A user-extensible solution for deploying fog computing in industrial applications

A user-extensible solution for deploying fog computing in industrial applications

Pietro d'Agostino
Politecnico di Torino
AROL Closure Systems
Torino, Italy
pietro.dagostino@polito.it

Massimo Violante
Politecnico di Torino
Torino, Italy
massimo.violante@polito.it

Gianpaolo Macario
AROL Closure Systems
Torino, Italy
gianpaolo.macario@arol.com

***Abstract*— This article discusses the use of the Internet of Things (IoT) in Industry 4.0, which involves connecting different devices and creating an integrated digital workspace to improve efficiency and coordination within a company. IoT allows peripherals to transmit relevant process data to a gathering system, typically on the cloud. However, in some cases, real-time communication is needed, and to reduce costs, firms may use other systems like edge and fog computing. These systems use additional local grids of storage systems near the sensors to execute some elaborations and reduce network bandwidth, latency, and costs of storing in the cloud. The article introduces a fog-computing device called Concentrator, which works as a local center where data are stored and elaborated before being sent to the cloud system. The Concentrator can communicate with different sensors, making it customizable. The article's key contribution is proposing a solution for user-customizable fog-computing systems, using low-cost hardware, by implementing a sandbox through a Linux container. The capabilities of the Concentrator can hence be extended by the user without the intervention of the Concentrator-maker.**

Keywords*— *Arduino, container, docker, industrial automation, IoT

I. INTRODUCTION

The Internet of Things (IoT), which is one of the key features of Industry 4.0, indicates the usage of the internet to connect different devices and create an integrated digital workspace [1]. This connection does not regard only the people but also the machinery inside of a company, to have direct control over how things are working. This technology leads firms to both vertical and horizontal system integrations between the different departments, bringing improved efficiency and coordination [2]. The difference between the two is that while the former is an alignment within the company, the latter is more centered on connecting the various parts of a supply chain [3]. As more time passes, the Internet of Things has taken a significant role in the constant increase of the complexity of industrial systems which cannot pursue their goals if not updated on the recent technologies. It has economic advantages since it improves operational efficiency and production, saves time and costs, having better collaboration and autonomous decisions because of decentralization. It has a new modus operandi, where it let the production change depending on the requests of the market, having also better quality on the final product due to the more personalization of products for the customer [2].

This application is nowadays used in every business, where data are transmitted by peripherals towards a gathering system, typically the cloud. This is called cloud computing and is the most used for data storage, analysis, and elaboration. However, in some cases, real-time communication is needed, and, reducing costs too, firms prefer to apply other systems like edge and fog computing. Both use additional local grids of storage systems near the sensors to execute some elaborations (usually done by the cloud system) to reduce the network bandwidth, latency, and costs of storing in the cloud [4]. This improves decentralized decisions.

Fog computing is an extension of the cloud computing method where the information is gathered from the different peripherals, elaborated, and stored inside a local central. That is why real-time applications are more applicable since the data are exchanged almost immediately between the two devices. The elaborations help have a local idea of how the machines are performing and reduce the amount of knowledge uploaded on the cloud, reducing the space and saving costs [5]. The first main distinction is where the data are located. Cloud computing stores data mainly on cloud services and does there its elaborations. In edge computing, the data are treated by gateway devices positioned near the edge sensors, and so the operations are done on the sensor itself. Instead, for fog-computing, data are transmitted on the LAN (Local Area Network) hardware so that they can be distinguished by sensors and actuators, doing their operations on different devices. The second main distinction is how those data are transmitted: while in cloud computing the measurements are always sent through internet connections and are more related for deep analysis in long terms, in fog and edge computing the transmission can be done offline, more securely and immediately, allowing the real-time solutions [6] [7].

This paper presents a fog-computing device, called Concentrator, which works as a local center where data are stored and elaborated before sending them to the cloud system. It is designed with the collaboration of AROL Closure Systems, a mechanical company producing capping machines, using their systems and their sensor nodes to test and deploy innovative smart solutions. The Concentrator can communicate with different sensors, provided by the producer company and a generic customer, becoming a customizable asset. The personalization of the system can be improved by using virtualization, to create different containers where client software can be uploaded to add features that were not implemented by default. These storages would share the same operating system and will not interfere with the others, to let the user try safely to append different software for the elaborations already done by the Concentrator itself.

