Received: 21 March 2017 | Revised: 20 May 2017 | Accepted: 14 June 2017

DOI: 10.1002/dac.3380

RESEARCH ARTICLE

WILEY

Middleware for integration of legacy electrical equipment into smart grid infrastructure using wireless sensor networks

Paulo Régis C. Araújo[1,2] | Raimir Holanda Filho[2] | Joel J. P. C. Rodrigues[1,3,4,5] | João P. C. M. Oliveira[2] | Stephanie A. Braga[2]

[1]University of Fortaleza (UNIFOR), Washington Soares avenue, 1321, Fortaleza, Ceará, Brazil

[2]Federal Institute of Ceará(IFCE), 13 de maio avenue, 2081, Fortaleza, Ceará, Brazil

[3]National Institute of Telecommunications (Inatel), Santa Rita do Sapucaí, Minas Gerais, Brazil

[4]Instituto de Telecomunicações, Universidade da Beira Interior, Covilhã, Portugal

[5]ITMO University 1, St. Petersburg, 197101, Russia

Correspondence
Joel Rodrigues, National Institute of Telecommunications (Inatel), Av. João de Camargo, 510, Centro, Santa Rita do Sapucaí, MG, Brazil.
Email: joeljr@ieee.org

Funding information
FCT—Fundação para a Ciência e a Tecnologia, Grant/Award Number: UID/EEA/500008/2013; Government of the Russian Federation, Grant/Award Number: 074-U01; Finep/Funttel, Grant/Award Number: 01.14.0231.00; CNPQ—Conselho Nacional de Desenvolvimento Científico e Tecnológico, Grant/Award Number: APQ/468898/2014-0; federal government of Brazil

Summary

To improve the efficiency of electricity distribution, smart grids allow communication between their devices. Pieces of legacy equipment operating in the distribution network do not communicate using any commercial protocol, such as DNP3, IEC 61850, or Modbus. Thus, herein, a middleware is proposed to allow the integration of the legacy electrical equipment into a smart grid using wireless sensor networks (WSNs). Each piece of legacy electrical equipment is connected to a sensor node, and the WSN sink node runs a middleware to enable the integration of this device with a smart grid, according to the commercial communication protocols. The middleware model is proposed to guide users in the development of a WSN-based system for integrating electrical equipment into a smart grid. The middleware was validated in a real environment, which is related to the concept of power metering. Experiments were performed using the software supervisory control and data acquisition and distributed test manager to validate the communication between the electrical equipment and the computer of the power substation control centre.

KEYWORDS

middleware, real deployment, smart grid, smart metering, wireless sensor networks

1 | INTRODUCTION

The monitoring and control activities in power substations are performed by the electrical system equipment and devices. Regarding the control and automation of these devices, there is a growing trend towards devices that have intelligence, which are called intelligent electronic devices (IEDs). According to the authors of Weerathunga and Cioraca[1] and Bi et al,[2] IEDs offer important characteristics for the smart grid, such as the possibility of communicating with supervisory control and data acquisition (SCADA) and control systems over an Internet Protocol (IP)-based network, the use of standard protocols for communication, and the improvement of the measurement, metering, monitoring, and control functionalities. The interconnection, communication, and control of these devices are embodied by the concept of the smart

grid, which is explained in other works.[3-6] In a smart grid, the power distribution is more efficient because the electrical devices must communicate with the operation and control centre to adjust their operations. The smart grid considered by the authors in Jain and Mishra[7] as the ′grid of tomorrow′ or the ′vision of future′ enables two-way communication, as well as the monitoring and/or control of the energy flow, consumer demand, voltage, etc. As many electrical devices may be located away from the power substation, wireless sensor networks (WSNs) provide an inexpensive and efficient way for them to communicate. Previous reports present expensive proposals for communication in the smart grid, such as Meloni and Atzori,[8] where satellite communication is used to integrate the smart-grid elements. The authors in Sayed et al[9] propose a hybrid communication system for the smart grid, which combines wireless communication with powerline communication.

When the WSN is used to communicate with the smart-grid elements, certain characteristics can be observed. For example, WSNs can be used in the traditional traffic applications to monitor traffic, analyse the traffic information at remote nodes at a given time, and then send the information back to a central gateway or sink.[10] In some WSN applications for the smart grid, it is not necessary to save energy, as proposed in other studies,[11-13] because the monitored devices are energised by the distribution network voltage, which can also energise the WSN sensor nodes. In a smart-grid application, communication must always occur between the equipment or IEDs and the control centre of the power substations. Thus, instead of using decentralised communication, such as D2D communication,[14] we choose to use centralised communication that is controlled by the power substation control centre computer.

The modernisation of power substations involves the adoption of new electrical devices that communicate using the commercial communication protocols. However, utility companies cannot dispose of their legacy equipment currently in operation. Some of these devices may be expensive or recent investments of the company. Thus, the utility must perform an integration such that the legacy electrical equipment can communicate with the smart grid. This integration can be considered as a WSN-based solution that enables the legacy electrical devices to communicate with the operation and control centre via a commercial communication protocol, such as Modbus, DNP3, or IEC 61850. In previous studies, WSNs have been used to integrate electrical devices or sensors into a smart grid without the use of a commercial protocol.[5,6,15,16] In other studies, no commercial protocol was used for the integration of the devices or sensors into a power system, but the devices satisfied some requirements of smart-grid application, such as the latency in communication.[17] Thus, we found a few studies wherein legacy equipment was integrated into an electrical systems using a commercial communication protocol. None of them involved integration via the 3 commercial protocols (Modbus, DNP3, and IEC 61850) used in smart grids.

