delivering *drones* as a service would allow a greater abstraction level to *drone* technicalities and requirements, making them easily integrable with external systems and scenarios, such as crop health assessment systems, crowded outdoor event surveillance, wildfire surveillance and assessment systems, etc (Motlagh et al., 2016; Gupta et al., 2016; Chen et al., 2014). Researching about the topic "*drones* as a service" clearly reveals a lack of evolution in this direction.

Our work proposes the building of a highly modular platform to support control abstraction and direct control decoupling, therefore effectively standardizing integration methodologies to allow *drones* as a service where high level commands can be issued to the platform, removing the complexities of actually flying the *drone* itself directly.

Through the course of this work, architectural

Areias, B., Humberto, N., Guardalben, L., Fernandes, J. and Sargento, S.
Towards an Automated Flying Drones Platform.
DOI: 10.5220/0006792405290536
In *Proceedings of the 4th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2018)*, pages 529-536
ISBN: 978-989-758-293-6

and implementation choices to the proposed automated platform are studied, implemented, analyzed and tested using real telemetry information gathered from real drones. The results show that the proposed platform is able to properly abstract and decouple the direct control, handling up to 32 *drones* with no significant impact to the observed performance. The platform is also capable of displaying and correlating sensor metrics obtained in real-time during flight.

The organization of this paper is divided in 5 sections: introduction, related works, automated flying drones platform, overall integration and evaluation setup and results.

2 RELATED WORKS

Unmanned Aerial Vehicles (UAV's), popularly known by *drones*, are already part of our everyday life, mainly due to their attractive cost and popularization. The *drones* area are in constant evolution: besides the laws and ethical issues (Wilson, 2014), *drones* technology are still being established, and many areas are considering *drones* as an alternative to enhance the complex tasks to reduce operating costs (Rossi and Brunelli, 2016). A wide number of applications and services have been proposed for many different areas, including agriculture (Tripicchio et al., 2015), gas detection and mapping (Rossi and Brunelli, 2016), delivering medical supplies (Thiels et al., 2015), search and rescue activities (Câmara, 2014), surveillance (Saska et al., 2014), humanitarian aid (Sandvik and Lohne, 2014), pest and disease control (Amenyo et al., 2014), to name a few.

The fast evolution on the cloud and IoT technologies created and catalyzed several ecosystems of solutions to support distributed system and on-line *drones* control and sensing solution. Cloud computing can improve the limited computational capabilities of resource constrained mobile nodes and enhance the *drones* systems stability (Wang et al., 2017). The work proposed in (Foina et al., 2015) describes a cloud based system for city-wide unmanned air traffic management to keep the city safe using a control system for collision avoidance. However, the authors do not clarify how the proposed cloud-based platform reacts when the flight volumes increase. Therefore, *drones* cloud-based systems can allow an effectively deployed solution in the aftermath of a disaster for effective disaster response and mitigation (Alex and Vijaychandra, 2016). Communication between *drones*, users and the Dronemap planner cloud solution through the MAVLink protocol is presented in (Koubâa et al., 2017), which is supported by commodity *drones*. The delivered experimental results show that Dronemap Planner is efficient in visualizing the access to *drones* over the Internet, and provides developers with appropriate APIs to easily program *drones*' applications. However, due to the asynchronous behavior of Internet, the QoS of *drone* control over the Internet should be monitored, and the impact of wireless communication delay and quality of *drones*' management over the network should be investigated as well.

Finally, and reinforced by survey works (Motlagh et al., 2016; Gupta et al., 2016; Chen et al., 2014), no prior related research work has addressed the goals towards a fully automated flying *drone* platform as proposed in this paper.

3 AUTOMATED FLYING DRONES PLATFORM

The proposed platform, illustrated in Figure 1, comprises two major parts: *Drone* Systems and Ground Systems.