The key contribution of the paper is to propose a solution for custom fog-computing systems, using low-cost hardware, trying to implement a sandbox where through containerization it is possible to modify easily and safely the features of a proposed device functioning as central. The AROL assets are used to evaluate experimentally the proposed solution, using the same sensors already mounted on their machines and simulating in a laboratory a first instance of what the final implementation would look like.

This paper is organized as follows. First, section II presents the state of the art, with different applications for virtualization in a fog computing system context; then section III presents the proposed solution, detailing the Concentrator design and the sensors we used, which simulates a heterogeneous scenario where AROL sensor and data processing algorithms are consolidated with third-party sensors and data processing algorithms. In this case, third-party sensors/algorithms mimic the scenario of an AROL customer that performs in-house customization of the fog architecture extending the capabilities of the Concentrator with proprietary code that cannot be used by AROL. Experimental results are discussed in section IV. Finally, section V draws some conclusions.

II. Related Works

The usage of containers and virtualization in fog computing systems has been discussed since the first decade of the 21st century, and recent topics have shown a great improvement on the matter.
As a starting point, the authors in [8] present a technological review of the different methods to perform virtualization, also depending on the different operating systems used, and how the clusters can be efficiently managed. They went through the technological requirements for a cloud system, the architectural principles, and the challenges that these networks must face. About virtualization, an overview is shown, with an explanation of what Docker is and how it works. Most of the users that implement these services use that program, alongside Docker Swarm or Kubernetes to manage the resources. This means deploying the different apps using principally Linux systems. However, examples for Windows and Cloud PaaS (Platform as a Service) are looked through, to present other solutions. They finish with an overview of cluster management and their orchestration, through the usage of different nodes.

An interesting approach has been presented in [9], where a virtualization system has been built using low-cost hardware (Arduino, raspberry pi boards, and some sensors) to create a smart home infrastructure. The network has been divided into different homes, each with different sensors, where the central fog node was represented by a Raggedpod, a micro data center.
Virtualization is used to cope with production-grade requirements and integration of the various systems, using the docker application with individual Linux containers. The aim of the experiment shows some differences, but it was interesting to notice how to proceed with virtualization using similar components. In the article [10], virtualization is used to create a novel on-demand fog creation framework based on requesters' needs anytime anywhere a volunteering device exists, to have the least initialization cost possible. They deployed on-the-fly services through Docker, managed by Kubeadm, with newly formed nearby fogs and efficient orchestration for better response time, studying the latencies of the different processes. To reduce them, in some cases, the clusters are created in advance and fog's IPs are always updated to localize the nearest volunteer to the customer. Every part has different responsibilities, depending on the management of the cluster.
An example of the use of virtualization is done by the experiment presented in [11], where virtualization is applied to a dew computing system. However, even though we used a fog computing system, the purpose of the experiment was interesting: defining a virtual resource as a software-defined component that becomes smart as they communicate with the cloud and stores data locally. They managed to create replicas of the service to give elasticity on an edge level. When a service is saturated by requests, a replica is awakened to solve those that are attending. These smart resources improve the portability of the services. They also focused the security, since they had no central gateways, with the risk of having corrupted devices. For this reason, they implemented a blockchain protocol achieving reliability and security at the edge level. Finally, in the work done in [12], the authors showed how the main framework are principally two: Container-Based Pair-Oriented IoT Service Provisioning, where two devices are cooperating and responsible for the interactions, the second one is Container-Based Edge-Managed Clustering, where serving IoT nodes are being monitored by a manager node. The difference between the two is how the creation of containerization is divided between the different devices, and how the IoT resources are managed by them, in terms of CPU and energy consumption.

In our paper, we present a hardware device, the Concentrator, used as a data storage/data crunching element in a fog-computing architecture, which offers the capability of hosting third-party data processing algorithms. Virtualization is used to sandbox algorithms belonging to two different domains: a native domain, which hosts the application coming from the Concentrator developer, and hosted domain, which is offered to the Concentrator end user for running custom code. Looking at the present system composition, the paper suggests using the first configuration, since there are multiple nodes and the IoT client must guarantee service continuity in case of failure. This approach can ensure a fast management procedure through direct interaction between the cooperating nodes.

This kind of utilization of virtualization has not yet been used, to create a completely customizable system for an industrial end user where the containers can be modified depending on his/her needs. By starting from the same operating system, the different applications will not interfere with each other, offering a secure implementation inside his/her production line.

III. Methodology and Architecture

In this section of the paper, we discuss the architecture of our proposed framework illustrated in Fig. 1. It is a fog

computing system, composed of three layers: the Wide Area Network (WAN) Local Area Network (LAN), and the Personal Area Network (PAN).

