International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 5 (2018) pp. 2624-2629
© Research India Publications. http://www.ripublication.com

Implementation of M2M Protocol for Wireless Sensor Networks Oriented to Its use in Smart Cities Supported by IoT

Irene Castañeda Lopez, Jhon Guzman Arias and Holman Montiel Ariza[1]

[1]Assistant Professor Technological Faculty, District University Francisco José de Caldas
Calle 68 D Bis A Sur No. 49F – 70, Bogotá D.C., Colombia.

[1]ORCID ID: 0000-0002-6077-3510

Abstract

This research focuses on the implementation of machine to machine (M2M) protocols as communications support for the integration of wireless sensor networks oriented to their use in the context of smart cities under the framework of the Internet of Things (IoT). This paper aims to show the integration of communication technologies that allow adapting this type of solutions to the IoT, this is carried out through the implementation of the FIWARE platform and the coupling to a WSN supported on Zigbee technology operating in API mode. In the document can be seen the block diagram of the proposed technological solution, configuration parameters and the development of tests to validate the operation of the installed platform.

Keywords: WSN, Middleware, Zigbee, Internet of Things

INTRODUCTION

It is not possible to talk about smart cities without leaving aside the technology of the Internet of Things (IoT), because this technology directly supports its vision and concept, seeking to generate a series of applications and services that improve the quality of life of citizens and even manage to substantially support the city administration processes [1]. However, this process involves the integration of heterogeneous technologies that generate a large amount of information to process and analyze [2], this is where it is necessary to develop technological solutions that serve as resource and information management systems that support this work.

Similarly, it must be taken into account that every day the number of elements that are connected to the web is increasing, not only referring to conventional mobile devices such as smartphones, tablets or PC. On the other hand, the increase of wireless sensor networks is growing day by day, its diverse applications are associated with intelligent parking systems [3], lighting controls [4], business systems and markets, home automation applications [5], sensors for prevention and detection of emergencies, precision agriculture [7], among many others. These multiple applications lead to the capture, transmission, processing and storage of a fairly high volume of information, generating a series of challenges to the processes of technological integration that must be carried out.

The objective of this paper is to show the technological integration carried out between producers and consumers of context in the framework of the Internet of Things (IoT). It is intended to describe the implementation associated with a WSN supported on Zigbee technology through its native protocol M2M in API mode and the coupling to a FIWARE middleware platform, in order to provide a functional, adaptable and scalable solution to any requirement established by the functions of a WSN within the framework of an intelligent city.

METHODOLOGY

The proposed development consists of two functional stages, the implementation of the FIWARE NGSI platform and the integration of the wireless sensor network with Zigbee technology using its API protocol [10], see Fig. 1.

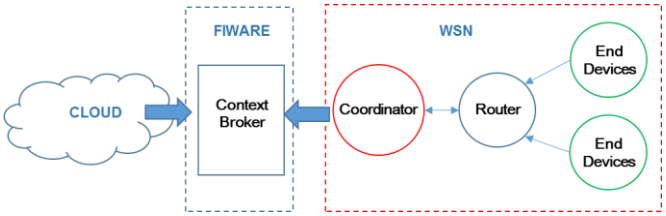

Figure 1: Block diagram of the proposed system.

FIWARE Platform

FIWARE is an open source platform, which enables the connection and development of applications for smart cities, since it offers interoperability and standard data models. Likewise, "FIWARE provides means to produce, collect, publish and consume large-scale context information and exploit it to transform the application into a smart application" [6]. Among the basic components for the development of internet applications of things are the following modules: Context data module, IoT module and Security module.

The aforementioned components allow the development of advanced interfaces, as well as the development of different applications that can be used by all the sensors that are connected to a specific platform. These components cover from the acquisition of data and the execution of commands at the level of physical devices, to the use of data from external applications.

• FI-WARE architecture

FI-WARE is composed of elements called Generic Enablers, "GE", which enable the interaction between different sectors

[7], among which we can mention: Cloud Hosting, Data / Context Management, Internet of Things (IoT) Services Enablement, Applications / Services Ecosystem and Delivery Framework, Security and Interface to Networks and Devices (I2ND). The components of the FI-WARE architecture for IoT cover everything from the data acquisition and the commands execution at the level of physical devices to the data use from external applications, see Fig. 2.