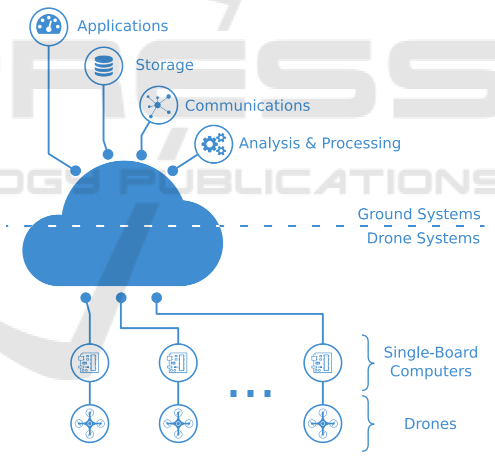

Figure 1: Platform overview.

Interactions between both parts are performed through the usage of publish/subscribe design patterns, ensuring a high level of decoupling, transport layer abstraction and interaction management. This comes even more useful by allowing *drones* to individually subscribe to task specific channels. The same can be said of applications, scripts or systems that either use or belong to the platform. In the ensuing sections, a detailed description of the internal components of both parts will be explored.

3.1 Drone Systems

Drones are a complex aggregation of hardware equipments with their own specifications, technicalities and firmware, but all of them are controlled through a microprocessor with auxiliary sensors that can either be built-in or external to the microprocessor board. Since this sort of technical details are out of the scope of this paper, further reference to these technical aspects will be mostly mentioned through the technical term for this piece of hardware, Flight Controller (FC).

The *Drone* Systems are composed by all the required flight components, each of these will play a critical role towards its safe operation.

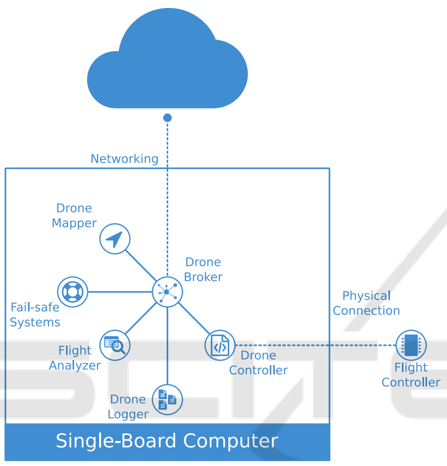

Figure 2: Drone Systems Architecture.

The composition of the architecture is illustrated in Figure 2, and further details of each of the components will follow:

- The ***Drone* Broker** supports the main communication bus on board of the *drone*. The system also has a relay mechanism which is responsible to directly relay back and forth messages with the Ground Systems.
- The **Flight Analyzer** connect to flight telemetry data to process and analyze the behavior of the *drone*. Through pattern detection and behaviour profiling, this system can detect anomalies to actively notify the Ground System and try to fix the issue. As a last resort, it can activate Fail-Safe Systems.
- The **Fail-Safe System(s)** is a mechanism that, when triggered, tries to minimize the consequences of a failure. These can be very basic like preventing takeoff, or forcing a safe landing on the detection of low battery levels.
- The **Drone Logger** taps in to the required broker channels or topic in order to create a local copy of all the events occurred within the *drone* for debugging and registry purposes.
- Represented by a code file, the ***Drone* Controller** acts as an adapter design pattern, translating broker command messages to messages that are readable by the FC, abstracting the platform from the technical aspects required to properly interact with the FC.
- The **Flight Controller (FC)** has the task of controlling the flying *drone* process and correct its behaviour, which properly represents the command end-point and the internal *drones'* telemetry (altitude, temperatures, barometer, accelerometer, acceleration, voltage and cell voltages, GPS) data source generator.
- The ***Drone* Mapper** consists on the further extension of actual geofencing functionalities of *drones*. In direct communication with its other half, Ground Mapper, it supports dynamic map loading based on current GPS data and a given radius. Moreover, it is capable to provide richer information like obstacles and minimum/maximum altitude restrictions, common to urban scenarios.

3.2 Ground Systems

The Ground Systems are composed of services, systems and processes that are auxiliary to flight (e.g.: data storage, management, front ends, etc.) or can extend flight functionalities (e.g.: support in-flight *drone* exchange, automated flight plan for area patrolling or aerial mapping/surveillance).