As discussed in Joel,[18] there are middleware and frameworks for WSN-based integration with a standardised smart grid using IP-based communication. Accordingly, a middleware is proposed to solve the problems related to the integration of legacy electrical equipment into a smart grid via a standardised communication protocol (DNP3, Modbus, or IEC 61850). This middleware runs on the sink node of a ZigBee-based WSN, which operates on the IEEE 802.15.4 physical radio specification. The WSN sensor nodes, which are connected to the legacy devices, enable these devices to communicate with the power substation control centre (PSCC) through the sink node, which is wired (Ethernet) to the control centre computer. This middleware allows, as the main activity, the translation of the messages communicated between the PSCC and the electrical equipment of the smart grid. Unlike wireless video communications, such as those presented in Zhou et al,[19] where messages must be transmitted with high quality, the messages communicated between the PSCC and the remote electrical device need to be transmitted and received with reliability, which can be improved by using ZigBee pattern features and implementations in the application layer.

Additionally a middleware model is proposed to guide the development of a WSN-based system to integrate the electrical equipment into a smart grid. This model is innovative because it guides the users to develop WSN-based integration systems for a smart grid. It is based on the layers and modules and can be expanded or adapted for other purposes. As a case study, a real application is considered to evaluate the requirements of a smart electrical metering application. In Zhou et al,[20] the authors detailed an architecture model that enables wireless communication involving the home smart meters.

A review of the literature in the area of smart grids revealed no research into a middleware allowing a WSN to integrate legacy equipment into a smart grid using one of the 3 communication protocols: DNP3, IEC 61850, or Modbus. The main contributions of this study are as follows:

- The proposed solution enables electrical utilities to integrate their legacy equipment into an electrical distribution network that uses any of the 3 smart-grid standards (DNP3, IEC 61850, or Modbus). This integration allows interoperability between electrical equipment from different manufacturers, as well as a cost reduction in the modernisation of the substation plant.
- A middleware model is proposed for this integration to guide WSN-based system developers in integrating the electrical equipment into a smart grid. This model is structured into layers, modules, and their interfaces to facilitate

the understanding of the features and requirements that a middleware must have to enable this integration.

The rest of the paper is organised as follows. In Section 2, the infrastructure and the middleware model for the integration solution is described in detail. Section 3 describes the middleware implementation, and the experiments and results are presented in Section 4. Finally, Section 5 concludes the paper and proposes further research.

2 | INFRASTRUCTURE AND MIDDLEWARE MODEL FOR INTEGRATION

An infrastructure is proposed to validate the middleware and its characteristics. The suggested environment for validating the experiments is a power substation, where a WSN is added to allow communication between the electrical equipment and the PSCC. As an electrical device to be integrated, an electronic power meter can be used to communicate with the power substation computer via smart metering.[21]

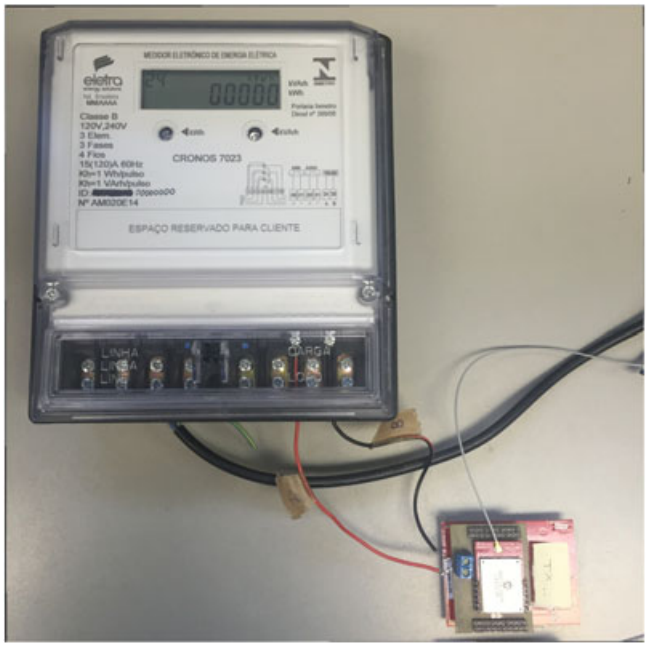

FIGURE 1 Power meter Cronos 7023 and sensor node

2.1 | Integration infrastructure components

The hypothetical infrastructure consists of the control centre computer of the power substation and a WSN. Regarding the architecture of the power substation, SCADA[22] runs on the control centre computer, and the middleware runs on the sink node of the WSN. The control centre computer, which runs a Linux OS and has an open-source license of ScadaBR,[23] is wired (Ethernet) to the WSN sink node, and the computer and the sink node are installed inside the power substation. The sensor node, which must be connected to the electrical device, can be installed outside the substation, for example, in the home of the customer in the case of a power meter.

As a case study, a scenario with the power meters connected to the sensor nodes to allow the remote reading of the costumer energy consumption is considered. A laboratory environment is used to validate the middleware. The validation scenario consists of a WSN with 3 sensor nodes and a sink node. Each sensor node, which must have a wired interface (using RS-485 or RS-232) with the electrical device, uses the ARM kit EKTM4C123GXL (Texas Instruments).[24] This connection allows the sensor node to receive the measurements from the device and send commands to operate it. For communication between the sink node and the sensor nodes, a 2.4-GHz IEEE 802.15.4 radiofrequency (RF) transceiver module called MRF24J40MC[25] is installed on the ARM kit. This RF transceiver module allows the sink node and the sensor nodes to exchange information using a ZigBee-based wireless communication. As shown in Figure 1, the sensor node is connected to an electronic power meter[26] (Cronos 7023) that allows the measurement of electrical quantities, such as the voltage and current per phase and the active and reactive electrical power. It uses the serial communication protocol ABNT NBR 14522, which uses the request/response model.[27] In this protocol, for every read request, there is a response of 258 octets, which is received 1 second after the corresponding request is sent. In this study, a periodic read request is performed on the meter every 5 seconds to satisfy the response time of Cronos 7023. Although the connection of these meters with the sensor nodes allows the monitoring functionalities of the distribution network,[28-30] our proposal uses this connection only for measuring the electrical power used by the consumers.

