Received September 22, 2021, accepted October 20, 2021, date of publication October 26, 2021, date of current version October 29, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3122829

Sensor Network Simulator Prototype With Real-Time Environmental Data Monitoring to Build Smart Application

WASHINGTON VELASQUEZ, (Senior Member, IEEE)
Faculty of Electrical and Computer Engineering, Escuela Superior Politécnica del Litoral, Campus Gustavo Galindo Velasco, Guayaquil 090112, Ecuador
e-mail: wavelasq@espol.edu.ec

This work was supported by the Escuela Superior Politécnica del Litoral (Ecuador) through the research project under Grant FIEC-51-2020.

ABSTRACT This paper states an innovative proposal of a simulation/emulation tool for wireless sensor networks for getting real environmental data. The sensor network's creation and design use a mathematical model that optimizes the sensor nodes' location through geographic coordinates (latitude, longitude). The software uses a distributed computational architecture based on containers (Docker), a web application developed in React.js with the ANT Design framework. It is managed through micro-services using the Lagom framework. It also uses the FIWARE-Orion component that controls the flow of sensor data, storing them in the Neo4j database and giving the user the option of using them in any external application. The simulator presents a significant contribution in the WSN field because it allows obtaining data in real-time using a meteorological API. Finally, the application's proposed architecture allows all the simulation data and the sensor network design to be used by creating any outdoors intelligent application.

INDEX TERMS Architecture, environmental data, sensor network, simulation tool.

I. INTRODUCTION

Nowadays, there is a range of equipment that facilitates the daily activities of the human being, e.g. smart lights, smart switches, appliances, and more; where all of them are connected to a data network (wired or wireless), allowing easy configuration, remote control, and monitoring of all these equipment. These devices internally are made up of several sensors, which, depending on the type, perform actions such as reading environmental data, sending information to a server or concentrator, setting monitoring thresholds, and other functionalities that vary according to the data communication model [1]. The sensors have some limitations such as electrical energy consumption, life cycle, and hardware; they do not generate a tremendous intelligent application by themselves. They are generally grouped in a sensor node that communicates with others through a communication link, resulting in a sensor network capable of measuring environmental conditions [2]. Finally, a concentrator (gateway) collects the data from all the nodes to process them and send them to the internet or another application; among the most common smart device in the market with all these characteristics are Amazon Alexa [3] or Google Home [4]. These allow to control other devices and process their data.

Wireless Sensor Networks (WSN) have allowed the creation of a myriad of intelligent environments [5], e.g. smart home, smart building, smart transport, healthcare, and so on; where the use of Internet of Things (IoT) devices have generated real value within smart cities [6]. This network type help automates systems that allow the monitoring, visualization, and remote control of various instruments to create innovative proposals [7]. Likewise, with advancements in Information and Communication Technologies (ICTs), WSNs can use multiple wireless communication protocols, e.g. Bluetooth, WI-FI, Zigbee, LoraWAN, among others [8], [9]. However, several things must consider before implementing a WSN, e.g. environmental metrics/variables, the communication link, the environment, the number of sensor nodes, the type of sensors, the equipment, and the kind of frame/package. Thereby, tools for simulation and emulation of a WSN appeared in multiple programming languages (python, java, c++) to analyze the WSN models from different perspectives [10], e.g. transmitted/received packets, advantages and disadvantages of a wireless communication model, the number of sensor nodes, the protocols, and more. WSN simulators allow the study of wireless networks from

The associate editor coordinating the review of this manuscript and approving it for publication was Hongwei Du.

the physical layer to the network scheme, allowing them to make comparisons that serve before implementation. However, the simulators may not generate the data that corresponds to a realistic environment, each node in a sensor network should be able to obtain data from the simulation environment (temperature, pressure, humidity, and more) to generate applications or case studies based on the location of the sensor network.

This paper presents a tool to simulate/emulate a WSN for outdoor environments and allow the interaction of the sensed data with other computer systems in real-time, and at the same time, use real environmental values of the area to be monitored using a meteorological API. The platform is designed through a distributed computational scheme using current technologies (Docker Containers). The user web application is designed/developed using React.js and the ANT Design framework, offering a non-intrusive, reactive, fast loading, and easy-to-manipulate web tool. The server consists of several micro-services through the Lagom framework. Also, It uses the FIWARE Orion Context Broker (OCB), one of the main components of the FIWARE platform [11]. It allows managing the entire life cycle of data in a sensor (create, read, update and delete) and sending it to other applications where the user needs them. This simulator aims to provide an external tool that allows the deployment of outdoor smart applications that need data for analysis.

The remainder of this paper is divided into seven sections. Section II presents state of the art, describing the components and necessary characteristics of a sensor network simulator and related work, emphasizing the current simulators and the benefit that this tool brings to the field of WSN. Section III describes the mathematical models for creating sensor networks, indicating the different mechanisms that the platform has to generate designs based on the separation radius of the nodes and the coverage area. Section IV presents the platform's distributed computational architecture, describing the graphical user interface, the data communication model, the micro-services, and the external components used to create the sensor networks. Section V describes the case studies to validate the platform (performance), and the hardware and software used in the development environment are indicated. Section VI presents a previous analysis of the platform using the Apache JMeter tool, in which the performance of the web application and the micro-services was analyzed using different metrics. Finally, Section VII presents a viability discussion of the platform with scenarios where the simulator can be used and a description of the future works.

II. BACKGROUND AND LITERATURE REVIEW

Sensor network simulation tools have allowed the deployment of multiple applications in different environments. There are currently many simulation tools where users can choose one depending on their other programming skills and understanding of wireless sensor networks. This section describes state-of-the-art on this topic, emphasizing current sensor network simulation tools and related work.

A. WSN SIMULATORS

The WSN's design-and-implementation are concrete for each application due to the work environment's peculiarity. It is rare to find predictive models that fit multiple sensor networks, mainly because of the network propagation model [12] and the kind of sensor. For this reason, to ensure that the WSN fulfills all its functions, robustness tests must be carried out with complex conditions that allow the variations of its different modules [13], [14], i.e. changes in events, the communication link, the work environment, the nodes, and sensors, transceiver and receiver, protocols and their applications.