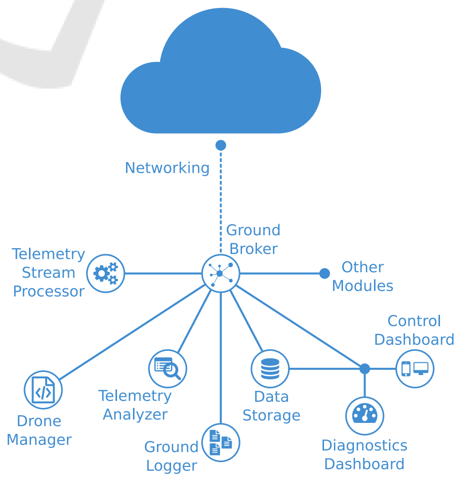

Figure 3: Ground Systems Architecture.

The ground systems' communications are based on brokers which play a key role through their message routing capabilities, development and production instances, which co-exist side by side without the need to build a development oriented deployment of the entire platform. The composition of the ground systems architecture is illustrated in Figure 3. Further details of each of the components will be explored as follows:

- The **Ground Broker**, similar to the *Drone* Broker, supports the main communication bus to the Ground Systems. Due to the possibility of growing (vertically scalable), it has added responsibility of filtering and routing of each individual *drone* message. Moreover, because of its clear potential to be a platform bottleneck for architectural purposes, it must be a cluster (horizontally scalable) capable broker system.
- The **Drone Manager** has the responsibility of tracking the *drones*' status and serve as an internal proxy between the *drone* controlling and the actual controlled *drone*. Being a proxy, it can reroute commands to a different *drone* without the need of major reconfigurations either to the controller of the *drones* involved.
- The **Data Storage** is the representation of a data storage system for later analysis or auditing purposes of the platform.
- The **Ground Logger** connect to the required broker channels or topic to create a local copy of all the events occurred within Ground Systems for debugging and registry purposes.
- The **Diagnosis Dashboard** is a visual front-end that can access performance and sensor data from the entire platform and *drones* with the lowest latency possible, allowing visual trouble diagnosis. Moreover, it shall be capable of review older datasets for history purposes.
- The **Control Dashboard** stands for a GCS like front-end dashboard in which users may control their active *drones* through high-level commands (e.g.: up/down, left/right, etc.). Moreover, they shall allow control for embedded-drone sensors like video cameras, thermal cameras, etc.
- The **Systems Monitor** is responsible for the monitoring of the Ground Systems for automated detection of service failures, network latency, disk usage and alert emission to the platform administrators.
- The **Telemetry Stream Processor** connect to telemetry data sent by the *drones*, and it processes the received data to generate new information to feedback the platform (e.g.: compute number of packets received by *drone*, compute statistical data for *drone* behavior analysis, etc.).
- The **Telemetry Analyzer** connect to the flight telemetry data to analyze the behavior of the *drone* and detect anomalies. Through this system, the Ground Systems can react to behaviour changes to mitigate the issue (e.g.: parachute deployment, emergency landing). Unlike its similar process on board the *drone* due to computational capacity available, it can perform complex analysis of flight parameters potentially gaining extra knowledge and awareness about the *drone* flight conditions.
- The **Ground Mapper** (depicted as other modules) is the concept to extend the natural geofencing supported by most of the drones. It has the responsibility to track each *drone* location and actively update it for hazards, safe zones, landing zones, no fly zones, etc. Using dynamic map loading techniques, therefore saving memory and storage in the drone also removes the need to re-update maps and geofencing configurations, keeping all the drones constantly up to date.
- The **Drone Coordinator** (depicted as other modules) has the task of ensuring that drones in a certain area can coexist without interfering with each other. Tapped to flight telemetry, it can detect distances between each *drone*, up on trespassing a minimum safe distance can notify each of them and even suggest safe actions on how to proceed, like an aerial traffic controller.

