A Cloud-based Control System Architecture for Multi-UAV

Chen Hong
Science and Technology on Parallel and Distributed Laboratory(PDL)
National University of Defense Technology
Changsha, China
hongchen16@nudt.edu.cn

Dianxi Shi
Science and Technology on Parallel and Distributed Laboratory(PDL)
National University of Defense Technology
Changsha, China
dxshi@nudt.edu.cn

ABSTRACT
Unmanned aerial vehicle is considered one of the most promising technologies. Controlling UAV over the cloud is an emerging research area motivated by the emergence of Cloud Robotics and the Internet-of-Things (IoT). This paper presents a control system architecture based on cloud platform which has been developed in order to allow multiple users to control and monitor multiple UAVs simultaneously irrespective of the location. Furthermore, the system allows to build and allocate missions to UAV and analyze the execution data that collect by sensors. The architecture of the system is fully based on open source software and protocols. To demonstrate the effectiveness of the system architecture, we implemented and validated it using SITL (software in the loop) simulator. Experimental results show that the system is efficient in monitoring and controlling UAV remotely through the Internet.

CCS Concepts
Computer systems organization → Cloud computing

Keywords
Unmanned aerial vehicle; Control system architecture; Cloud platform; SITL simulator.

1. INTRODUCTION
Unmanned aerial vehicle (UAV) is a kind of advanced robot, which is controlled either autonomously by onboard computers or by the remote control of a human operator on the ground. These flying robots have potential in many fields such as detection patrol, agricultural plant protection, aerial photography, package delivery etc. to name a few. Although these low-cost micro UAVs can be used in extensive application areas, there are still many challenges. Since the size and weight of the UAV is strictly limited, its battery and computation are also affected. In fact, consumer UAV does not fly more than one hour in a single flight as well as the current capabilities of UAV is unable to capable perform complex tasks such as image analysis. Besides, the typical use of UAV is to establish a point-to-point connection with them either through telemetry device or a WIFI connection to the WIFI hotspot, the communication range between the UAV and the control location will be limited to a maximum of few hundred meters. In order to make up for the problem of insufficient capacity of a single UAV, multi-UAV cooperation is a suitable approach. There is not too much research on the control system of multi-UAV so far.

In this paper, we proposed a solution to these problems by using cloud computing. Cloud-based control system for UAV could provide extra storage and computation and cloud server with internet connectivity allows controlling and communicating with UAV anywhere anytime without any restrictions on the communication range. The solution was inspired by the concept of IoT and cloud robotics. The Internet of things (IoT) is the network of physical devices, vehicles, and other items embedded with electronics, software, sensors, actuators, and connectivity which enables these objects to connect and exchange data[1]. The UAV move freely in the air and equip with all kinds of sensors and actuators (e.g. camera), which is the typical representative of the IoT intelligent hardware. James Kuffner[2] presents Cloud Robotics at Conference of Humanoids 2010. According to his idea, the robot does not have to store all information and perform intensive computation, connect to server to get data when needed.

There are several works attempted to improve robotic ability by offloading computation to cloud. DAvinCi[3] is a software framework that provide advantages of cloud computation for robot, the authors use ROS(Robot Operating System) [4] distributing messaging mechanism interact with Hadoop[5] cluster of back-end and increase some of the robotics algorithms efficiency through Map/Reduce[6], but they did not address communication reliability. Rapyuta[7] is an open source cloud robotics platform which provide customizable computing environment for robots to offload heavy tasks. A few attempts to deploy UAV to cloud have been made. Gharibi et. al[8] present a conceptual model of Internet-of-Drones(IoD) and lay out an architecture for implementing the IoD. In [9], the authors validate IoD through a real implementation and experimentation. They present a software architecture, named DroneMap Planner, which is a cloud-based management of UAV connected through the Internet. The paper only supports control of one UAV at a time.[1]

In our paper, we consider the case of controlling multiple UAVs at same time. We propose an innovative and modular control system architecture that provides access to multiple UAVs through two different interfaces. The architecture supports multi-user control of multiple UAVs and improves the performance of unmanned systems. We specifically address the control of multiple UAVs over the cloud. The architecture is fully based on open source libraries and it has been designed as a robust and decentralized system. We developed a web-based ground control

ICRCA '18, August 11–13, 2018, Chengdu, China

ACM ISBN 978-1-4503-6530-7/18/08…$15.00
https://doi.org/10.1145/3265639.3265652

[1] The National Key Research and Development Program of China(2017YFB1001901)

station based on the interface provided by this architecture and demonstrated in simulation experiments.