1) COMPONENTS OF A NETWORK SIMULATOR

The simulators should have a range of configurations that help shape the required designs. These generally consist of some modules that the user can configure to establish a design of sensor networks. The main modules are [10]:

- *Events*: Represents the moment when an event happens. This module manages the necessary methods for the operation of the network, e.g. process events, compare them, and obtain them as a function of time.
- *Communication Link*: This module establishes the wireless/wired communication model. It allows the different sensor nodes to transmit and receive the data from the sensors. The link contains variable as a bandwidth, wavelength, and propagation model.
- *Environment*: This module sets the work environment similar to the real world, where the physical phenomena of interest to simulate are selected, e.g. temperature, light, humidity, sound, magnetic field, and more.
- *Node (Sensors)*: It represents a single node in the design of a wireless sensor network. The components include, e.g. processor, transceiver, sensors, actuators, energy source (such as a battery), network protocols, location, and identification.
- *Protocols*: The protocols can be from physical and network layers. These can provide services for sending and receiving packets, detecting the load/energy used in each packet, and routing messages through different nodes.

Additionally, a sensor network simulator essential must-have characteristics allow knowing the network's behavior [10], e.g. the best communication protocol and the optimal design of a network. The main ones are described below:

- *Availability*: Simulators are used to test novel techniques with realistic scenarios, where new techniques can be compared to existing ones.
- *Performance and Scalability*: These depend on the type of programming language or graphical user interface, and in turn, are limited to the memory, processor, and register storage size requirements of the computer.
- *Graphical Interface*: It facilitates the design of the experiments and the handling of the simulator modules.

- *Operability and Debugging*: It must have interpreters (commands) that allow you to interact with the simulation and efficiently obtain the expected results. Also, it must incorporate tools that allow finding flaws in the designs.

The large number of variables involved in this type of simulation requires a high-level scripting language, a friendly graphical user interface (GUI), and, as far as possible, support the ingestion or retrieval of extensive data due to the multiple repetitions or the duration of the experiment [15]. However, some simulators are limited by the stack of protocols, precision, and designs' scalability, which compromises the development time, the understanding, and the characterization of the sensor networks in a natural environment.

2) SENSOR NETWORKS SIMULATORS

Given the great demand for current applications that use wireless sensor networks, particularly when measuring environmental variables, researchers have launched a range of WSN simulation tools, which allow users to analyze these networks from different perspectives. Table 1 shows an overview of the most popular proposals for WSN simulators. These can be executed in various operating systems *(FreeBSD, Linux, Windows, Mac OS)* [16], although, to carry out a sensor network's deployment, the user must know the simulator's programming language [17]–[19] and the tool's handling.

The simulators' variety does not necessarily mean that they can be used by all users, especially if they do not have a GUI that correctly shows the simulation's information/setting/output. Some simulators focus on a specific application, e.g. [20], [21] describe an early warning fire detection system, where a network of sensors monitors a specific area.

TABLE 1. Wireless sensor network simulators.

Simulator	Language	Key features
NS-2	C++	- Display environment - Multiple protocols available
TOSSIM	nesC	- Display environment - High accuracy or running
Aurora	Java	- Supports large-scale networks - Validation of time-dependent properties
SENS	C++	- Platform-independent - Assemble application-specific - Work environment as a grid
J-Sim	Java	- Provides support for energy modeling - Support mobile wireless networks. - Component-oriented architecture.
GloMoSim	Parsec	- Parallel simulation capability - Display environment
SENSE	C++	- Memory-efficient, fast, extensible, and reusable. - Component-based simulation - Simulation configuration in parallel or sequential.
(J) Prowler	Matlab/Java	- Probabilistic WSN simulator
CASTALIA	C++	- Fully supports mobility of the nodes - Base on OMNeT++
VisualSense	Java	- Behaviors defined in different domains - Hardware component simulation - Modular platform
Proposal	Web Application	- Micro-services architecture - Realistic data based on study area - Display Environment - Real-time storage - Simulation (Sensors) in parallel

Note: The features are the most noticeable of the simulators.

B. RELATED WORK

Simulators allow the study of a WSN from different perspectives, e.g. [22] describes a process to analyze the reliability and lifetime performance of a WSN from a set of parameters using the JSim simulation tool. The study provides a mechanism to predict the significant parameters of a WSN and avoid creating intricate designs. In [23] describes a study to reduce the energy consumption of a WSN. That paper proposes a technique for predicting battery life (NBLP) with scenarios using different life-cycle factors and voltages/energy the WSN with the BeanAir and XLP simulator. Likewise, In [24] describes a scenario that analyzes AODV (Adhoc on Demand Vector Protocol) in simulation execution time using four simulators NS-2, NS-3, OMNet++, and MATLAB, to provide insight into network protocols currently employed in a WSN. Finally, [25] states a simulation of a WSN using the NS-2 simulator. In this study, the physical and radio signal propagation environments' criteria are analyzed, allowing the classification of the radio-electric propagation models according to the environment, e.g. nature, position, and obstacles. Nevertheless, most simulation scenarios analyze WSN from a functional perspective (physical layer, communication network, propagation model, among others...); this has allowed the deployment of countless applications in different environments; however, If the WSN analyzes is from the point of view of the ingestion/collection of the ''data'', the following stand out. Ref. [26] describes a distributed data aggregation technique using modified K-means to improve WSN lifespan (energy reduction). This method is divided into three stages: 1. sensor reading, data collection, and storage, 2. K-means is used to group the data, and 3. a representative reading of each group is transmitted. The network's performance is managed through the OMNET++ simulator and is based on actual data collected from the WSN and [27] describes a WSN proposal focused on IoT, where the data collected by the sensors uses a machine learning technique that significantly reduces redundant and erroneous data. The proposal allows for greater precision in the grouping of data and improves energy efficiency.

The WSN deployment generally involves using several nodes with multiple sensors that collect data, e.g., temperature, humidity, carbon monoxide, carbon dioxide, and pressure, mainly if the implementation is carried out outdoors (forest, city, parks). There are multiple proposals for WSN deployment where it has been necessary to acquire the required equipment to carry out the tests on-site, e.g., [28] details the structure of a WSN for the detection and mitigation of forest fires. The proposal describes the detection system, prediction algorithms, WSN topology, and location techniques using satellites and drones for fire mitigation or in [29] describes the implementation of a precision agriculture system to mitigate the effects of soil and climate change on crops. This paper results in the optimal ranges to improve productivity and avoid losses due to uncontrolled agroclimatic variables such as relative temperature, soil temperature, humidity, soil humidity, and luminosity. These proposals design WSNs to collect data in real-time, generally to solve a specific problem. The data obtained by the sensors is of vital importance for creating prediction or detection algorithms in an application. However, most WSN simulators do not focus on collecting simulated or real-time data, or that an application external to the WSN or the simulation can use that data without the need to perform a data curation (delete duplicate data). Based on this context, a platform was designed that simulates an outdoor sensor network; the user can design the network with only a few configurations (node separation radius, study area, and starting point for the creation of the network), uses a mathematical model to find the optimal location of the sensor nodes (with choice of changes by the user). Simultaneously, a meteorological API collects real-time data from the nodes' location (latitude, longitude), and a micro-services scheme with a concentrator (gateway) allow external applications to subscribe to the simulated network design to receive the sensors' data. Finally, the sensor network provided by the simulator was previously tested for creating an external application described in [30]; this aims to know the spread of a wildfire while monitoring an area in real-time with the sensor network in the simulator. These features of sending, obtaining, and monitoring data in real-time highlight the simulation platform compared to others.