4 OVERALL INTEGRATION

This section describes the process on how to provide a persistent means of monitoring and controlling the *Drone* System and constant access to the flight controller. An external single board computer is embedded in the drone in order to manage the permanent connection between the flight controller, which allows not only navigation requests to be assigned, but also the flight and internal sensor data to be acquired, which is a requirement to establish a stable telemetry link.

Given that this point of access for control and information is available at the drone system, a message structure is established for communications between ground and *drone* systems.

```
{
    "type": "objrequest",
    "name": <requested object name>
}
```

Code 1: Information request message structure.

As shown in Code 1, the message follows the JavaScript Object Notation (or JSON) format. Requests for information may be assigned from the ground system and may ask for specific sensor or flight data. The *name* field indicates what is the desired information (e.g. actual altitude, GPS position, estimated flight time). Upon reception of these requests, drone systems query the internal flight controller for data and assemble a response message (Code 2).

```
{
    "type": "objresponse",
    "name": <requested object name>,
    "timestamp": <creation timestamp>,
    "data": <serialized object data>
}
```

Code 2: Information response message structure.

Response messages share a similar structure but add *timestamp* and *data* fields. The *timestamp* field contains a timestamp (millisecond precision) with the moment when the response message was assembled. The requested information is stored in an object which is generated by the flight controller, serialized and then encapsulated in the message's *data* field. In turn, the cloud system can deserialize the data contained in the *data* field upon response reception and obtain its original values.

In addition to information requests, these messages can also be used for navigation purposes. The structure specified in Code 3 details the contents of the messages used in navigation requests

```
{
    "type": "navigate",
    "lat": <requested latitude>,
    "lon": <requested longitude>,
    "altitude_delta": <altitude shift>
}
```

Code 3: Navigation request message structure.

A navigation request may be issued by the cloud system in a message that shall include the target geographic coordinate (latitude and longitude) and, along with this information, an altitude delta. This consists in the altitude difference (meters) between the drone system's actual position and the target position (e.g. an altitude delta of 0 would cause the drone to maintain its altitude along the flight path). This functionality may be particularly useful when, for example, the drone shall get farther from the ground during takeoff, or closer to its landing spot during landing. Upon reception of this message, the drone system will parse the target coordinates and initiate its flight towards the desired direction, until it reaches the given target.

5 EVALUATION SETUP AND RESULTS

Our initial requirement of the architecture is to deploy a system where the *drone* (see Figure 4) is able to carry payloads up to 1 kg (ex.: small sensor arrays, cameras, communication interfaces, etc.), perform vertical takeoff/landing, hovering capability, easy to maintain and with an autonomy of at least 10 minutes.

From several hardware configurations explored, a DIY (Do It Yourself) solution is settled with the following components: 3S - 5000 mAh Li-Po battery, four 30A Simon ESC, four A2212/13T 1000KV rotors with 22mm blades, and most importantly, a Flame Wheel F450 ARF Kit from DJI for the frame and wiring exposure, ease to change peripherals.

To directly control the Electronic Speed Controllers (ESCs), and therefore the behavior of the *drone*, an OpenPilot Revolution[1] is used. Due to the fact that it does not have a built-in GPS module, an external GPS and magnetometer module are connected to it.

The equipment used as payload is: single board Raspberry Pi 2 with a 4G USB dongle, Arduino Nano and a 5V 5000mAh Power Bank. An extra computational node is required to support the *drone* systems, the Raspberry Pi 2 is picked to execute that task, and as the underlying networking USB 4G dongle for end-to-end communications and *drone* control. The Arduino Nano is used for direct control abstraction purposes, and the Power Bank is used to power up both the Raspberry Pi and the Arduino.

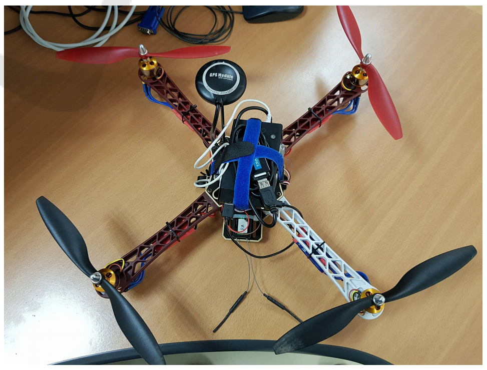

Figure 4: Full *drone* setup implementation.