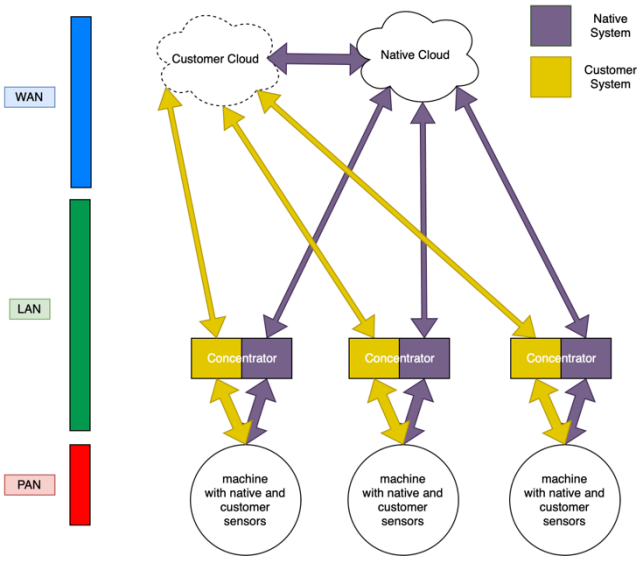

Fig. 1 - General architecture

The first layer is the connection between the fog nodes to the cloud, which is responsible mainly for storing data: from the elaborations made by the fog system to the images used for the containerization with Docker. The second layer is the interconnection between the Wireless Sensor Networks (WSN) gateway with its nearest fog node. It is the part of the system that gathers all the information, elaborates them, and sends them to the cloud. The LAN is composed of one or more devices, which are inserted inside an industrial system, and connected to a machine from a production line. The third layer is the interconnection of all the information extraction devices (the sensors). They can be from the industry for which the concentrator has been created or from external providers, to measure whatever they need.

A. The Cloud

The Cloud is the main component of the WAN layer. It has the role of the database, where all the information is stored, including the history of every device. It will communicate directly with the different Concentrators, gathering the elaborations and saving them in the company computer system. Since this system is thought for machine producers that must be inserted in customer production lines, it should communicate also with the customer cloud, depending on if they are legally authorized to do so, to have a more complete idea of how the different parts are working. Since the fog system uses containerization to host different applications, the Cloud must contain and exchange also the images that the programs (like Docker) should use to create the environments.

B. The Concentrator

The Concentrator is a central device part of the LAN layer, which will be interconnected between the edge sensors, the local systems, and the cloud with different protocols. Its presence would allow to:

1) Reduce the usage of bandwidth and the amount of data transmitted between the cloud and the peripheral sensors.

2) Reduce the delay between the data measurement and the elaborations made by them

3) Implement better control to exploit warnings or other types of anomalies in the system

It will have the ability to gather and store the information given by the components, through different types of communication. The types of data are shown in Fig. 2, with all the parts for a single machine, since, in a production line, each of them will have a central device connected. This device will register all the information about how the system is performing (number of pieces made, machine efficiency, number of functioning hours, etc.) through an ethernet connection. It is linked also to a local system (like a PLC on the machine itself) to set an event-based communication, for anomalies warnings, or other messages. The edge sensors, instead, use a wireless protocol for transmitting their data. Another connection will be established with the cloud not only to send the elaborations from these data but also to download the images used for containerization.

The Concentrator will implement a containerization system, like Docker, which will create a sandbox for third-party's programs that could be used to measure or perform actions decided by the customer. This in fact will create storages inside the system, which will all have the same operating system, without them being interfered with by the others. In this way, it is added flexibility to the system, and it is given the opportunity to customers to add features that are not implemented by default on the Concentrator in a secure and controlled way.

C. The Sensors

The edge sensors are applied in the production line inside the different machines to execute predictive maintenance under the development of Industry 4.0. They will be present in critical places to measure parameters like temperatures, vibrations, or any other kind of value that can be useful to the customer. They will use wireless communication protocols to send data to the Concentrators. It is very important for them to be small, easy to manage, and preferably low cost, to have a functioning system without any large expense. Their initial condition will be machine off–sensor off, to save battery power, and they will be initialized and configured at the start of the system.

IV. Experiment and Discussion

We dedicate this section to describing the implemented scenario and experiments to prove the feasibility and effectiveness of our approach.