To the best of our knowledge, such a robust and decentralized architecture has not been proposed before. In the following we discuss related work of UAV system architecture in Section II and introducing the modules of our architecture in detail with reasons and solutions in section III. Section IV presents simulation setup and implements a prototype control station based on the architecture. we also demonstrate the feasibility and effectiveness of the control system in controlling multiple UAVs using several use cases. Finally, we draw conclusions and discuss future work in Section V.

2. SYSTEM ARCHITECTURE

Figure 1 presents the architecture of the control system addressing the above problems. It comprises of three parts: UAV Executive Layer, Cloud Layer, Human Operator Layer. Cloud Layer divided into two parts: Cloud Core Layer and UAV Shadow Layer. Cloud Core Layer provides cloud environment support. UAV Shadow Layer acts as a moderator between UAV and Cloud and simplifies the control system development process by taking care of lower level functionality such as flight control, battery management, status transmission and communication. The purpose of each layer is described in detail below.

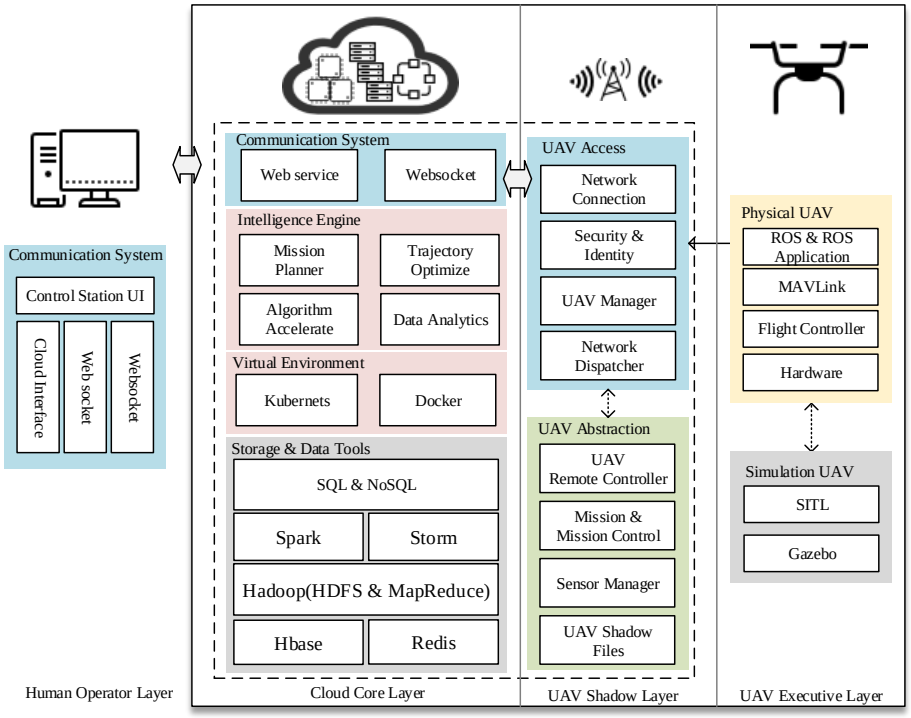

Figure 1. System architecture overview.

2.1 The UAV Executive Layer

There are several ways to simplify the operation of the UAV hardware. On top of the hardware, the flight controller is an on-board computer that couples control information from the hardware with sensor information to adjust the thrust at each propellor and fly the UAV as desired. Besides MAVLink is a lightweight message marshalling library for micro air vehicles that allow UAV to exchange messages with ground control station over kinds of transport protocols (e.g. TCP, UDP, serial, USB). Robot Operating System (ROS) is a flexible robot software framework with the goal of simplifying robot control. MAVROS is a MAVLink extendable communication node for ROS with proxy for Ground Control Station. We can send MAVLink messages to UAV through MAVROS with ROS publish/subscribe communication mechanism. We use these open source software and hardware to interact UAV which advantage is that we can get UAV information and send control instructions without the background of robotics or embedded systems. In fact, although these two methods are optional, we provide two kinds of support considering ROS is widely used.

Considering reduce economical costs of UAV, the repeatability of the experiment environment and the lack of hardware difficulties, we provide support for simulation environments. The SITL (software in the loop) simulator allow us to run Copter without any hardware due to it run ArduPilot flight controller program on our PC and collect data from a flight dynamics model in a flight simulator. Gazebo is a well-known robotics simulator that offers the ability to accurately and efficiently simulate populations of robots in complex physical environment. we can set up gazebo simulator for SITL.