For all purposes, the *drone* battery can be used to power all components, but in our perspective it is not a desirable behavior. A standing-by *drone* would be draining its own flight autonomy just to power the Raspberry Pi. With a second smaller battery just for

[1]https://librepilot.atlassian.net/wiki/spaces/LPDOC/pages/26968084/OpenPilot+Revolution

the payload, it can keep its flight autonomy capacity and have the *drone* systems active and waiting for activation.

5.1 Results and Discussions

Obtaining precise behaviour parameters such as the number of transmitted messages, latencies and average packets per second is a requirement for assessing the communications reliability of the platform. The values for these parameters may vary depending on the actual state of the drone and the variety of the connected sensors. Additionally, both the length and the frequency of the transmitted messages differ according to the type of information being sent (e.g. GPS position information is less frequent than estimated altitude updates).

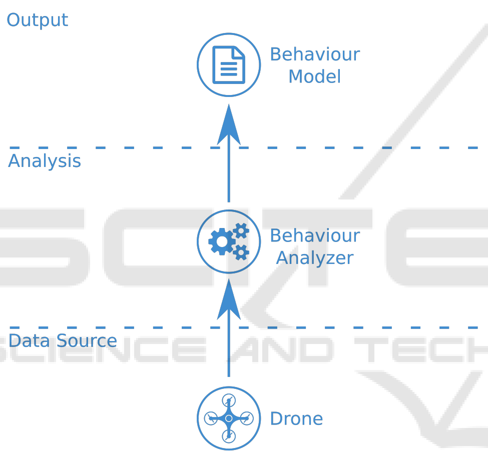

Figure 5: Data acquisition process flow.

To determine how the platform scales in the presence of an arbitrary number of drone systems, a Python-based emulation mechanism has been developed. This emulation must take into account that, like previously mentioned, the length and frequency of messages may vary depending on the state and complexity of the drone system. In an attempt of accurately emulating the behavior of a drone system, the preparation stage of the emulation consists in an automated data acquisition process in which details are obtained about every different type of information (e.g. actual attitude, GPS position, estimated flight time) sent by the flight controller, along with the total length and frequency of each. These details form a model of the behavior of a specific drone system, which is stored in a single file for later access. The data flow of this first stage is depicted in Figure 5.

The second and final stage of the emulation consists in parsing a previously created behavior model and programmatically replicating an arbitrary number of drone systems by replaying messages with every different type of information present in the model, using the observed length and frequency. The flow of this final stage is depicted in Figure 6.

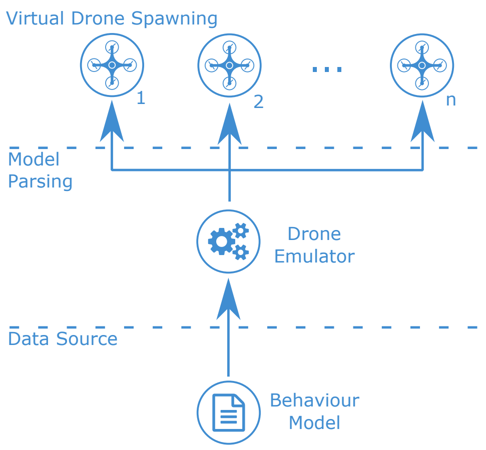

Figure 6: Drone emulation process flow.

With the emulation mechanism, platform performance tests are conducted to determine the platform reaction in terms of throughput and delays to a variable number of *drones*. For periods of one hour, various tests evaluated the amount of packets and delay of the broker based communications using powers of two to determine the number of drones.

To abstract from networking delays and have accurate timings about clock synchronization, the entire experiments are conducted within the same host with a 8 core 2.66 Ghz and 8 Gb of memory, running Ubuntu 17.04. The emulation of the *drones* and the Ground Systems are deployed on the host through the usage of LXC container system.