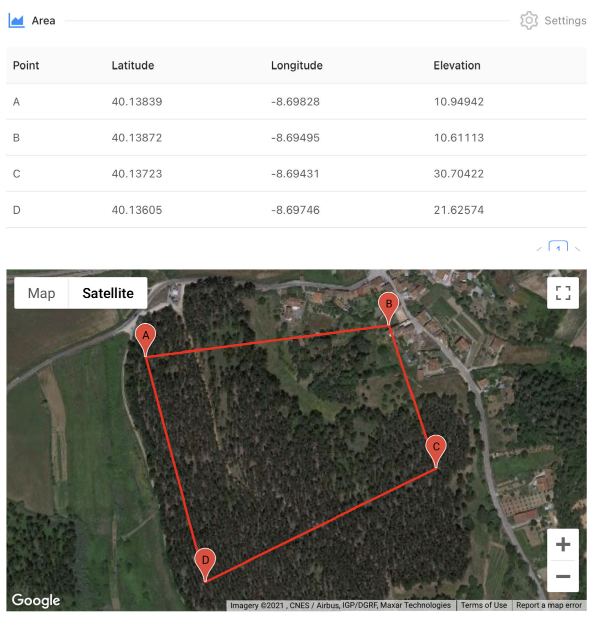
III. SENSOR NETWORK MODEL

The sensor network design requires a coverage area. The area is configured by four stationary points that draw a rectangle on Google Maps. The rectangle R is denoted by:

$$\begin{aligned} R &= (A, B, C, D) \\ &= \{(x_A, y_A, z_A), (x_B, y_B, z_B), (x_C, y_C, z_C), \\ &\quad \times (x_D, y_D, z_D)\} \end{aligned} \tag{1}$$

where A, B, C, and D are polygon points with geographical coordinates (latitude, longitude, elevation), as shown in Figure 1. These points allow for the establishment of the coverage area of the sensor network in the simulator.

The points are necessary to find the best location of the sensor nodes, i.e. find the points (x, y) within the nodes'



Point	Latitude	Longitude	Elevation
A	40.13839	-8.69828	10.94942
B	40.13872	-8.69495	10.61113
C	40.13723	-8.69431	30.70422
D	40.13605	-8.69746	21.62574

FIGURE 1. Coverage area configuration.

coverage area. The coverage area A to draw the sensor network mesh is expressed by:

$$\begin{aligned} A &= f(x, y, t) \\ &= \{(x_k, y_k) \ldots (x_n, y_n)\}; \quad k = \{1, 2, 3, \ldots\} \end{aligned} \tag{2}$$

where (x, y) are geographic coordinates (latitude, longitude) and t represents the simulation time.

The sensor node is defined by:

$$S = (d, f_0) \tag{3}$$

where d is a range of transmission (radius), the area covered by radio signal for message exchange with a neighboring node $S_{neighbour}$. f_0 is the frequency of regular transmissions (5 seconds by default). Each node uses the meteorological API to collect environmental values in real-time.

A. MODELING USING R POINTS

The sensor network can be assembled in five different ways; everything will depend on the initial configuration point and the nodes' separation radius. The mesh is created in swarm mode with a separation r, and it is created circularly, as shown in Figure 2.

The mesh uses Joseph Louis Lagrange's theorem [31], according to which the arrangement of lattice circles with the highest density is when six other circles surround each circle. The density is defined by:

$$\eta_h = \frac{\pi}{2\sqrt{3}} = 0.9069\% \tag{4}$$

Distance between two centers is equal to circle radius multiplied by 2: 2r. When selecting the size of r distance between circle centers should be less than the radio-signal range (d), i.e. $2r < d$. This requirement only satisfies the communication between two neighboring nodes because a

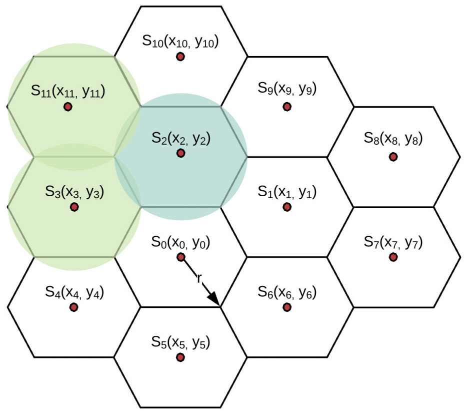


FIGURE 2. Swarm point configuration.

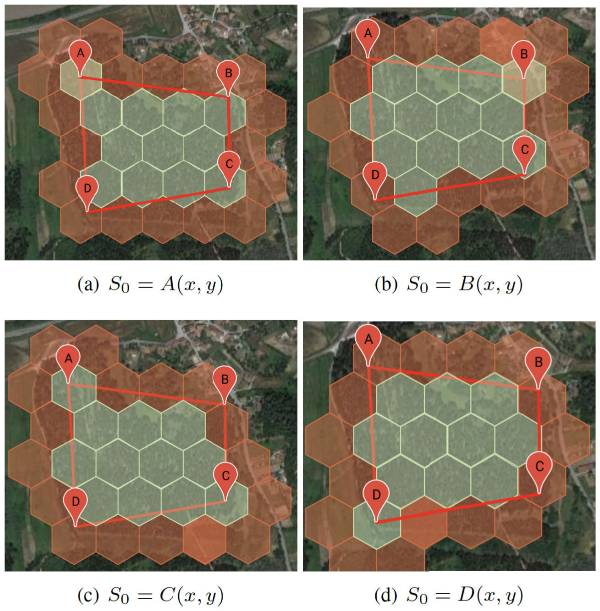


FIGURE 3. Network configuration models starting from an initial point of the coverage. *R* = (*A*, *B*, *C*, *D*).

network is related to the terrain and other adverse conditions, e.g. type of sensor, weather, and more. Besides, the mesh is created in a $2D$ environment; currently, the simulator does not consider the variable "z" (terrain elevation).