As shown in Figure 2, the sink node, which is represented by the Raspberry PI model B,[31] is wired (Ethernet) to the computer. This node runs a middleware that consists of modules that were developed using the Python language. In a previous study,[32] a middleware to be installed in the remote terminal unit (RTU) was developed, and its purpose was integrate electrical devices into a smart grid using WSNs.[32] Old or small power substations do not have a RTU, which motivated us to develop a novel middleware to be installed in the sink node of WSNs. The main advantages of this middleware over the previous ones are that it does not require a RTU to run and allows device integration using any of the 3 commercial communication protocols (DNP3, IEC 61850, and Modbus). The connection between the sink node and the computer allows the SCADA application to request the reading of a device variable and receive a response to that request.

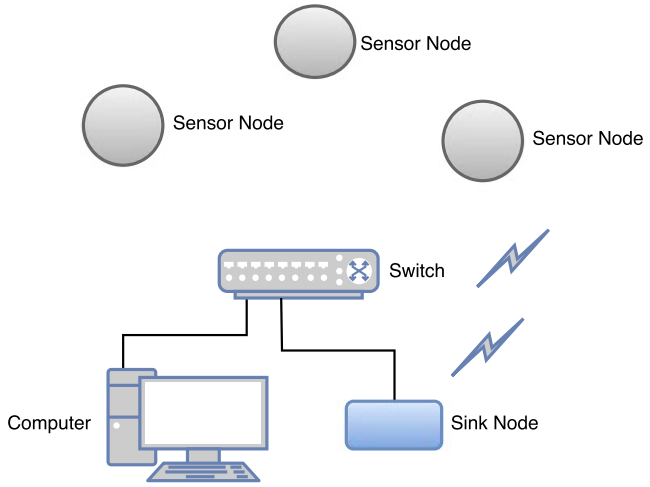

FIGURE 2 Wireless sensor network (WSN) topology and computer

Transmission Control Protocol (TCP)/IP in their communication. Thus, it is important that the middleware model has a TCP/IP protocol suite to allow communication between the smart grid and the equipment.

Conversion/Translation Layer—The legacy electrical equipment can be integrated to a smart grid that uses one of the following protocols: DNP3, IEC 61850, or Modbus. The model has a conversion suite that allows the translation of the message structure according to the specific protocol (DNP3, IEC 61850, or Modbus). After the TCP/IP stack is executed, the message payload is extracted, the protocol type is recognised by the TCP communication port, and the translation process converts the message structure of the specific protocol into the WSN message structure.

Data Storage Management—This module manages the data storage in the SCADA application request and its response. It

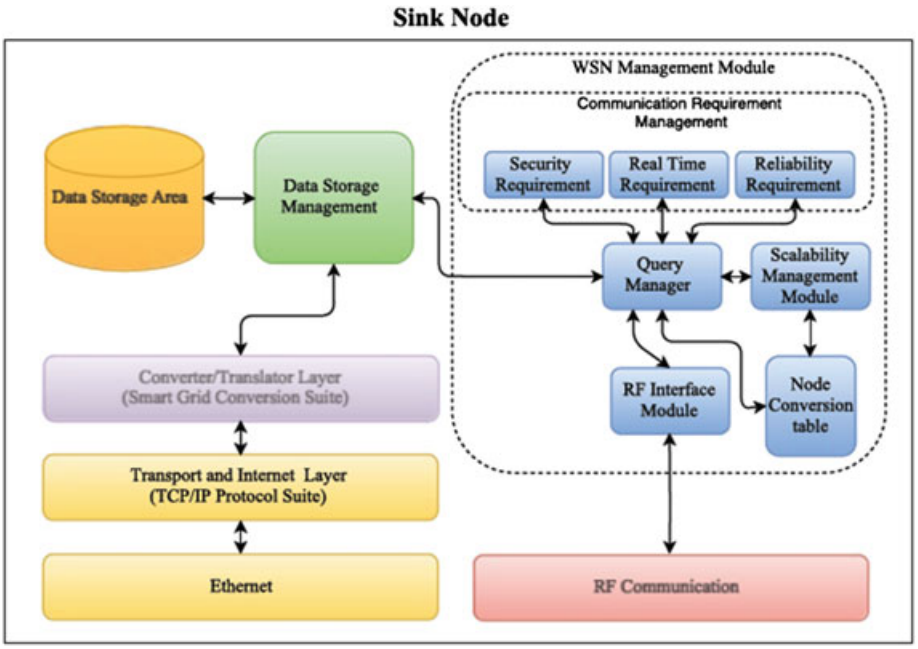

FIGURE 3 Model of a middleware for the wireless sensor network (WSN)-based integration of the electrical devices to a smart grid

A middleware model is proposed to guide users in the development of a WSN-based system for integrating electrical equipment into a smart grid. Figure 3 presents a flow diagram of the model, where the bidirectional arrows represent connection or interface between the modules. The modules are detailed as follows.

Ethernet—The electrical devices of the modern power substation are connected using local area networks. The Ethernet standard (IEEE 802.3) is one of the technologies used for communication between the control centre and the IEDs or RTU. Thus, a physical and data link layers should be included to allow the integration.

Transport and Internet Layer—Some protocols for smart grids, such as Modbus, DNP3, and IEC 61850, use

is responsible for receiving the converted message from the conversion layer. It extracts the fields of this message, which represent the data, and stores them in the data storage area. To maintain compatibility with the IEC 61850 standard, which uses an XML-based language called the System Configuration Description Language (SCL), the data can be stored in the XML format. After the message fields are stored, this module must inform the query manager of the arrival of a request or query. The storage of the response data must also be managed by this module.

Communication Requirement Management—The smart-grid monitoring and operation are critical applications, and the communication related to these functions must satisfy certain requirements. For communication between electrical

equipment and a smart grid, WSNs must exhibit characteristics such as security, a short response time, and reliability. The query manager uses the strategies implemented by the communication requirement management to allow communication with these characteristics.

Query Manager—This module is responsible for performing queries on the sensor network. It detects the arrival of a request, receives the data (message fields) from the data storage management module, packs these data to create the query message, and, by using the strategies of the communication requirement management module, performs the query.