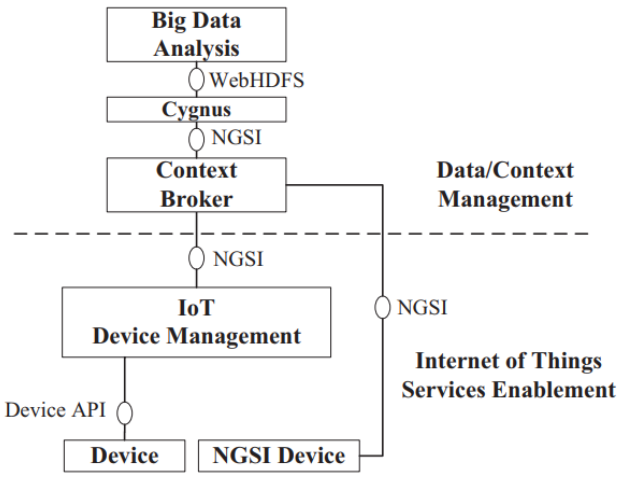

Figure 2: FI-WARE architecture for IoT [9]

NGSI

NGSI is defined as a standard, which enables the connection or communication of the information generating devices and the consumer or client devices of this information with the Context Broker, through the "definition of the interfaces that a context manager must implement" [8]. NGSI, allows the standardization of the APIs implemented in the FIWARE architecture, through the definition of data models applicable to the management of context information, through the use of entities also defined by the Context Broker.

API Protocol Zigbee Modules

The construction of wireless sensor networks on Zigbee technology is widely favored when using its Xbee API Protocol. This way work API is oriented specifically to the communication between applications, in such a way that it is managed to guarantee a high efficiency both in the transmission of information, as in the habilitation of services between the diverse modules that are part of a network; in general terms, these types of interfaces are not designed for direct human interaction [10].

The implementation of the API Protocol involves the manipulation of a series of frames composed of a series of specific bytes from which all the functions of the Zigbee can be validated. Its basic structure is composed of: Start Delimiter, Length Bytes, Frame Data Bytes and Checksum. Some of the frame types of the API mode are: 0x08 AT command, 0x09 AT command (queued), 0x17 Remote Command Request, 0x8A Modem Status, 0x10 TX request, 0x92 RX I/O data received among many others [10].

IMPLEMENTATION AND RESULTS

Oriented to the realization of a wireless sensor network, as part of the implementation should be considered the right platform, one of the best services coupled to the FIWARE platform is the Orion Context Broker. The implementation of this service will allow to provide information management in a robust manner. To do this, function programming is done through the FIWARE NGSI V2 standard, which allows structuring the data frame coming from the sensors and thus the manipulation of the information. All this in a context enabled for the subsequent implementation of visual interfaces in which the information can be presented in a more orderly way.

Orion Context Broker

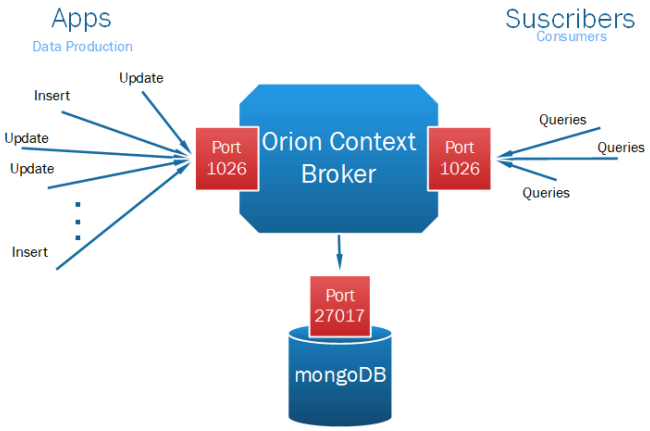

Figure 3: Flow of the Context Broker.

Fig. 3 shows the operational structure of the Orion Context Broker, its main components and the way in which it interacts. This component is one of the main elements within the FIWARE platform defined in the context of information, a service that can mediate and manage the sending and receiving of data under the definition of the NGSIv2 standard, implemented as a data model in the Context Broker. Thus, it considers the information as an element of the context called entity, to then administer the updates and query its attributes. The Context Broker would thus have a virtual representation of the physical devices, stored in mongoDB.

The Context Broker, is the component that within the FIWARE architecture governs the information of the data producers (sensors) and keeps them available for data consumers at their discretion (WEB applications, SmartPhone, other sensors, etc.). About this definition Orion Context Broker enables infinity of external applications oriented to the internet of things in an easy and efficient way, summarizing all existing tasks in the complexity of the capture and information administration.