To demonstrate the utilization of virtualization on industrial systems and create a sandbox for the final user to be implemented and modified, a first instance of the Concentrator described before is created. The experiment's purpose is to implement multiple containers running through Docker Compose, each doing a different task. By using this system, it can be possible to create a single network for an application, dividing it into multiple containers running at the same time. In this way, these containers can send each other

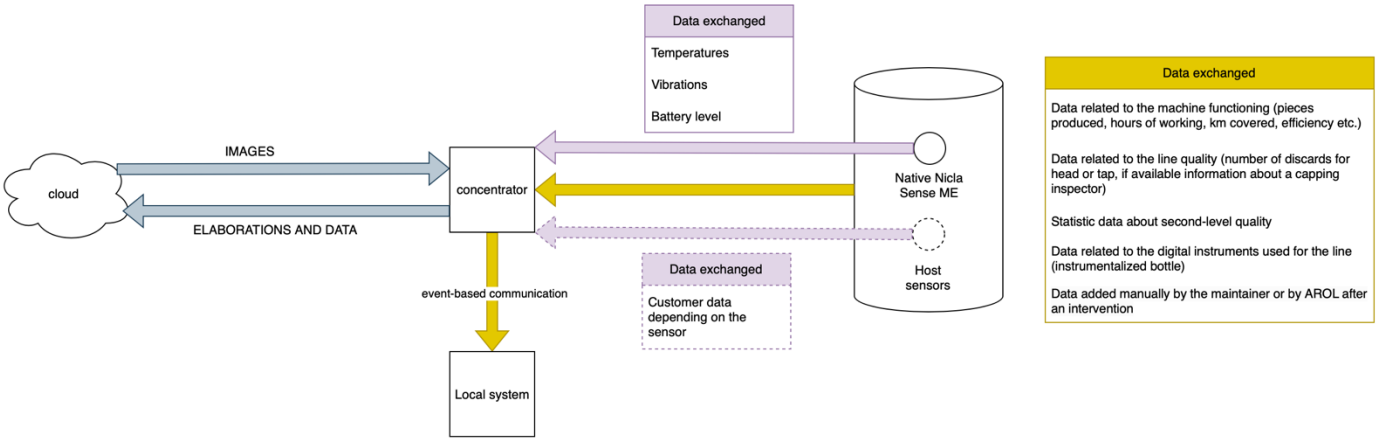

Fig. 2 - Concentrator data exchange

requests and share data. The sensor chosen is the Nicla Sense ME from Arduino. We will use multiple instances of the same board, making them work some as native sensors, and others as host sensors, measuring different variables.

The Nicla Sense ME is a tiny, low-power tool that combines four state-of-the-art sensors from Bosch Sensortec. It analyses 'Motion' and 'Environment' with industrial-grade Bosch sensors that can accurately measure rotation, acceleration, pressure, humidity, temperature, air quality, and CO2 levels. This board is their smallest form factor yet and gives the possibility for Bluetooth Low Energy connectivity (version 4.2), by using an ANNA-B112 module [13].

As for the Concentrator, a first instance is created using a PC Linux using Ubuntu 20.04.5 LTS 64-bit with GNOME version 3.36.8, where the instance of Docker Compose will be implemented using JavaScript and the library node-ble. The version of Docker is 20.10.23, Docker Compose is version 2.15.1, node version 14.21.2, and node-ble version 1.9.0.

The experiment, described in Fig 3, consists of dividing the Concentrator into two main containers using Docker. The first is the *Native* one, which will connect to all the sensors used for the experiment, record their measurements, and produce a report about only the native sensors. The second is

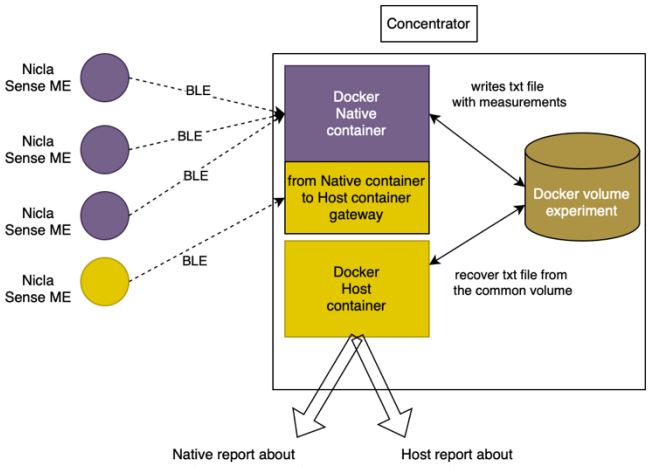

Fig. 3 - Experiment representation

the *Host* one, which will take the measurements from the *Native* and produce a different report with respect to the previous one. To make them communicate, the first container will write two text files: one called host-report, with the measurements from all the sensors, and a second called native-report, with the measurements of only the native sensors. They will be created inside a volume called *experiment* that could be accessed by all the different containers. The application logic handles read-write access to shared data stores. The volume will be both writing and reading for the *Native* container, while it will be read-only for the *Host* container, to prevent accidental data corruption since they have only to read what the sensor measured. For multiple containers using the same volume, it must be individually designed the applications running in those containers to handle writing to shared data stores.