Figure 3 presents the different configurations of sensor networks using the R points. It shows how the mesh takes as S_0 one of the points A, B, C, or D, and in each case, the number of sensor nodes to monitor the area is different, e.g. Figure 3(a) shows the configuration of point $A(x, y)$, which begins to create the mesh until it covers the region R (as shown in Figure 2). The number of nodes within R is 12 (green); however, if the user wants to extend the network's design, he/she can make use of the nodes of the contour; in this case, there are additionally 18 nodes (red).

B. MODELING USING THE CENTER POINT

The sensor network can be designed using the central point of the rectangle. The platform calculates this point and considers it $S_0(x_0, y_0)$ with new latitude and longitude values.

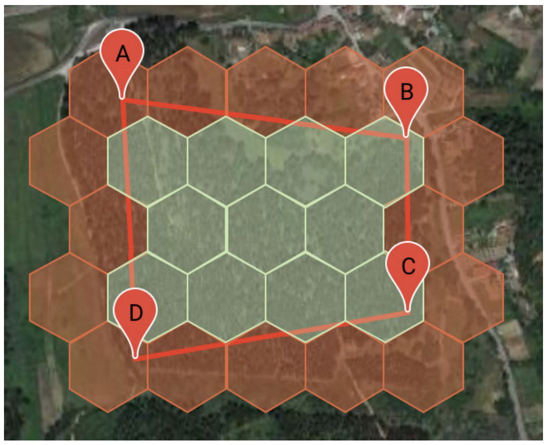


FIGURE 4. Configuration using the center point of the *R* coverage region.

This model allows creating a more homogeneous network, and, as far as possible, the use of the sensor nodes is less than those used with the R points. Figure 4 shows a central point network, resulting in a sensor network with 11 primary nodes (green) and 16 complementary nodes (red).

IV. SIMULATION PLATFORM ARCHITECTURE

This section describes the different stages and technologies used in the platform's implementation. Figure 5 illustrates a general scheme of the distributed architecture of the application, where it is detailed how the technologies interconnect with each other and certain functionalities of the server.

A. USER WEB APPLICATION

The development of the user's web application was with Ant (https://ant.design) Design Framework and React.js (https://es.reactjs.org). These tools allow creating an application with fundamental characteristics such as multiple communication channels, event handling, reactive modules, and independent components/views [32]. This application deployment type helps the proposed architecture because several communication lines exist in the same web component. Thus, a sensor node can establish communication with the OCB and other applications to keep updated each variable's state-monitored.

Figure 6(a) shows a general view once the sensor network has been configured/designed (Section III). Figure 6(b) shows the network nodes so that the user can interact with them (turn on, turn off). When the sensor node is on, an iterative process (5 seconds by default) reads information from the OCB component and displays it in an orderly fashion, as shown in Figure 7.

B. ORION CONTEXT BROKER (OCB)

OCB is one of the main components of the FIWARE platform (https://www.fiware.org/). This component allows an application to manage the entire flow of data by creating entities. The sensor node is configured as an entity within OCB, as shown in Figure 8; in each entity's parameter (temperature,

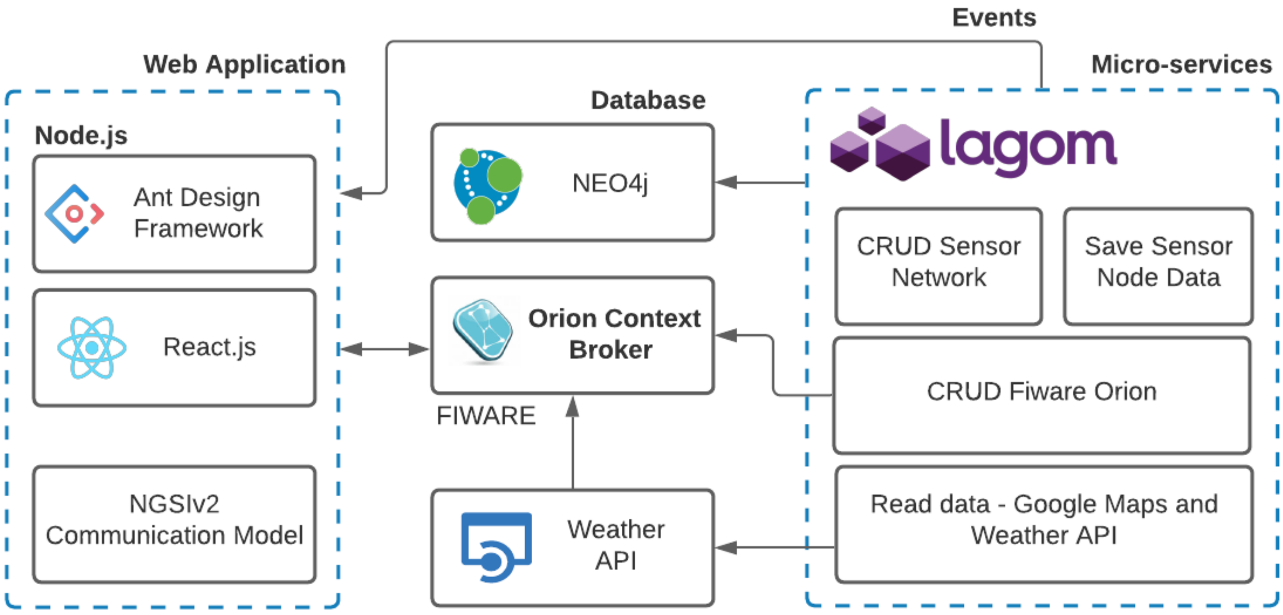
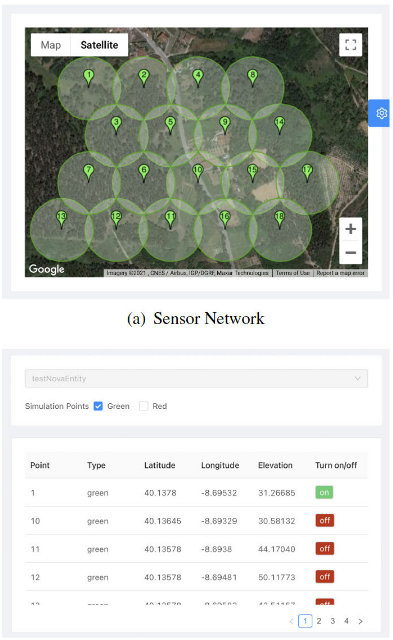


FIGURE 5. The general technological architecture of the simulation platform.