Scalability Management—Electrical power utilities receive new consumers daily. This module allows a new sensor node (integrated into a piece of electrical equipment) to be automatically recognised and added to the WSNs. The new sensor node receives an identifier (ID), which is stored in the node conversion table. In the case of the ′IP network overlay sensor network′ approach,[33] this table can be used to perform conversion between the node virtual IP and the node ID.

RF Interface—This module operates the radio transceiver, including sending and receiving data and configuring the radio transceiver, among other functions. It receives the query packet and calls a sending function to send the packet to the destination node. Another important task performed by this module is detecting the arrival of the response packet and calling a receiving function to receive it.

This model is used as a reference for the implementation of the middleware for integrating legacy electrical equipment into a smart grid. This implementation is presented in the next section.

3 | MIDDLEWARE BASED ON WSN FOR INTEGRATING LEGACY ELECTRICAL EQUIPMENT INTO SMART GRID

The middleware presented herein represents a novel approach of the concentrator core that was developed in a previous research and published in Arajo et al.[32] Electrical power substations of small communities and old substations do not have an RTU, preventing the installation of the previous middleware.[32] Thus, a novel middleware based on the model presented in the previous section is proposed to be installed in the sink node.

As shown in Figure 4, this novel middleware consists of Python modules that run within the converter and WSN interface layers. The middleware modules, which are related to the model described in the previous section, are detailed as follows.

Master.py—This module starts the other Python modules and, regarding the middleware model, is the core of the ′WSN Management Module′. It has a Python class called *Management*, which has a method for initialising the modules *protocol.py*, *smartgrid.py*, and *interface.py*. It also has a method called *request()*, which is responsible for starting the query on the WSNs. Regarding the model, this method performs one of the tasks of the ′Query Manager′ module and uses the strategies of the ′Communication Requirement Management′ to satisfy the security, reliability, and response-time requirements for WSN communication. These strategies are not be presented herein, as they were discussed in a previous paper.[32]

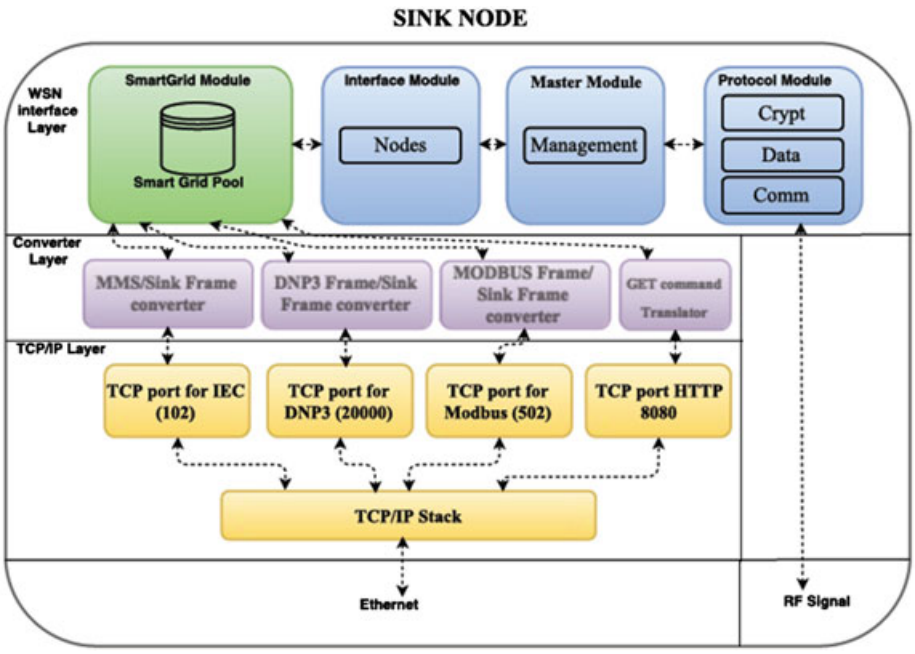

FIGURE 4 Middleware components within the sink node

Interface.py—This is another important module in the middleware. It has a class called *nodes*, and regarding the model presented previously, it has methods that represent tasks of the ′Query Manager′ module. This class has methods such as *request_lock()*, *request_release()*, *del_NODE()*, and *add_NODE()*. The methods *request_lock()* and *request_release()* function as a lock, and the queries performed via the SCADA application are prioritised for sending data to the sensor network. Thus, these methods block any command sent by IP access of the Web application, providing control because of SCADA. The method *add_NODE()* aims to initialise the node data and create a new alias for the network interface, thus creating a unique virtual IP for the sensor node. The unique IP, the ID of the node, and its basic information are stored in a routing and control table. The method *del_NODE()* performs the reverse operation of *add_NODE()*. It removes all sensor node information from the routing and control table. The method *add_NODE()* allows the integration of a sensor node in the network without the manual configuration of this node.

Protocol.py—The module *protocol.py* is basically the WSN communication protocol. It performs activities that are related to the tasks executed by the ′Query Manager′, ′RF Interface Module′, and ′Communication Requirement Management′ modules of the middleware model. It has the following classes: *crypt*, *data*, and *comm*. The *crypt* class is responsible for data encryption. It comprises the methods *encrypt()*, *decrypt()*, *crypt()*, *setHASH()*, *getHASH()*, and *getNEXThash()*, which are used to encrypt and decrypt the data to be communicated in the WSN, as well as to create, configure, and change the encryption keys. In the middleware model, this class is related to the ′Communication Requirement Management′ module. The class *data* basically consists of the methods *encode()* and *decode()*. These methods are intended to convert the default message structure used by the middleware to the query packet structure used by the WSNs. *Encode()* uses the method *pack()* to make the WSN query packet and *decode()* uses the method *unpack()* to extract the fields of the WSN responses. Message redundancy techniques (sending and checking CRC) are used by the methods *encode() and decode()* to allow greater reliability in the WSN communication. The class *data* are strongly related to the activities performed by the ′Query Manager′ module of the middleware model. The class *comm* is responsible for communication between the Raspberry PI and the radio transceiver. It consists of methods such as *init(), close(), cleanSerial(), send(), and receive()*, which are intended to set the parameters of the radio, close the serial connection, clean the serial buffer, send data, and receive data, respectively. According to the middleware model, the methods of the class *comm* perform tasks related to the ′RF Interface Module′. This module and the previous ones are beyond the scope of the present work, as they were selected to be presented in a future paper.