Four Nicla Sense ME will be implemented: three will use their sensors and will be registered as native, while the fourth will be seen as a host sensor, to make it act as an external custom sensor. In this way, the *Native* controller will consider only the three native peripherals for the report, while the *Host* will consider all the sensors used. This will simulate the case in which the native container will study how the machine is performing, while the host container will see only what the final customer would need, producing different elaborations from the same data.

To illustrate the advantage of our proposed architecture, we have taken into consideration the differences between running the solution with the host Linux system and running it with the Docker solution. Three main areas have been exploited: resource usage, the time needed for the solution to run, and the security that it offers.
While for the Docker application, the terminal is used and so are the commands belonging to the infrastructure, the application is made to run in the host Linux system by using the same JavaScript codes, debugged using Nodejs.

A. Resources

The first main characteristic is studied by looking at two factors: CPU usage and memory space needed. The gnome system monitor is employed on the host device to see how the situation changes depending on the solution applied. By

Table 1 - Resources used in the system

Type of system used	Resources	
	Memory space	***CPU usage***
Linux host system	1.6Gb (10.2%) of 15.5Gb	CPU1 : 0.0%
Application run on Linux host system	2.4Gb (15.3%) of 15.5Gb	CPU1: 38.6%
Application run on Docker – Native container	2.8Gb (17.9%) of 15.5Gb	CPU1: 20.04%
Application run on Docker – Host container	2.8Gb (18%) of 15.5Gb	CPU1: 35.3%

looking at the results in Table 1**,** generated by the system monitor, the CPU usage is almost null when the system does not perform any action. Instead, when the application is running on different systems, they do not have similar behavior. In fact, during the *Native* script, the usage of the CPU is at 20.4% while it is at 35.3% during the *Host* script Docker functioning. Instead, when run on the Linux host, it is always at 38.6%. These values are not constant and they fluctuate a bit, however, the range in which they stay is not that far from the one signed by the images.

Regarding memory occupation, the Docker solution occupies more space than the single application on the host system. This is due to different factors: the usage of containers, the reinstallation of dependencies and useful programs, and the overhead introduced by Docker itself due to the copies of packages present in different images. It can be noticed though that the difference between the two is just 3% about the 15,5 GB present on the host system. The problem with the comparison between the two solutions is that we cannot determine how much memory space the different installed packages would occupy. To do so, we should take a mint Linux system and verify the values for every application installed, and all their dependencies.

B. Time

The difference in time is measured by looking not only at the actual performance of the application but also at the time needed for the installation of the different packages. In fact, 4 minutes and 3 seconds have been spent on downloading and installing all the resources for the Docker application to build the images of the two containers. This time however cannot be measured on the host system, since we should erase and reinstall everything from the beginning and see how much it would take to perform everything. The Docker overhead is for sure present also in this experiment since some packages are the same in the images and so doubled. However, the time needed to run the Docker compose application is 36,89s (mean of three times: 36,67s, 37,04s, 36,96s) while the time needed to run the application without Docker is 31,7s (mean of three times 32.15s, 30,56s, 32.4s). This means that the first solution is slower by 13,55% than the other. It must be taken into consideration that there is little time between the closing and the opening of the different containers, instead in the second solution the switch between the two scripts is immediate.

C. Security

The main topic of this comparison, for which this architecture is presented in the first place, is the fact that Docker let the creation of multiple environments share the same operating system and host, without them interfering with the other. A Linux system is more scalable, suitable for different operating systems, and the application can be partitioned into smaller parts, making them communicate with other volumes. A customer can create applications or introduce external features in an already existing system without causing any problems. This can be done only by protecting the Native part of the network, reducing the possibility of administration to the host customer. For this reason, Docker has been used: it limits access to different files and separates containers even though they share the same host architecture and the same operating system. The two containers can only share those data inside the volume created. Files cannot be modified, deleted, moved or any other kind of action can be performed since the Docker system protects them. On the other hand, there is no that kind of limitation on the Linux system. A folder or a file can be deleted or modified without administrator privileges, causing a program to may not work. The only method to let Docker act like the host system would be to use the root user, nullifying the security purpose of the architecture.