The context broker defines two operations based on the persistence of information for producers and consumers:

1. publication of entities and their attributes when dealing with information sent by producers, this task is interpreted in the Context Broker as an update of the context (updateContext).
2. reading of entities and their attributes when a request for reading (queryContext) is sent from consumers.

All this context information is robustly handled in large amounts of entity messages, by the Orion Context Broker, the operations of the producers and consumers are structured by the specification of the standard or REST interface, through which data is obtained or there are generated operations on them in various possible formats, within the most well-known XML and JSON, thus allowing a simple manipulation of the data.

Installation of Orion Context Broker

The resources for the installation are:

-Virtual Box version 5.1.30

-Virtualized service [9]

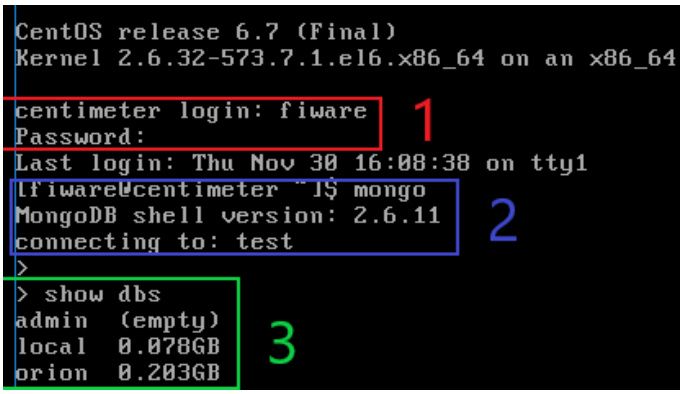

Figure 4: Verification of connection to database.

The virtual machine downloaded in Virtual Box must be imported, enabling all the permissions to communicate by network with the host computer. When starting the virtual machine, it must be identified before the operating system with the ***fiware*** username and password.

Fig. 4 shows the process to be performed even from the start of the guest operating system, to verify connectivity to the database. In step 1, the user is identified in the system based on specified credentials. In step 2 the "**mongo**" command is executed in order to know if the context has connectivity to the mongoDB database of the Context Broker, and if the connection is satisfactory the system enters the mongo shell and returns the version used (in this case 2.6.11) and the name of the instance with which the connection was achieved, in this case and by default: test. Finally, in step 3 it is executed the "**show dbs**" command, so the response describes the databases available in the mongoDB instance, in this case orion for the operation of the broker context. To exit the mongo shell the "exit" command is executed.

Once the connection to the database has been verified, it is necessary to verify the service of the Orion Context Broker, for this it is needed to know the IP address of the virtual machine. In Fig. 5 this procedure is detailed. Executing the command "**ifconfig**", it is possible to consult the parameters corresponding to the network interface of the operating system in the virtual machine, in this case 192.168.8.5. Once the IP address is known, the first REST request will be made to the Context Broker, for which the following link is entered in the host computer's Internet browser: http://192.168.8.5:1026/version (The corresponding port 1026 is specified) to the listening port of the Context Broker).

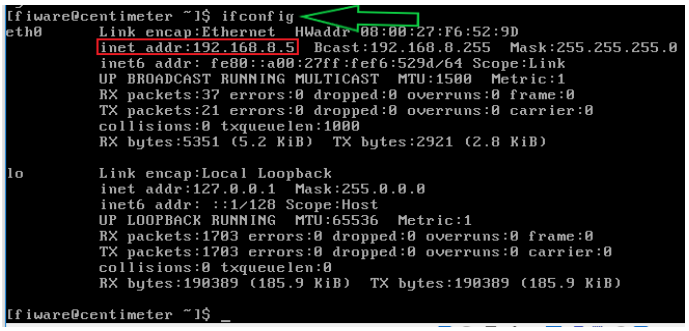

Figure 5: Verification of connection to database.

Table 1 shows the descriptors associated with the verification of service availability, see Fig. 6.

Figure 6: Confirmation of the execution of the Orion Context Broker.

Table 1: List of descriptors

<version>	Describes the version of the Orion Context Broker service that is running.
<uptime>	Describes the time of high in the service for each instance, that is, the execution time since the system started.
<git_hash>	Verification of the compilation hash in the FIWARE GIT repository
<compile_time>	Date of compilation of the version.
<compiled_by>	Author of the compilation of the version.
<compiled_in>	Version compilation system.

At this point the platform would be correctly configured and available to perform the requests for creation, update, query and deletion of entities in the data context of the FIWARE platform.

Basic functions programming

The basic functions on any data consist in the creation, updating, elimination and consultation of the records in a system.