Smartgrid.py—The module *smartgrid.py* creates a link between the SCADA application and the sensor nodes. This module has a class called *smartpoll*, which has the methods *waitRequest(), sendResponse(), and commInterface()*. The method *waitRequest()* waits for a request from the converter layer and receives it. Then, the method *commInterface()* is invoked to send the request to the module *Interface.py*. The module *smartgrid.py* has a poll (buffer) to store data received (responses) from the sensor nodes. These stored data can be used as a response for the SCADA application. The method *sendResponse()* is invoked to send the response stored in the poll to the converter layer.

Converter Layer—To receive requests from the TCP/IP layer, the converter modules in the converter layer create new servers on the ports related to the standard protocols: DNP3, IEC 61850, and Modbus. For example, if the query is performed by the SCADA application using the Modbus communication protocol, the module ′MODBUS frame/sink frame converter′ creates a new server on port 502 to receive the commands from SCADA. According to the TCP port that receives the request, the server associated to it is required.

The module *DNP3conv.py* (DNP3 Frame/Sink Frame Converter) has a class called *DNP3Server* with 4 methods: *startServer(), linkLayer(), transportLayer(), and AppLayer()*. The method *startServer()* instantiates the server and receives the request parameters of the SCADA application using the DNP3 protocol. The method *linkLayer()* extracts the fields of the DNP3 frame header and then breaks and reassembles the frames to compose the fragments. The method *transportLayer()* breaks and reassembles the fragments to compose the message. The method *AppLayer()* maps the static data of the request to the electrical parameters recognised by the middleware and maps the point indeces of the request to the sensor node IDs.

MODBUSconv.py (MODBUS frame/sink frame converter) is a module composed of 5 methods: *init(), setValues(), StartTcpServer(), getALLdata(), and getSINGLEdata()*. These methods perform several tasks, eg, instantiate the server and receive the request parameters, extract the fields of the Modbus Application Protocol header, convert the slave address to the sensor node ID, extract the function code and convert it into the query parameter, and break and reassemble the Modbus TCP message.

The module *IECconv.py* (Manufacturing Message Specification [MMS] frame/sink frame converter) comprises some classes, among which the most important is *IedServer*, which has the methods *IedServer_create()*, *IedServer_start()*, *IedServer_stop()*, and *IedServer_destroy()*, among others. These methods allow us to create an IEC 61850 server application and manage the complete MMS protocol stack and the IEC 61850 data model. Additionally, they start the protocol

stack, listen for client connections, feed process values to the MMS server, react to client activities, stop the MMS server, and free all the resources allocated by the IedServer instance. The class *IedServer* also has methods for feeding process values, ie, handling process values of a client application for the MMS server. These methods are *IedServer_lockDataModel()*, *IedServer_updateAttributeValue()*, and *IedServer_unlockDataModel()*.

3.1 | Mapping messages of commercial protocols and WSN

The converter layer receives the message via the commercial smart-grid protocols and translates it into a message structure recognised by the WSN. In the opposite case, it translates the response from the WSN into a message recognised by the protocol that made the request. A mapping of the protocols is proposed to perform these translations.

3.1.1 | Mapping Modbus and WSN messages

The MODBUS protocol has a message structure defined by a simple protocol data unit (PDU) consisting of 2 fields: the data and a function code. Fields can be added to the PDU frame to create an application data unit. The Modbus protocol presents a data model based on 4 data types: discrete input, coils, input registers, and holding registers.[34] The main validation of this work is the remote reading of the power-meter voltage. The data type ′holding register′ is chosen because its size is a 16-bit word and it allows the data to be changed by an application program. To facilitate the mapping, we organise the stored data into blocks. Thus, the smart-grid poll has a memory block to store the holding registers, as shown in Figure 5.

Modbus requests made by the SCADA software are identified by the converter layer, which is running Modbus service. This service extracts the important fields from the request, such as the function code, the initial address, and the number of registers, and stores them in the smart-grid poll. The module *smartgrid.py* has a method called *waitRequest()* that identifies the action related to the function code. As the action is a read, the method *waitRequest()* takes the data stored in the WSN response buffer for Modbus and stores them according to the initial address of the holding registers area. In this study, to reduce the response time in the request of a power meter reading, a periodic query (each 5 min) is performed in the WSN. Periodically, the method *Comm()* is invoked by the method *request()* (module *Master.py*) to perform a query on all the nodes of the WSN. Thus, when the SCADA application performs a request on the network, the readings related to the power meters are already stored in the WSN response buffer. As shown in Figure 5, the WSN message frame is represented by the following octet blocks: ′node identifier′, which has 4 octets and represents the target sensor node; ′action′, which has one octet and represents the request and response commands; ′data′, which has 12 octets and stores the electrical quantities, with each quantity represented by 4 octets; and ′error check′, which has 2 octets. The ′data′ block has 12 octets because each request can return 3 electrical quantities related to phases A, B, and C.