Map
Satellite
Google
Imagery ©2021 , CNES / Airbus, IGP/DGRF, Maxar Technologies | Terms of Use | Report a map error

(a) Sensor Network

testNovaEntity

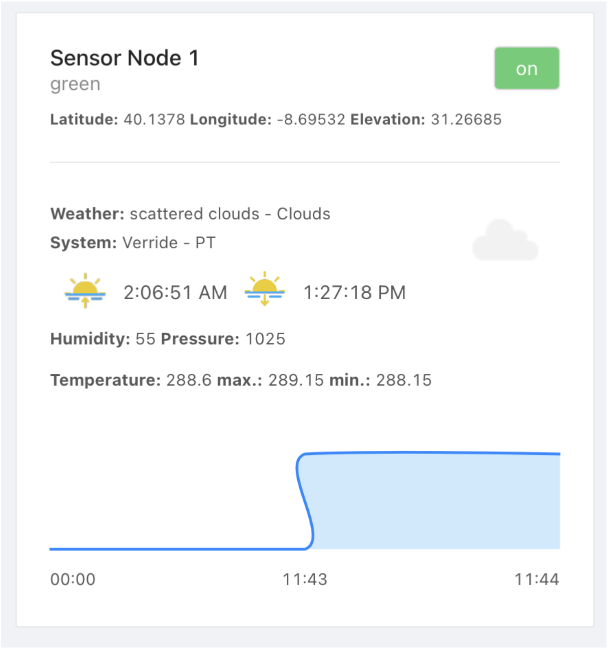
Simulation Points [✓ Green] Red

Point	Type	Latitude	Longitude	Elevation	Turn on/off
1	green	40.1378	-8.69532	31.26685	on
10	green	40.13645	-8.69329	30.58132	off
11	green	40.13578	-8.6938	44.17040	off
12	green	40.13578	-8.69481	50.11773	off

< 1 2 3 4 >

(b) User interaction with sensors

FIGURE 6. General view of the simulation platform (interaction with sensor nodes).



Sensor Node 1
green
on

Latitude: 40.1378 **Longitude:** -8.69532 **Elevation:** 31.26685

Weather: scattered clouds - Clouds
System: Verride - PT

2:06:51 AM 1:27:18 PM

Humidity: 55 **Pressure:** 1025

Temperature: 288.6 **max.:** 289.15 **min.:** 288.15

FIGURE 7. Sensor node information (web application).

pressure, humidity), CRUD can be applied through REST API calls. The entity is described in detail in Appendix A.

OCB uses the publish-subscriber communication model defining NGSIv2 REST API messages (https://fiware-orion.readthedocs.io/en/master/). This model allows different external applications to connect to a sensor network's entities to receive the simulation's sensed data. In this way, subscriptions can be created based on the entire entity or specific data (Appendix B); the data can be used to create multiple applications in different environments. The OCB component uses the MySQL database by default to save entities information and subscription. It is important to note that the components work as Docker containers.

C. DATA COMMUNICATION MODEL

The different modules that make up the platform, from the user's web application to the micro-services (server), use the JSON format. However, different classes have been created in the Scala programming language to manage the data flow (server) to handle persistence and data serialization. Message schemes are described below.

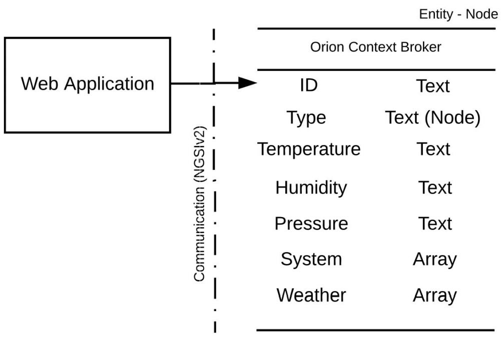


FIGURE 8. **Sensor node information (OCB scheme).**

1) WEB APPLICATION

The web application connects to the micro-services to extract data from the network (design, creation, storage). However, the *ngsi.js* (https://www.npmjs.com/package/ngsijs) library is used to connect with OCB; this allows a direct connection with the entities to manage the simulator's parameters in real-time.

2) METEOROLOGICAL API DATA

While the sensor node is on (simulation is running), a micro-service begins to extract information using the node's position (latitude, longitude) from the web. The data is obtained through the interface provided by "Weather API" (https://openweathermap.org/api). The user needs to previously obtain an API key to be able to use the API. The following REST API call uses the *"Current Weather API"* with the format: http://api.openweathermap.org/data/2.5/weather?lat=lat&lon=lon&appid=API key

The web-service response model in JSON format is as follows ("latitude": 40.1378,"longitude": −8.6953):

```
{"base":"stations",
 "clouds":{"all":40},
 "cod":200,
 "coord":{"lat":40.1378,"lon":-8.6953},
 "dt":1614703415,"id":2732782,
 "main":{"feels_like":284.54,
         "humidity":55,
         "pressure":1025,
         "temp":288.6,
         "temp_max":289.15,
         "temp_min":288.15},
 "name":"Verride",
 "sys":{"country":"PT",
        "id":6896,
        "sunrise":1614668811,
        "sunset":1614709638,
        "type":1},
 "timezone":0,
 "visibility":10000,
 "weather":[{"description":"scattered",
             "icon":"03d",
             "id":802,
             "main":"Clouds"}],
 "wind":{"deg":290,"speed":4.63}
}
```

Finally, a micro-service receives the response; it is sent to the OCB component so that the web application can read the information.

3) MICRO-SERVICE TO OCB

The data received by the meteorological API is tagged to be sent to OCB. The message format is:

```
{"ID";{SENSOR}, "weather": {"value":{
 "id":{"value":"802",  "type":"Text"},
 "main":{"value":"Clouds", "type":"Text"},
 "description":{"value":"scattered␣clouds",
 "type":"Text"}, "icon":{"value":"03d",
 "type":"Text"}}, "type":"Array"},
 "sys": {
  "value":{
   "country":{"value":"PT", "type":"Text"},
   "sunrise":{"value":"16", "type":"Text"},
   "sunset":{"value":"16", "type":"Text"},
   "name":{"value":"Verr", "type":"Text"}},
   "type":"Array"},
 "temperature":{"value":"2","type":"Text"},
 "pressure":{"value":"10", "type":"Text"},
 "humidity":{"value":"55", "type":"Text"},
 "temp_min":{"value":"28", "type":"Text"},
 "temp_max":{"value":"28", "type":"Text"}
 }
```

D. MICRO-SERVICES

The platform uses the Lagom framework (https://www.lagomframework.com) with "Scala" language programming to process the different functionalities of the application. Lagom allows the creation of micro-services that are responsible for the following functionalities:

- Get data from the meteorological API.
- Use the Google Maps API to obtain geographic data.
- CRUD of sensor network designs.
- Store the network in the Neo4j database.
- Communication with OCB.
- Periodic updating of network data.