V. CONCLUSIONS

With the advancement of Industry 4.0, the need for new systems to collect data efficiently and elaborate them is of paramount importance. New kinds of systems have been presented, from cloud computing to edge and fog computing, improving the communication between the different parts of a firm and performing actions like real-time measurements, reducing costs, and predictive maintenance of the types of machinery. However, it is often impossible to modify those systems without the risk of introducing bugs or problems in the software itself. The new device Concentrator presented in this paper allows the final users not only to benefit from a native system that executes some elaborations, but it lets him/her add new features and/or entirely new applications through the utilization of the Docker containers. We built an example where the system would recognize the sensors used in the architecture and generates two reports to describe their measurements. These are the native report, which will consider only the sensors Nicla Sense ME built-in in the system, and the host report, which considers all the sensors used in the architecture. The application is safe since the two containers do not interfere one with the other, sharing their data only through a common volume. In future work, the goal is primarily to confirm these results by creating a dedicated embedded system for the architecture and then optimize it, due to some particulars that are inefficient. An example is the overhead given by the copies of the same packages in the different images, or the time spent to open and close the containers instead of making them run at the same time. Another objective is to implement in the system an application for the host container as close as possible to a real application in production lines, where this device will be used, according to the help of AROL Closure Systems and Politecnico di Torino.

ACKNOWLEDGMENT

Major thanks to AROL Closure Systems for the availability of their knowledge and assets, without which this work could not be presented. We would like also to thank Politecnico di Torino for the access to some essential equipment for the research. Thanks also to my colleagues who helped me during this period.

VI. BIBLIOGRAPHY

[1] Lucidworks, «The Difference Between Industry 4.0 and IoT,» [Online]. Available: https://lucidworks.com/post/how-are-iot-and-industry-4-related/.

[2] A. Gilchrist, Industry 4.0: The Industrial Internet of Things, Apress, 2016.

[3] M. S.-M. J. M.-S. J. Pérez-Lara, «Vertical and horizontal integration systems in Industry 4.0,» *Wireless Networks,* vol. 26, p. 4767–4775, 2020.

[4] P. Sandonnini, «Internet4Things,» 23 October 2020. [Online]. Available: https://www.internet4things.it/iot-library/cose-il-fog-computing-e-quale-ruolo-ha-nellinternet-of-things/.

[5] R. J. W. G. B. W. Hany F. Atlam, «Fog Computing and the Internet of Things: A Review,» *Big Data Cognitive Computing,* vol. 10, 2018.

[6] M. S. Z. a. K. A. H. Aazam, «Deploying Fog Computing in Industrial Internet of Things and Industry 4.0,» *IEEE Transactions on Industrial Informatics,* vol. 14, n. 10, pp. 4674-682, 2018.

[7] A. S. U. K. a. A. Y. Z. Abbas, Fog Computing: Theory and Practice, vol. 1, Wiley Series on Parallel and Distribuited Computing, 2020.

[8] C. P. a. B. Lee, «Containers and Clusters for Edge Cloud Architectures -- A Technology Review,» in *3rd International Conference on Future Internet of Things and Cloud, 2015, pp. 379-386, doi: 10.1109/FiCloud.2015.35*, 2015.

[9] F. -G. O. a. T. C. L. Letondeur, «A demo of application lifecycle management for IoT collaborative neighborhood in the Fog: Practical experiments and lessons learned around docker,» in *2017 IEEE Fog World Congress (FWC)*, Santa Clara, CA, USA, 2017.

[10] H. S. a. A. Mourad, «Towards Dynamic On-Demand Fog Computing Formation Based On Containerization Technology,» in *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2018.

[11] C. E. a. R. D. M. Samaniego, «Smart Virtualization for IoT,» in *IEEE International Conference on Smart Cloud (SmartCloud), pp. 125-128, doi: 10.1109/SmartCloud.2018.00028*, 2018.

[12] I. F. A. I. a. T. T. R. Morabito, «Evaluating performance of containerized IoT services for clustered devices at the network edge,» *IEEE Internet of Things Journal,* vol. 4, n. 4, p. 1019–1030, 2017.

[13] Arduino, «Arduino Nicla Sense ME,» Arduino, [Online]. Available: https://docs.arduino.cc/hardware/nicla-sense-me.