2.2 The Cloud Layer

In order to make the system more modular we split the cloud into two sub-layers. Different components are loosely coupled and each layer has its specific purpose.

2.2.1 *UAV Shadow Layer*

As the name suggests, the purpose of this layer is to abstract UAV into resources in the cloud and provide to client with an operating method that is completely independent of UAV characteristics. In addition, as a bridge between physical UAV and cloud, the layer

is responsible for the data returned by the UAV transfer to the cloud for processing (offload computation) as well as send results processed by cloud to UAV execution (allocate mission). The UAV Shadow Layer decomposed into two main subsystems:

UAV Access: We define it as the underlying communication module. As mentioned above, UAV can send messages through several network protocols so that we need to provide different communication interfaces for different protocols and maintain them. In order to communicate with the ROS node of package, one way is to run a ROS node in the cloud, but this approach is not reliable because the ROS Master node crash easily. After investigation, we used the rosbridge[10] to actively initiate Websocket-based communication to the UAV Executive layer. There are many benefits: on the one hand, Websocket is supported by many programming languages, and it is easier to integrate with web applications than UDP or TCP. On the other hand, rosbridge using json format message transmission, not only supporting heterogeneous operating system but also being lighter. UAV needs to be authenticated to require the cloud services at the access point, once the authentication is successful, the cloud will maintain the connection and save it by a thread pool. Therefore, this module is designed as a multi-threaded server, so as to be more efficient handling MAVLink and rosbridge messages that comes from UAV Executive layer. Since the system is intended for multiple UAVs and multiple operators, permission matching is required between the UAV and control station operators to prevent conflict control. One way is to group the UAVs and set up different group permissions for operators. We need to store some of the UAV inherent information (such as UAV type, size and firmware information, etc.), thus UAV manager is designed to manage UAV information so that we can more quickly maintain UAV resources. Considering that there may be different users accessing the state information of the same UAV at the same time, in order to avoid overloading physical UAV communication link (the onboard computer), we designed a network forwarding mechanism in the Network Dispatcher module which can forward messages to other network ports through the network proxy.

UAV Abstraction: To simplify the client developers understanding of UAV hardware, we have encapsulated UAV on the UAV Access layer to provide simpler UAV control methods, allowing developers to focus on control station development without needing to know hardware. There are four components in this layer. The UAV Remote Controller contains all action information that could be executed on the UAV and represents all the MAVLink Command message and the ROS Topic related UAV action including take-off, land, go-to, return-to-launch, take-photo, etc.

The Mission & Mission Control component is used to assist UAV to accomplish tasks more autonomously. There are many different mission types that offer different behavior. The most common is the waypoint mission. A waypoint mission is a series of pre-defined locations (waypoints) the UAV will fly to. A location is a latitude, longitude and altitude. A series of actions (such as take photo) can be executed at each waypoint. A waypoint mission is uploaded to and executed by the UAV but is limited by the amount of storage in the flight controller, usually only 99 waypoints can be executed per waypoint mission. Therefore, some more complex missions need to be managed and pretreated by Mission & Mission Control component. Mission Control handles execution of missions. Either single mission can be run through dedicated mission operators, or a series of missions and actions can be run serially using the timeline. During the mission, the mission can be manually adjusted through this component as well as the mission speed can be sped up, slowed down or even executed in reverse.

Because UAVs carry many sensors and the data types of the sensors are also different, especially in ROS, different Topics may be used by the same type sensors to publish messages, which brings trouble to the development process. We proposed Sensor Manager module to integrates the sensor information into a unified representation way.

The core of UAV Shadow Files component is to reflect the status of a UAV in the physical world through the Shadow Files that is the digital twin of a UAV in the cloud. UAV reports the internal status to UAV Shadow through the Sensor Manager when system is running, then we save the temporary status using a json-formatted file. Other components can access the status of the UAV through this file. The UAV and the Shadow file can be synchronized with each other in which UAV upload the status to the shadow file and the control station delivers the desired status information to the UAV by the shadow file. The design of this module is to consider the instability of the network. Wireless connectively can be unpredictable in challenging wireless environments. One condition is that the network instability causes UAV to go online and offline frequently. At this time, the information request and control command of the control station will be disorderly. The Shadow file that store in the cloud could avoid this problem: Firstly, it stores the latest status of the UAV and synchronizes immediately once the state changes; Secondly, the shadow file store the control command with the timestamp which can be retrieved after the UAV is reconnected and the execution is determined according to the timestamp. In addition, the Shadow file can also relieve network load pressure. The system has to perform multiple network dispatchers when multiple control stations want to request UAV status, but these results are uniform. The best way is to synchronize the status with the UAV Shadow files in which the control station and the UAV are decoupled, and the capability of the UAV is also liberated. The data flow of the Shadow file is shown in figure 2.