Our experimentations are performed up to 32 drones, in which the communications keep stable for the entire length of the experimentation period, delivering a median and weighted average of 1 and 1.22 milliseconds of delay respectively. As of the packet throughput, it is kept stable at 22.8 packets per second per *drone*. In Fig 8 the results show that more than 95% of the packets are received below 5 milliseconds, as well as that most of the packets are received below 5 milliseconds.

Taking the results into consideration, we can conclude that the platform behave better in a 32 real *drones* scenario due to the possibility of concurrency, inducing latencies and bottlenecks.

To determine if specific behaviours of the *drone* are viably detectable by the platform modules, we perform multiple experiments with real *drones*. One of the most promising results is illustrated in Fig 7:

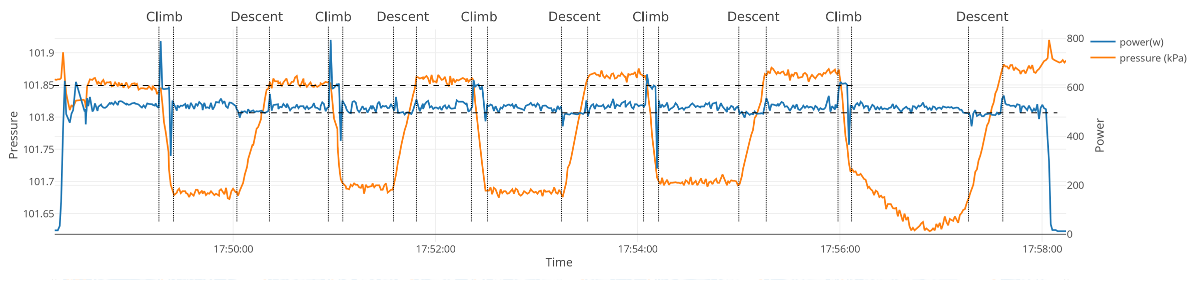

Figure 7: Correlation between power consumption versus changed in altitude.

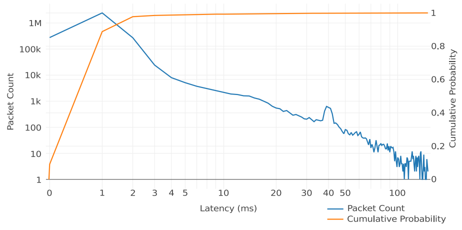

Figure 8: Packet delay analysis of one hour emulation with 32 simulated drones.

the experiment aims to understand if climbing and descent behaviour is systematically detectable by only looking into the power consumption.

The results are conclusive and show that average power consumptions change to stable value ranges in climbing and descent, in the Fig 7 represented by the two horizontal dashed lines. They clearly show a higher and lower, respectively, consumption average in relation to power consumption while hovering or moving horizontally. This behaviour can be used as a triggering mechanism for the fail-safe systems.

6 CONCLUSIONS

This paper proposed an automated flying drones platform, building a modular platform to support control abstraction and direct control decoupling, therefore effectively standardizing integration methodologies to allow *drones* as a service, where high level commands can be issued to the platform, removing the complexities of actually flying the *drone* itself directly. Moreover, the platform supports user-friendly control of drones, which is an important step towards integrating multiple drones and multiple types of drones within the platform, therefor creating a value added tool to develop and support more complex tasks and use cases with usage of flying drones. The results show that the proposed platform is able to properly abstract and decouple the direct control, handling up to 32 *drones* with no significant impact to the observed performance. The platform is also capable of displaying and correlating sensor metrics obtained in real-time during the *drones'* flight.

As future work we plan to integrate external sensors and provide mechanisms for data gathering, as well as thermal cameras for tracking objects and people.

ACKNOWLEDGEMENTS

This work is funded by FCT/MEC through national funds and when applicable co-funded by FEDER PT2020 partnership agreement under the project UID/EEA/50008/2013.