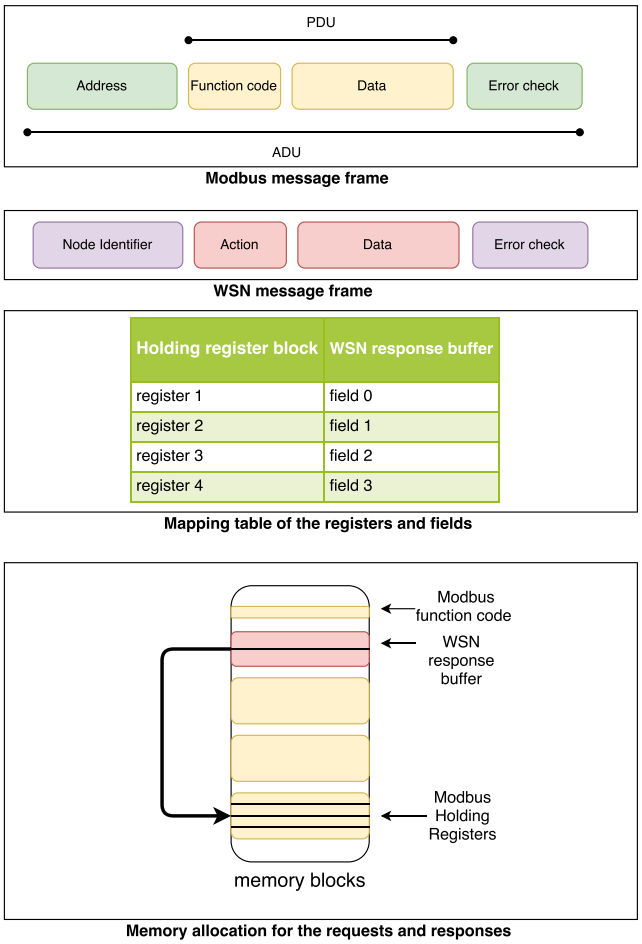

FIGURE 5 Mapping the message frames of the protocols

3.1.2 | Mapping DNP3 and WSN messages

The DNP3 protocol uses master/outstation communication, where the master sends a request to the outstation, which must respond.[35] The outstation can be represented by an IED, which controls or monitors electrical devices. For example, a potential transformer (PT) can be connected to the IED by a point (instance of a point type) of the type ′analog input′. Inputs are values that are measured, read, or generated by the electrical device. Some PTs can be connected to the same IED, which consists of a point array of the same type. Each point of this array has an index number called the ′point index′, which is a zero-based numeric ID that differentiates unique instances of points having the same point type.[35] Each message communicated between the master and

the outstation must have a 'group', which classifies the type of data in the message. For example, the request to read the PT voltage at a given time is configured by group 30 (current value of the point). Finally, the variation informs the encoding format for many of the data types.[35] Using the previous example, the voltage value generated by the PT can be represented by variation 5, which has a format of a 32-bit floating-point value with flag. After certain concepts of the DNP3 protocol are explained, the memory mapping will be presented.

The DNP3 service, which runs in the converter layer, constantly listens on TCP port 20000 for a request. When the SCADA software performs a DNP3 request for the voltage reading of an electronic meter, which is connected to a sensor node, this service receives the DNP3 data frames, assembles the segments, and traces the DNP3 fragment. This request fragment is stored in the smart-grid poll. Because the validation of the middleware is related to the power-meter voltage readings, we select the following DNP3 data characteristics: point type as the analog input, a point index starting at 0, a group type of 30 (current point value), and a variation of 5 (32-bit floating-point value with flag). Figure 6 shows the DNP3 fragment structure, the mapping table with the data of the DNP3 memory block and the WSN response buffer for DNP3, and the memory allocation and copy for the requests and responses.

After receiving a request, the DNP3 service stores the fragment in the smart-grid poll and notifies the module *smartgrid.py* that a request has arrived. This module has a class called *waitRequest()* that extracts the second octet of the application header. This octet contains the function code related to the action to be performed at the point (electronic meter). In this validation, the action has been set for reading whose function code is 0x01. The method *waitRequest()* extracts the object header octets, whose fields contain information regarding the object type (0x1E - group 30, 0x05 - variation 5), object prefix code (0x04 - objects are prefixed with object size), and range specifier code (0x07 - range field contains 1-octet count of objects). Then, method *waitRequest()* assembles the response in the WSN response buffer for DNP3 as follows: It repeats the first octet of the application header; stores the value 0x81 in the second octet (solicited response function code); sets the bits of third and fourth octets, which are related to the internal indications of the response; repeats the octets of the object header; stores the value 0x04 in the octet related to the object prefix (data size: 32-bit floating-point); and finally stores the 32-bit data value in the last 4 octets. After mounting the response, the method *waitRequest()* performs a memory copy of the WSN response buffer for the DNP3 memory block

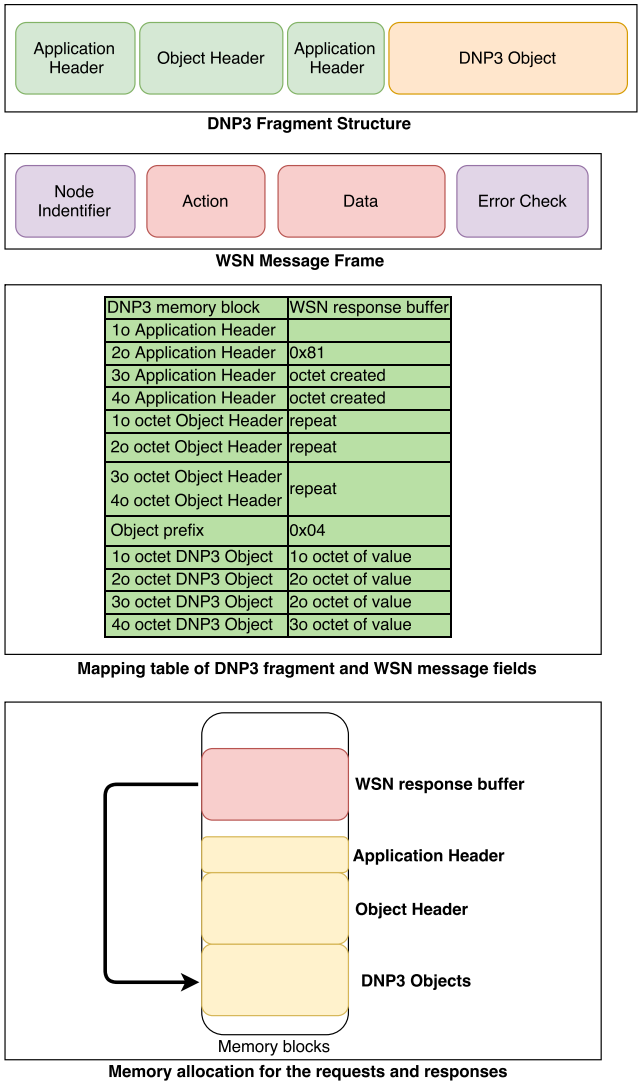

FIGURE 6 Mapping the DNP3 fragments and the wireless sensor network (WSN) messages