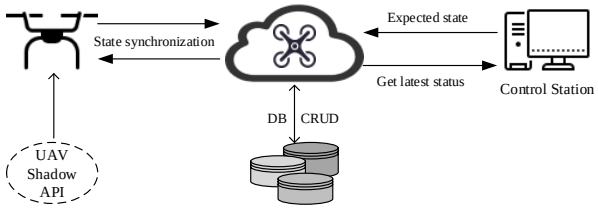

Figure 2. UAV shadow data flow.

2.2.2 Cloud Core Layer

Storage & Data Tools: They provide storage services for kinds of data originated from UAV. Different type of data needs to be stored, retrieved and accessed in the UAV system, such as mission data, environment information and transmitted data including different sensor data (image, video, GPS, etc.) We store different types of data in different databases depending on the applications requirements. For example, SQL databases can be used to store regular structured data, such as the information of UAV device, and its authentication. The NoSQL database can be used to collect unstructured data such as location coordinates, temperature and so on. The UAV control instructions needs to be processed immediately in case of danger, so we use the Storm streaming processing framework, which is one of the best options for near

real-time processing and can provide results with lower latency. For large-capacity files such as UAV mission and flight records, it is suitable for batch operations, which is not high time requirement. Hadoop is a framework dedicated to batch processing. It reads data using the HDFS file system and processes it with the distributed technology Map/Reduce, then extracts useful data finally.

Virtual Environment: The single-node server cannot support large-scale computation with the number of UAVs increase. Virtual machine technology becomes an important means for server clusters. We choose Docker and Kubernetes for centralized server management and scheduling, where Docker is responsible for providing the virtual container runtime environment and Kubernetes is responsible for managing, orchestrating, and scheduling container groups built by the Docker engine.

Intelligence Engine: UAV needs a lot of algorithm support during the task execution, such as task planning, trajectory optimization, SLAM, etc. Intelligence Engine can perform parallel computation through Hadoop cluster and use Map/Reduce to improve algorithm execution efficiency, besides various big data tools in cloud can assist Data Analytics algorithms. In general, they aim at providing smart functionalities and reasoning in cloud for UAV by artificial intelligence.

2.3 The Human Operator Layer

Control station client based on two types of interface development provided by cloud: Websocket and Web Service. Websocket is a full-duplex communication protocol that allows server to send information to clients initiatively. Compared to HTTP long poll and Ajax polling, it consumes less resources to keep cloud and control station connected and ensures that the cloud actively pushes UAV information to the control station. The rosjs library in rosbridge package supports Websocket natively, the MAVLink message is accepted through the socket firstly and then forwarded by the Network Dispatcher to the control station via Websocket. Web service interface is used to develop mission, UAV management and other non-continuous data flow. We use REST Web services to monitor and control UAV for control station through invocation of Web services. Compared with the traditional Web Service, Web Service based on Restful framework has the advantages of less resource consumption, simple realization and so on.

3. SIMULATIONS AND EXPERIMENTS

In this section, the experimental evaluation study using the proposed SITL to demonstrate the effectiveness and performance of the control system architecture are presented. First of all, we show how to use SITL to simulate multiple real unmanned aerial vehicles to make up for the deficiency of experimental conditions. Then we show the web-based control station application based on our system architecture. We used the system to control and monitor UAV over the Internet, in particular through the Web. This is possible thanks to the Web services and Websockets interfaces provided by the Cloud Layer.