It should be noted that the functionalities described allowing to the management of the complete cycle of a simulation of a sensor network, i.e. the design, interaction, and data storage.

E. DATA STORAGE

The platform uses the Neo4j (https://neo4j.com) database to save the sensor networks' designs and the information of the nodes obtained from the meteorological API. The objective of using this database is because it stores the information in a graph so that the sensor network can use the benefits of this type of data structure, i.e. edges, to create relationships between sensor nodes (communication) and the nodes that would be each sensor node in the simulation.

Figure 9 shows the structure of the graph in Neo4j. Each design of a sensor network handles the same scheme. The central node (WSN) can have one or many sensor nodes (Node), and in turn, each (Node) can have one or multiple

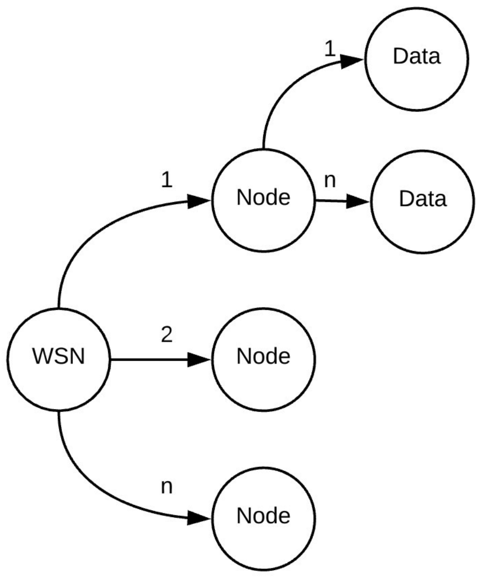


FIGURE 9. Structure of a WSN within Neo4j database.

nodes (Data). The application manages the nodes (Data) so that the information is not repeated; this is due to the OCB component's use, i.e. when there is a change in the entity's parameters, only that value or dataset will be stored in Neo4j. The purpose of this process is to avoid data redundancy and have better communication flow management. e.g. the temperature is a value that does not have sudden changes over time, and it would not be necessary to store that data every second; consequently, unnecessary data traffic on the network is avoided.

V. CASE STUDY

For the platform validation, it carried out two scenarios, which are described as follows:

Scenario 1: The system is evaluated with various requirements per minute on different application pages (network design and configuration), and communication with Orion (OCB) is controlled through a distributed environment via CRUD requests. Hardware and Software requirements are described in Table 2.

Scenario 2: The microservice-based architecture using Lagom is evaluated by the Apache Jmeter application submitting various CRUD requests.

VI. APPLICATION ANALYSIS

This section presents an analysis of the application using the Apache Jmeter tool. The analysis is performed on the web application pages and the microservices. The test scenario involves using three computers to simulate the distributed environment within a wireless local area network (WLAN). The components are divided into:

- Computer 1: WEB Application
- Computer 2: Microservices (Lagom)
- Computer 3: OCB and Neo4j (Docker Containers)

TABLE 2. Requirements for local hardware and software.

Hardware	Software	
	FrontEnd	BackEnd
Ubuntu 18.04.5 LTS	NodeJS LTS x64	FIWARE Orion 2.4.2
AMD A9-9425 r5	Npm 6.14.9	Docker 19.03.13
Ram 12 Gb	Yarn 1.22.10	MySQL 5.7
SSD 100 Gb	And Design Librerias	Neo4j

TABLE 3. Communication analysis.

	Design	Simulation	Orion	TOTAL
# Samples	10000	10000	10000	30000
Average	905	754	852	837
Min (ms)	2	2	2	2
Max (ms)	3438	3335	3714	3714
Std. Dev.	577.67	382.97	542.02	511.80
Error %	650%	20%	330%	333%
Throughput	299.09673	309.94297	311.64298	893.06978
Received KB/sec	316.98	326.06	347.86	960.94
Sent KB/sec	59.20	62.34	66.22	182.05
Avg. Bytes	1085.2	1077.3	1143.0	1101.8

TABLE 4. Micro-services analysis.

	MService WSN	MService Node	TOTAL
# Samples	43905	43855	87760
Average	64	64	64
Min (ms)	5	2	2
Max (ms)	282	265	282
Std. Dev.	15.43	15.14	15.29
Error %	0%	0%	0%
Throughput	731.34776	731.68494	1461.86264
Received KB/sec	769.20	769.56	1537.53
Sent KB/sec	267.11	283.67	550.33
Avg. Bytes	1077.0	1077.0	1077.0

A. APPLICATION PAGES AND ORION

In this scenario, the web application pages and the communication with Orion were tested. The analysis is carried out in:

- Test 1 (Desing): Design/creation of the sensor network
- Test 2 (Simulation): Simulation of the sensor network.
- Test 3 (Orion): Read information.

Table 3 shows the results of the test. Ten thousand samples (request) were performed for each test. It is observed that the error of the pages for the design and simulation of sensor networks is minimal, even with the libraries and mechanisms they use. The performance rate showing the number of requests (traffic/users) that the server has addressed every minute is 53.584,187/minutes, with a standard deviation of 511. Therefore, it is demonstrated that the page loading and the communication with Orion are exemplary and can withstand a maximum load.

B. MICRO-SERVICES

The analysis of the micro-services was carried out in two main ones:

- Test 1 (MService WSN): Read the information of a WSN from the Neo4j database
- Test 2 (MService Node): Read information from a specific node sensor.

Table 4 shows the results of the tests of these scenarios. 43.905 Samples (Request) for test one and 43855 samples for test two were used. The micro-services did not show any error of packet loss. The throughput (traffic/users) is 87.711,759/minutes, with a standard deviation of 15, demonstrating that the proposed architecture based on micro-services can address this type of simulation requirement.