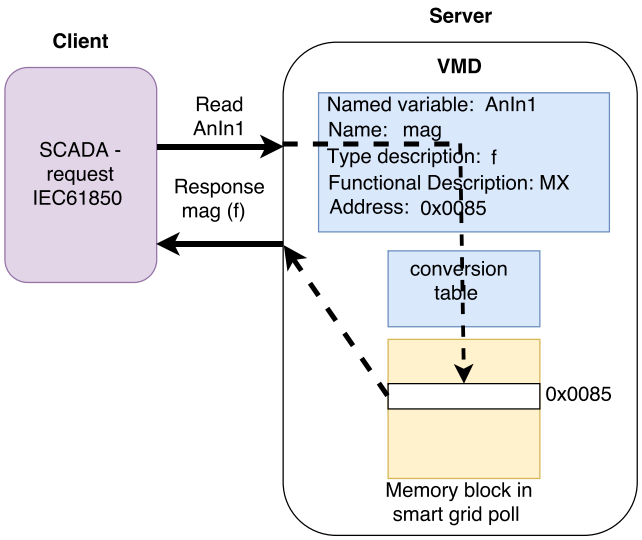

FIGURE 7 Mapping the IEC 61850 objects and the wireless sensor network (WSN) messages

and notifies the DNP3 service that the response fragment is ready.

3.1.3 | Mapping IEC 61850 and WSN messages

In the converter layer, the IEC 61850 conversion uses the MMS protocol in the client/server model. The client is the computer running SCADA or another supervisory software, and the server is the hardware device (electronic meter) or its control system. This MMS server must offer the client services such as read, write, and report management. The sink node has a Virtual Manufacturing Device (VMD) object that functions as a container in which all other objects are stored. Within it, the *named variable* object is defined. This object represents the voltage value of the electronic meter. Its key attribute is the *variable name*, and it has other attributes, such as the *type description*, *access method*, *functional constraint*, and *address*. To validate this work, we use the following objects and attributes: *GGIO* as the VMD object, which is the container of other objects; *AnIn* is the *named variable* object, which is related to the voltage value of the meter; *mag*, which is the *variable name* attribute; f (floating point), which represents the *type description* attribute; and MX, which is a *functional constraint* attribute that represents the measurement functionality. Additionally, the *access method* attribute is configured as *public*, allowing the read service to access the object attributes such as *address*, which contains the variable address in the smart-grid poll.

As shown in Figure 7, a read request performed by SCADA using the IEC 61850 protocol sends a PDU that contains information about the ′AnIn1′ object to the server. This activity is represented by the line ′Read AnIn1′. The converter layer (listening port 102) receives this data, and the MMS protocol extracts the attribute fields, such as the variable name. Next, the MMS service uses a conversion table to find the address of the electronic meter voltage in the smart-grid poll, by using the variable name. Then, a response PDU containing the power-meter voltage value is assembled and sent to the client. The response PDU sent by the server to SCADA is represented by the line labelled ′Response mag′ in Figure 7. In this figure, the nondotted lines represent the PDUs sent between SCADA and the server, and the dotted lines represent the sequence of activities performed by the server to convert a read object to a value of the electrical magnitude to be returned to SCADA.

4 | ANALYSIS OF RESULTS

As mentioned in the previous section, the test scenario consists of a WSN with 3 sensor nodes and a sink node, and the sink node is connected to a computer using Ethernet. Once this setup was assembled, a test was performed to validate the converter layer of the middleware, ie, to determine whether requests made by ScadaBR or a free version of the Distributed Test Manager (DTM) software[36] using any of the 3 protocols (Modbus, DNP3, or IEC 61850) were answered.