3.1 Simulation

The SITL (Software In The Loop) simulator allows we to run Plane, Copter or Rover without any hardware based autopilot and communicate with it using MAVLink over the local IP network. In fact, SITL simulates a flight dynamic model which core is based on the ArduPilot flight control algorithm. So when we start SITL, our PC is just another UAV platform that ArduPilot can be built and run on. However, SITL is a build of the autopilot code using an ordinary C++ compiler which complicate operations. DroneKit-SITL is the fastest and easiest way to run SITL on different OS. Because DroneKit-SITL is developed by Python, it is installed on all platforms through Python's PIP tool and provides a few simple commands to start pre-built vehicle binaries. The following command is used to start SITL for build of copter:

```
dronekit-sitl copter-3.3 --home=-35.363261,149.165230,584,353
```

SITL waits for TCP connection on port 5760 after starting. we may need to observe the UAV state simultaneously through multiple software during the simulation (consider the situation: UAV are controlled via scripts and data steam is acquired via ground station software), but SITL cannot satisfy this requirement because there is only one connection port. To solve this problem, we use MAVProxy forward the MAVLink packets from UAV over the network via UDP to multiple other ground station software on other remote devices. Useful if using multiple ground station computers or relaying the stream through an intermediate node. The following command represents MAVProxy connected to UAV via TCP connection 5760 port and forwards MAVLink data stream to UDP connection 14550 and 14551 ports.

```
mavproxy.py --master tcp:127.0.0.1:5760 --out 127.0.0.1:14550 --out 127.0.0.1:14551
```

Simulating multiple UAV should set DroneKit-SITL parameter of -Instance N, and SITL adds 10*N to all port numbers. The following command will start two simulated UAVs separately on the 5770 and 5780 ports of TCP connection:

```
dronekit-sitl copter -I1
dronekit-sitl copter -I2
```

Note that MAVProxy can only connect to one UAV at a time. Multiple MAVProxy sessions should be used if control of multiple UAVs is desired and one session for each UAV. Figure 3 shows the communication relationship between the simulated UAV and other software. TCP connection is used for linking MAVProxy module with SITL instance. MAVProxy forward message through UDP connection, and we use MissionPlanner to monitor the message forwarded by MAVProxy. MAVProxy connection could also be linked by other software or interface, and we are sending information to the cloud based on this feature.

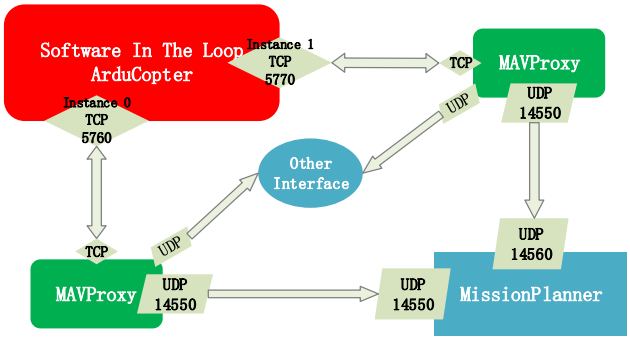

Figure 3. Mutil-UAV SITL communication diagram. In this case two SITL simulator instances are connected to the MissionPlanner via MAVProxy UDP port.

3.2 Web-based Control Station

The control system Cloud Layer is developed by Django that is a Python web framework. We deployed the server into a Docker

instance in a personal computer. It is also possible to deploy it in other public cloud like Aliyun after simple configuration. Once the Server is launched the UAV Access module would starts its service as well as listen external connections and UAV is able to access UAV Access Layer through IP address and corresponding port number.

Figure 4 depicts the web-based control station system graphical user interface. It can be observed that two actives UAVs are shown in the map view with a fully functional control panel and a HUD. The web station contains most important information about the UAV, including altitude, ground speed, air speed, battery, attitude angle, GPS coordinate, fly time and so on. As mentioned above, to ensure timeliness of the information, a reliable bidirectional communication is required to receive MAVLink data stream from UAV Access component. The web control station uses JavaScript Websocket client to connect Websocket interface provided by Cloud. When an UAV is connected to the system, a new UAV control panel widget will be created with a unique color and identifier. The control panel shows the UAV alias that we modified and a green LED that represents the status of the communication link. There are also some buttons to send control command to UAV. From left to right according figure 7 is: takeoff, hover (pause-mission), go-to, take-photo, land, restart-mission, return-home, arm(disarm). In addition, there are some other control buttons, like change altitude. All these control commands are performed through Restful web service interface provided by cloud with remote method invocation. For example, to takeoff, the web control station system invokes the takeoff method of Cloud Layer Web service called: GET http://ip:port/UAVControl-System/api/ControlService/takeoff/x/y. (x represents the UAV identifier and y represents the takeoff height).