VII. CONCLUSION AND FUTURE WORKS

This paper describes a platform for creating outdoor sensor networks. The platform makes use of a range of technologies that help to create a powerful web application. The sensor network creation model allows the deployment of five network models, where the user can choose the best design that adapts to a potential client's needs, i.e. consider the physical equipment of the sensor network. The simulator allows external applications that need a network of sensors (design) with environmental data (temperature, pressure, humidity, wind direction, and wind speed, among others) can obtain this information with basic configurations.

The user can simulate the network and use it to interact with other applications; this is due to Orion allowing applications to subscribe to entities to send them the information. The simulator has been used to monitor dry forests in Guayaquil (Ecuador) to predict ignition points, and forest fire spread. This example gives the assurance that the simulator has potential in the deployment of smart applications. The platform has certain limitations in its current state. It is necessary to have an infrastructure for the components' installation (servers) due to the distributed design of the web application; this differs from other simulators, which are generally installation packages for different operating systems. The sensor networks only contemplate a 2D modeling (surfaces), and it is only possible to choose one of the five models provided for network design compared with other simulators.

The platform is still under development, and functionalities are being incorporated that allow handling communication protocols and physical characteristics of the sensors, e.g. the type of sensor, the propagation of each node, sensor networks using Particle Swarm Optimization (PSO), type of

```
curl localhost:1026/v2/entities/${uuid} -s
-S \ --header 'Accept:␣application/json' |
python -mjson.tool
{
"id":"cabed392-3871-4b29-9250-c97b3229b5e1",
"type": "NodeWSN",
"humidity":{"type":"Text","value":"71"},
"pressure":{"type":"Text","value":"10"},
"temp_max":{"type":"Text","value":"28"},
"temp_min":{"type":"Text","value":"28"},
"temperature":{"type":"Text","value":"0"}
"elevation":{"type":"Text","value":"3"},
"latitude":{"type":"Text","value":"4"},
"longitude":{"type":"Text","value":"-8.694"},
  "sys": {
    "metadata": {},
    "type": "Array",
    "value": {
      "country":{"type":"Text","value":"PT"},
      "name":{"type":"Text","value":"Ver"},
      "sunrise":{"type":"Text","value":"1"},
      "sunset":{"type":"Text","value":"1"}
    }
  },
  "weather": {
    "metadata": {},
    "type": "Array",
    "value": {
      "description":{"type":"Text","value"
         :"few␣clouds"},
      "icon":{"type":"Text","value":"02n"},
      "id":{"type":"Text","value":"801"},
      "main":{"type":"Text","value":
         "Clouds"}
    }
  }
}
```

```
curl localhost:1026/v2/subscriptions/${uuid}
-s -S  \ --header 'Accept:␣application/json'
| python -mjson.tool
{
    "id": "601f285f6bd12fbe95674e40",
    "description": "A␣subscription␣to
␣␣␣␣get␣info␣about␣Node:␣${uuid}",
    "expires": "2040-01-01T14:00:00.00Z",
    "status": "active",
    "subject": {
        "entities": [
            {
                "id": "uuid",
                "type": "NodeWSN"
            }
        ],
        "condition": {
            "attrs": [
                "temperature"
            ]
        }
    },
    "notification": {
        "timesSent": 8,
        "lastNotification":
        "2021-02-06T23:45:11.00Z",
        "attrs": [
            "temperature"
        ],
        "onlyChangedAttrs": true,
        "attrsFormat": "normalized",
        "http": {
            "url":"http://IP_ADDRESS:9000/
␣␣␣␣␣␣␣␣␣␣␣␣wsn/orion/save"
        },
        "lastSuccess":
        "2021-02-06T23:45:11.00Z",
        "lastSuccessCode": 200
    },
    "throttling": 5
  }
```

micro-controller, node energy source (battery), energy consumption, and the elevation of the ground. However, all the previous sections' functionalities allow managing the sensors' data and their respective designs. The platform will soon be available for user use to get environmental data in real-time of a sensor network.

APPENDIX
COMMUNICATION MODEL (NGSIv2) IN FIWARE-ORION TO CREATE AN ENTITY AND SUBSCRIPTIONS

A. SCHEME OF A NodeWSN ENTITY

As shown at the bottom of the previous page.

B. SCHEME OF A SUBSCRIPTION

As shown at the bottom of the previous page.

REFERENCES

[1] Y. Hajjaji, W. Boulila, I. R. Farah, I. Romdhani, and A. Hussain, "Big data and IoT-based applications in smart environments: A systematic review," *Comput. Sci. Rev.*, vol. 39, Feb. 2021, Art. no. 100318.

[2] C. Ioannou, V. Vassiliou, and C. Sergiou, "An intrusion detection system for wireless sensor networks," in *Proc. 24th Int. Conf. Telecommun. (ICT)*, 2017, pp. 1–5.

[3] R. Bogdan, A. Tatu, M. M. Crisan-Vida, M. Popa, and L. Stoicu-Tivadar, "A practical experience on the Amazon Alexa integration in smart offices," *Sensors*, vol. 21, no. 3, p. 734, Jan. 2021.

[4] K. Noda, "Google home: Smart speaker as environmental control unit," *Disab. Rehabil., Assistive Technol.*, vol. 13, no. 7, pp. 674–675, Oct. 2018.

[5] D. Antolín, N. Medráno, B. Calvo, and F. Pérez, "A wearable wireless sensor network for indoor smart environment monitoring in safety applications," *Sensors*, vol. 17, no. 2, p. 365, 2017.

[6] S. Hanif, A. M. Khedr, Z. A. Aghbari, and D. P. Agrawal, "Opportunistically exploiting Internet of Things for wireless sensor network routing in smart cities," *J. Sens. Actuator Netw.*, vol. 7, no. 4, p. 46, 2018.

[7] A. González and W. Velásquez, "Event viewer design of a wireless indoor sensor network used in emergencies," in *Proc. 1st Int. Conf. Comput. Appl. Inf. Secur. (ICCAIS)*, Apr. 2018, pp. 1–5.

[8] A. Lavric and A. I. Petrariu, "LoRaWAN communication protocol: The new era of IoT," in *Proc. Int. Conf. Develop. Appl. Syst. (DAS)*, May 2018, pp. 74–77.

[9] J. Chu and W. Han, "Internet communication system protocol based on wireless sensor," *Radioelectron. Commun. Syst.*, vol. 62, no. 8, pp. 422–429, Aug. 2019.

[10] H. Sundani, H. Li, V. Devabhaktuni, M. Alam, and P. Bhattacharya, "Wireless sensor network simulators a survey and comparisons," *Int. J. Comput. Netw.*, vol. 2, no. 5, pp. 249–265, 2011.