Defined as an API, which is implemented in the Visual Studio 2015 development environment and with Python 3.6 programming language on the host computer and based on the FIWARE NGSI standard to make requests to the Orion Context Broker. These requests are made through REST calls in JSON format. Next, the structure of the project and initialization of parameters in the file config.ini is observed, facilitating the modification of these without intervening the source code of the solution.

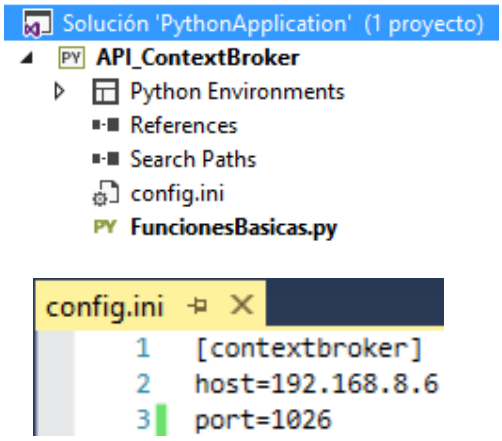

Figure 7: Structure of the project in VS 2015

Implementation

The following libraries must be imported into the source code: Request, Json and Configparser.

1. Reading of INI file

In this way the host and port variables are defined and initialized. They will keep this value in the API execution time and will be used to make the REST calls to the guest host where the Orion Context Broker is in service, see Fig. 8.

```
CONFIG_FILE = "config.ini"
parser = configparser.SafeConfigParser()
parser.read(CONFIG_FILE)
sec = parser.sections()
if len(sec)>0:
    host = parser.get('contextbroker', 'host')
    port = int(parser.get('contextbroker', 'port'))
else:
    print("No file INI")
```

Figure 8: Reading configuration parameters

2. Defining the JSON format

In Fig. 9 it can be seen 4 methods that meet the definition of the FIWARE NGSI standard. In number 1, a list of attributes is defined, depending on the type of data. Method number 2 defines the structure for the insertion of entities, there the attrib parameter references a list of attributes generated in method number 1. Method number 3 defines the structure for updating entities, the attrib parameter references a list of attributes generated in method number 1. Method number 4 defines the assignment of values to entities, and can be used both in the creation and updating of entities.

```
#Structure JSON - PAYLOAD
def struct_list_atributes(tipo,valor):
    if tipo.lower() == "float" or tipo.lower() == "integer":
        list={"type":tipo,"value":valor}
    else:
        if tipo.lower() == "text":
            list={"type":tipo,"value":"{}".format(valor)}
    return list                                                    1

def struct_add_json(id,tipo,attrib,lista_attrib):
    datos = {
        "id":id,
        "type":tipo,
        attrib:lista_attrib
        }
    return datos                                                   2

def struct_patch_json(attrib,lista_attrib):
    datos = {
        attrib:lista_attrib
        }
    return datos                                                   3

def struct_value_json(value):
    datos = {
        "value":value
        }
    return datos                                                   4
############################################################
```

Figure 9: JSON structure implementation

3. Defining the basic functions under the FIWARE NGSI standard

✓ Consulting entities:

```
def findEntities(entidad):
    if entidad == "":
        r = requests.get("http://%s:%d/v2/entities" %(host,port))
    else:
        r = requests.get("http://%s:%d/v2/entities/%s" %(host,port,entidad))
    return r.json()
```

Figure 10: FIWARE NGSI – Query structure implementation

Figure 10 shows the method of consulting entities to the Orion Context Broker. In compliance with the FIWARE NGSI standard and depending on the value of the entity parameter, the execution of this method will return the list of all the entities stored or a specific entity:

The result of Figure 11 has the JSON format, widely known and implemented for the presentation of data in different user interfaces, such as web applications, mobile applications or desktop applications.

```
{'id': 'Animal', 'type': 'Domestico', 'Altura': {'type':
 'integer', 'value': 30.0}}
```

Figure 11: Querying stored entities

✓ Inserting entities:

Figure 12 shows the method of inserting entities into the Orion Context Broker. In compliance with the FIWARE NGSI standard, the execution of this method will return the result of the transaction using JSON response code. Code 201 will report successful registration, code 200 registration failed.

```
def addEntities(datos_json):
    r=requests.post("http://%s:%d/v2/entities" %(host,port),headers=headers,
        data=json.dumps(datos_json))
    print(r.text)
```