The reason why we call the web service to control UAV very succinctly is the UAV Shadow Layer makes a good encapsulation of the basic actions of UAV. We call the take-off service just pass the parameter of height, but there are a lot of repetitive actions to perform at the UAV Remote Controller module: check that UAV is able to arm firstly, then set the mode to GUIDED, after that send arm command and take-off command. In addition, there is no message sent back from the UAV to inform the client that the target altitude has been reached. To address the issue, we obtain the relative height of UAV at regular intervals and compare it with the expected altitude. If the error is within a preset range, it is considered to reach target altitude.

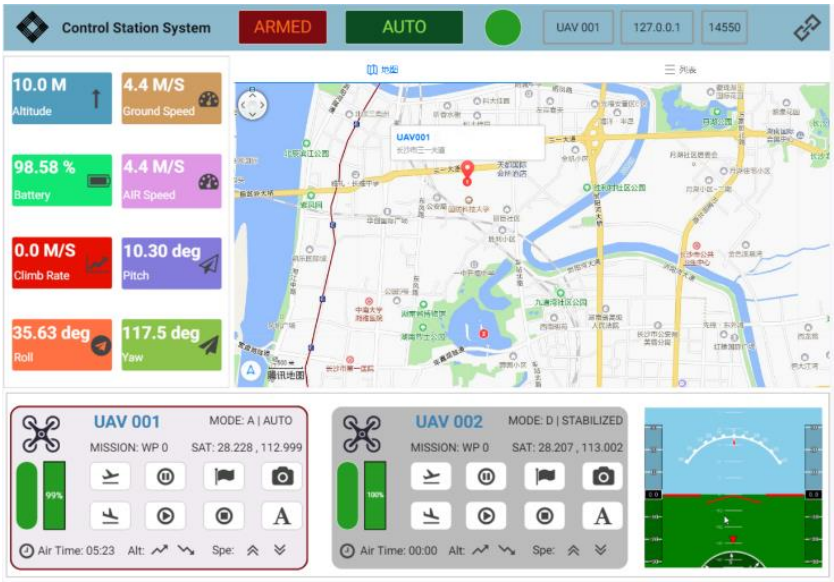

Figure 4. Web graphical control station UI.

4. CONCLUSIONS

In this paper, we present a novel system architecture for multi-UAV. The architecture integrates UAV into Internet-of-Things (IoT) and cloud-robotics and allows to remotely control and monitor multiple UAVs over the Internet at the same time. To verify the effectiveness of the architecture, we implement a web-based control station based on the control system architecture interface. As unmanned system is extremely complex, there are some potential extensions of our architecture. First, we use open source software to accelerate development, but we did not pay attention to security issues. Another challenge is multi-UAV collaborative mission problem. Although our architecture supports to control multiple UAVs, UAVs did not cooperate to complete mission.

5. ACKNOWLEDGMENTS

This work is supported by The National Key Research and Development Program of china (2017YFB1001901).

6. REFERENCES

[1] Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: a survey. *Computer Networks*, *54*(15), 2787-2805.

[2] Kuffner, J. (2010). Cloud-enabled humanoid robots. *Humanoid Robots*.

[3] Arumugam, R., Enti, V. R., Liu, B., & Wu, X. (2010). DAvinCi: A cloud computing framework for service robots. *IEEE International Conference on Robotics and Automation* (Vol.58, pp.3084-3089). IEEE.

[4] Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., & Leibs, J., et al. (2009). ROS: an open-source Robot Operating System. *ICRA Workshop on Open Source Software* (Vol.3).

[5] Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010). The Hadoop Distributed File System. *IEEE, Symposium on MASS Storage Systems and Technologies* (pp.1-10). IEEE Computer Society.

[6] Dean, J., & Ghemawat, S. (2008). *MapReduce: simplified data processing on large clusters*. ACM.

[7] Hunziker, D., Gajamohan, M., Waibel, M., & D'Andrea, R. (2013). Rapyuta: The RoboEarth Cloud Engine. *IEEE International Conference on Robotics and Automation* (pp.438-444). IEEE.

[8] Gharibi, M., Boutaba, R., & Waslander, S. L. (2016). Internet of drones. *IEEE Access*, *4*, 1148-1162.

[9] Koubâa, A., Qureshi, B., Sriti, M. F., Javed, Y., & Tovar, E. (2017). A service-oriented Cloud-based management system for the Internet-of-Drones. *IEEE International Conference on Autonomous Robot Systems and Competitions* (pp.329-335). IEEE.

[10] Crick, C., Jay, G., Osentoski, S., Pitzer, B., & Jenkins, O. C. (2017). Rosbridge: ros for non-ros users.