REFERENCES

Alex, C. and Vijaychandra, A. (2016). Autonomous cloud based drone system for disaster response and mitigation. In *Robotics and Automation for Humanitarian Applications (RAHA), 2016 International Conference on*, pages 1–4. IEEE.

Amenyo, J.-T., Phelps, D., Oladipo, O., Sewovoe-Ekuoe, F., Jadoonanan, S., Jadoonanan, S., Tabassum, T., Gnabode, S., Sherpa, T. D., Falzone, M., et al. (2014). Medizdroids project: Ultra-low cost, low-altitude, affordable and sustainable uav multicopter drones for mosquito vector control in malaria disease management. In *Global Humanitarian Technology Conference (GHTC), 2014 IEEE*, pages 590–596. IEEE.

Câmara, D. (2014). Cavalry to the rescue: Drones fleet to help rescuers operations over disasters scenarios. In *Antenna Measurements & Applications (CAMA), 2014 IEEE Conference on*, pages 1–4. IEEE.

Chen, Y., Zhang, H., and Xu, M. (2014). The coverage problem in uav network: A survey. In *Computing, Communication and Networking Technologies (ICCCNT), 2014 International Conference on*, pages 1–5. IEEE.

Erdelj, M., Natalizio, E., Chowdhury, K. R., and Akyildiz, I. F. (2017). Help from the sky: Leveraging uavs for disaster management. *IEEE Pervasive Computing*, 16(1):24–32.

Foina, A. G., Sengupta, R., Lerchi, P., Liu, Z., and Krainer, C. (2015). Drones in smart cities: Overcoming barriers through air traffic control research. In *Research, Education and Development of Unmanned Aerial Systems (RED-UAS), 2015 Workshop on*, pages 351–359. IEEE.

Gupta, L., Jain, R., and Vaszkun, G. (2016). Survey of important issues in uav communication networks. *IEEE Communications Surveys & Tutorials*, 18(2):1123–1152.

Keane, J. F. and Carr, S. S. (2013). A brief history of early unmanned aircraft. *Johns Hopkins APL Technical Digest*, 32(3):558–571.

Koubâa, A., Qureshi, B., Sriti, M.-F., Javed, Y., and Tovar, E. (2017). A service-oriented cloud-based management system for the internet-of-drones. In *17th International Conference on Autonomous Robot Systems and Competitions (ICARSC 2017). 26 to 30, Apr, 2017.*

Motlagh, N. H., Taleb, T., and Arouk, O. (2016). Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives. *IEEE Internet of Things Journal*, 3(6):899–922.

Rossi, M. and Brunelli, D. (2016). Autonomous gas detection and mapping with unmanned aerial vehicles. *IEEE Transactions on Instrumentation and Measurement*, 65(4):765–775.

Sandvik, K. B. and Lohne, K. (2014). The rise of the humanitarian drone: giving content to an emerging concept. *Millennium*, 43(1):145–164.

Saska, M., Chudoba, J., Precil, L., Thomas, J., Loianno, G., Tresnak, A., Vonasek, V., and Kumar, V. (2014). Autonomous deployment of swarms of micro-aerial vehicles in cooperative surveillance. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 584–595. IEEE.

Thiels, C. A., Aho, J. M., Zietlow, S. P., and Jenkins, D. H. (2015). Use of unmanned aerial vehicles for medical product transport. *Air medical journal*, 34(2):104–108.

Tripicchio, P., Satler, M., Dabisias, G., Ruffaldi, E., and Avizzano, C. A. (2015). Towards smart farming and sustainable agriculture with drones. In *Intelligent Environments (IE), 2015 International Conference on*, pages 140–143. IEEE.

Wang, J., Jiang, C., Han, Z., Ren, Y., Maunder, R. G., and Hanzo, L. (2017). Taking drones to the next level: Cooperative distributed unmanned-aerial-vehicular networks for small and mini drones. *Ieee vehIcular technology magazIne*, 12(3):73–82.

Wilson, R. L. (2014). Ethical issues with use of drone aircraft. In *Proceedings of the IEEE 2014 International Symposium on Ethics in Engineering, Science, and Technology*, page 56. IEEE Press.