[11] T. Zahariadis, A. Papadakis, F. Alvarez, J. Gonzalez, F. Lopez, F. Facca, and Y. Al-Hazmi, "FIWARE lab: Managing resources and services in a cloud federation supporting future Internet applications," in *Proc. IEEE/ACM 7th Int. Conf. Utility Cloud Comput.*, Dec. 2014, pp. 792–799.

[12] S. Sundresh, W. Kim, and G. Agha, "SENS: A sensor, environment and network simulator," in *Proc. 37th Annu. Simul. Symp.*, 2004, pp. 221–228.

[13] R. C. Estrella, I. Marin-Garcia, J. A. Munoz-Arcentales, V. Asanza, and W. A. V. Vargas, "Communication network model that supports a sensing system based on robustness criteria," *IEEE Latin Amer. Trans.*, vol. 16, no. 10, pp. 2600–2608, Oct. 2018.

[14] G. Chen, J. Branch, M. Pflug, L. Zhu, and B. Szymanski, "SENSE: A wireless sensor network simulator," in *Advances in Pervasive Computing and Networking*. Boston, MA, USA: Springer, 2005, pp. 249–267.

[15] W. Velásquez, A. Munoz-Arcentales, and J. S. Rodriguez, "A case study: Ingestion analysis of WSN data in databases using Docker," in *Proc. 1st Int. Conf. Comput. Appl. Inf. Secur. (ICCAIS)*, 2018, pp. 1–6.

[16] G. Gautam and B. Sen, "Design and simulation of wireless sensor network in NS2," *Int. J. Comput. Appl.*, vol. 113, no. 16, pp. 14–16, Mar. 2015.

[17] M. Jevtić, N. Zogović, and G. Dimić, "Evaluation of wireless sensor network simulators," in *Proc. 17th Telecommun. Forum (TELFOR)*. Belgrade, Serbia: Citeseer, 2009, pp. 1303–1306.

[18] D. Pediaditakis, Y. Tselishchev, and A. Boulis, "Performance and scalability evaluation of the Castalia wireless sensor network simulator," in *Proc. 3rd Int. ICST Conf. Simulation Tools Techn.*, 2010, pp. 1–6.

[19] P. Baldwin, S. Kohli, E. A. Lee, X. Liu, and Y. Zhao, "VisualSense: Visual modeling for wireless and sensor network systems," Citeseer, Princeton, NJ, USA, Tech. Rep., 2005, vol. 5.

[20] A. Aksamovic, M. Hebibovic, and D. Boskovic, "Forest fire early detection system design utilising the WSN simulator," in *Proc. XXVI Int. Conf. Inf., Commun. Autom. Technol. (ICAT)*, Oct. 2017, pp. 1–5.

[21] W. Velásquez, A. Munoz-Arcentales, T. M. Bohnert, and J. Salvachúa, "Wildfire propagation simulation tool using cellular automata and GIS," in *Proc. Int. Symp. Netw., Comput. Commun. (ISNCC)*, Jun. 2019, pp. 1–7.

[22] N. Cao, P. Liu, G. Li, C. Zhang, S. Cao, G. Cao, M. Yan, and B. B. Gupta, "Evaluation models for the nearest closer routing protocol in wireless sensor networks," *IEEE Access*, vol. 6, pp. 77043–77054, 2018.

[23] S. Yasin, T. Ali, U. Draz, and A. Rasheed, "Simulation-based battery life prediction technique in wireless sensor networks," *NFC IEFR J. Eng. Sci. Res.*, vol. 6, pp. 166–172, Oct. 2018. [Online]. Available: http://nijesr.com/ojs/index.php/archive/article/view/186

[24] R. Sharma, V. Vashisht, and U. Singh, "Modelling and simulation frameworks for wireless sensor networks: A comparative study," *IET Wireless Sensor Syst.*, vol. 10, no. 5, pp. 181–197, Oct. 2020.

[25] A. E. Mouaffak and A. E. B. E. Alaoui, "Considering the environment's characteristics in wireless networks simulations: Case of the simulator NS2 and the WSN," *Int. J. Inf. Commun. Technol.*, vol. 14, no. 4, pp. 427–438, 2019.

[26] A. K. Idrees, W. L. Al-Yaseen, M. A. Taam, and O. Zahwe, "Distributed data aggregation based modified K-means technique for energy conservation in periodic wireless sensor networks," in *Proc. IEEE Middle East North Afr. Commun. Conf. (MENACOMM)*, Apr. 2018, pp. 1–6.

[27] I. Ullah and H. Y. Youn, "Efficient data aggregation with node clustering and extreme learning machine for wsn," *J. Supercomput.*, vol. 76, no. 12, pp. 1–27, 2020.

[28] K. Grover, D. Kahali, S. Verma, and B. Subramanian, "WSN-based system for forest fire detection and mitigation," in *Emerging Technologies for Agriculture and Environment*. Singapore: Springer, 2020, pp. 249–260.

[29] J. M. Nunez V., F. Fonthal, and Y. M. Quezada, "Design and implementation of WSN for precision agriculture in white cabbage crops," in *Proc. IEEE XXIV Int. Conf. Electron., Electr. Eng. Comput. (INTERCON)*, Aug. 2017, pp. 1–4.

[30] K. Castro-Basurto, F. Jijon-Veliz, W. Medina, and W. Velasquez, "Outside dynamic evacuation routes to escape a wildfire: A prototype app for forest firefighters," *Sustainability*, vol. 13, no. 13, p. 7295, Jun. 2021.

[31] C. G. Fraser, "Joseph Louis Lagrange's algebraic vision of the calculus," *Historia Math.*, vol. 14, no. 1, pp. 38–53, Feb. 1987.

[32] Y. Mao, F. Ren, G.-C. Wang, and X.-R. Hu, "Building city planning information system based on the newest script style library: Take ReactJS and ant design as an example," *Geomatics Spatial Inf. Technol.*, no. 8, pp. 81–84, 2017.

WASHINGTON VELASQUEZ (Senior Member, IEEE) received the Ph.D. degree in telematics system engineering and the master's degree in telematics services and network engineering from the Universidad Politecnica de Madrid, Madrid, Spain. He is currently a Professor with the Faculty of Electrical and Computer Engineering, Escuela Superior Politecnica del Litoral, Guayaquil, Ecuador. He has authored/coauthored several articles in indexed journals and has led projects related to sensing and networking. His research interests include telemetry, remote control, smart cities, and big data.