Figure 12: FIWARE NGSI - Insertion structure implementation

In Fig. 13, the addEntities method is executed and the entities are subsequently consulted to verify the correct insertion, although the printing of the code 201 is enough to demonstrate this.

```
if __name__ == '__main__':
    lista_attrib = struct_list_atributes("integer","400")
    addEntities(struct_add_json("Electrico","Nissan Zero","Watts",lista_attrib))
    print(findEntities("Electrico"))
```

Figure 13: Executing the addEntities method

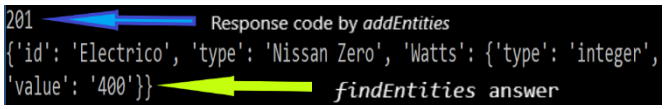

Figure 14: Insertion and response code.

Software validation tests were performed on a virtualized CenOS67 system on a 64-bit Windows 10 PRO operating system, with Intel Core I5-4210U CPU 2.4 GHz and 12 Gb in RAM. While the virtualized service was executed on VirtualBox Linux CentOS67 Red Hat 64 bits, with 2-core processor and 2 Gb in RAM.

CONCLUSIONS

The FIWARE platform was configured in the framework of the internet of things, the solution successfully integrated the context or entities management service; In a structured manner, the steps to be followed for configuring and staging the service are represented, showing several outstanding aspects and capabilities that can be expected from the platform.

All the initial steps are available for future research lines where it is necessary to deepen and complement the multiple functionalities of the FIWARE platform; leaving a bit of the educational context, this project can be taken as a guide for the implementation of much more ambitious commercial projects, related perhaps to the environment, transportation systems, both public and private, economic and / or governmental lines, among others.

One of the most complex phases of the present project was the understanding as such of the FIWARE platform and the configurations or enablers to be used according to our need to manage the information obtained by a sensor network. After carrying out the respective investigation and analysis of the information, the correct configuration and optimum functioning of the Orion Context Broker service was achieved. Once the service was established, the implementation of the appropriate standard for the management of data and communication towards the Orion Context Broker was carried out, supported by the extensive existing documentation of FIWARE NGSI V2.

ACKNOWLEDGMENT

This work was supported by the District University Francisco José de Caldas, Technological Faculty. The views expressed in this paper are not necessarily endorsed by District University. The authors thank the research group ARMOS for the evaluation carried out on prototypes of ideas and strategies.

REFERENCES

[1] Upadhyay, K., Singh, S. and Rawat, S.: IoT: Pillars and Technology. In: Indian Journal of Science and Technology, 9 (47), pp. 1-8 (2016)

[2] Amit, J., Gurpreet, S. and Gagandeep S.: Internet of Things: A Beginners' Précis and Future Scope. In: Indian Journal of Science and Technology, 9 (47), pp. 1-9 (2016)

[3] Rajalekshmi, R., Radhakrishnan, B. and Suresh, L.: Intelligent parking space detection and number plate extraction. In *2017 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pp. 1-6, (2017)

[4] Shanthi, G., Sundarambal, M. and Dhivyaa, M.: Design and Implementation of Monitoring and Control System Based on Wireless Sensor Networks for an Energy Conservation in Building. In: ARPN Journal of Engineering and Applied Sciences, 10 (1), pp. 207-212 (2015)

[5] Gunardi, Y., Adriansyah, A. and Anindhito, T.: Small smart community: an application of internet of things. In: ARPN Journal of Engineering and Applied Sciences, 10 (15), pp. 6341-6347 (2015)

[6] Abascal, A.: Platform integration for context management through the use of FIWARE Generic Enablers for the Internet of Things. Pregrado. Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicación, pp.16-18 (2015).

[7] Matínez, R., Pastor, J., Álvarez, B. and Iborra, A.: Diseño e implementación de un banco de pruebas para evaluar plataformas middleware de internet de las cosas: un estudio acerca de FI-WARE en el ámbito de la agricultura de precisión. Anuario de Jóvenes Investigadores, 9, pp.216-217 (2016).

[8] Cristescu, G.: Development of a Context Management Platform for the Internet of Things based on OMA NGSI-9 and NGSI-10 standards. Pregrado. Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicación, pp. 34-39 (2015).

[9] Catalogue.fiware.org.: Publish/Subscribe Context Broker - Orion Context Broker | FIWARE Catalogue. [online] Available at: https://catalogue.fiware.org/enablers/publishsubscribe-context-broker-orion-context-broker/downloads. (2017).

[10] Faludi, R.: Building Wireless Sensor Networks. O'Reilly Media. United States of America, (2011).