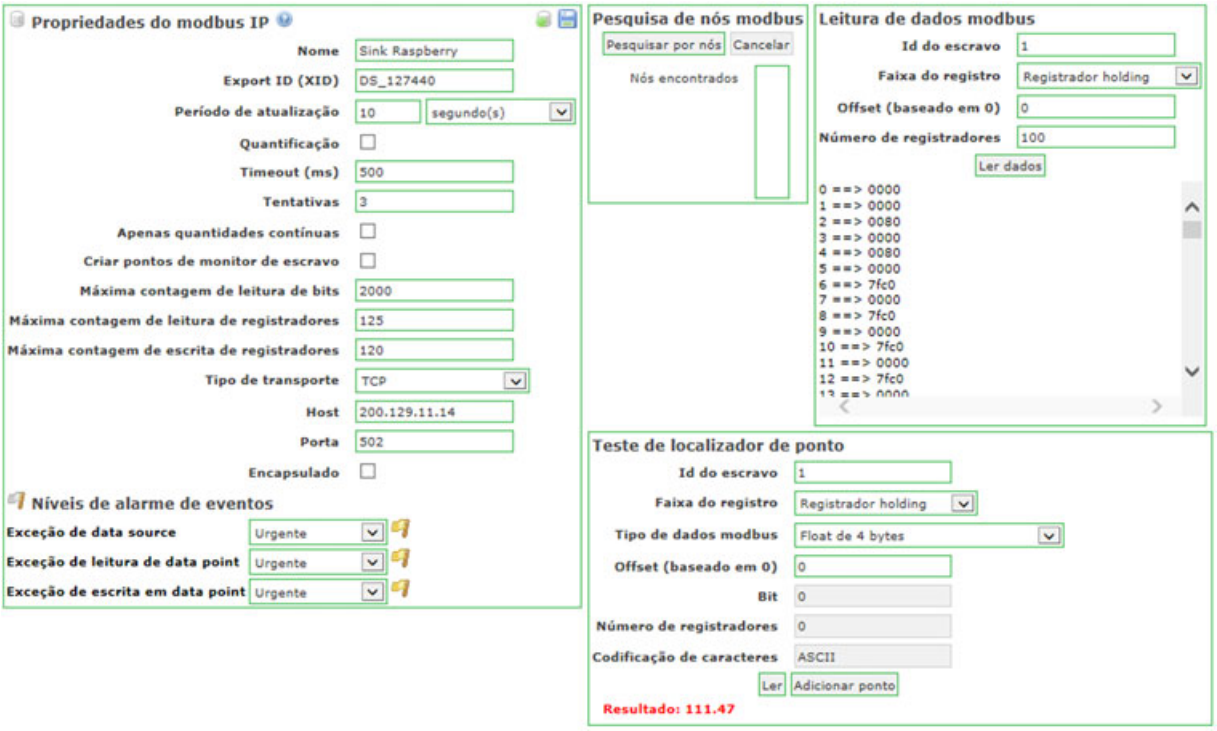

FIGURE 8 Screenshot of ScadaBR software using MODBUS protocol

Figure 8 shows a screenshot of the ScadaBR software. It is a supervisory software with an open-source license. This screenshot shows a response to a request made by the ScadaBR software using the Modbus protocol. The left window on the screen presents information regarding the Modbus IP properties, such as the time period of the reading updating, which was set as 10 seconds; the number of retries in the timeout exception, which was set as 3; the timeout value, which was set as 500; the type of the transport layer, which was set as TCP; the Modbus port, which was 502 by default; and the IP address of 200.129.11.14, which was configured for the sink node. Importantly, the TCP port was configured for port 502, which is the TCP port used for the Modbus protocol. The right-side window, which is called the ′point locator test′ presents information related to the characteristics of the point to be queried, such as the Modbus data type (4-byte float), register type (holding register), number of registers (related to the data type), and character encoding. This window also contains a command button called ′read′ for performing queries on the target sensor node (target point). When this button is clicked, a query is performed on the sensor node, and a result is returned (red-colour string). In this test, the obtained result

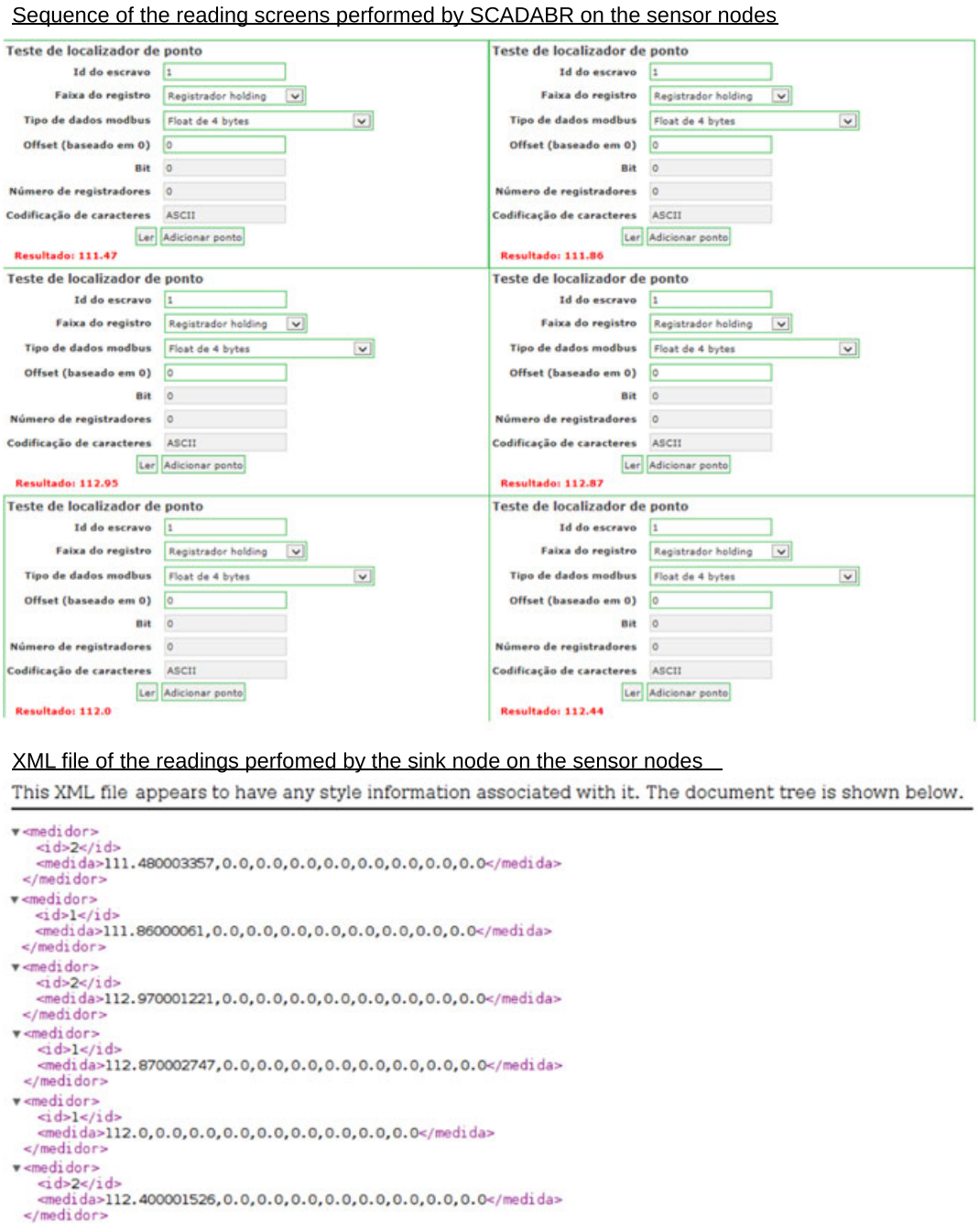

FIGURE 9 Comparison between the readings performed by ScadaBR and the sink node on the wireless sensor network (WSN)

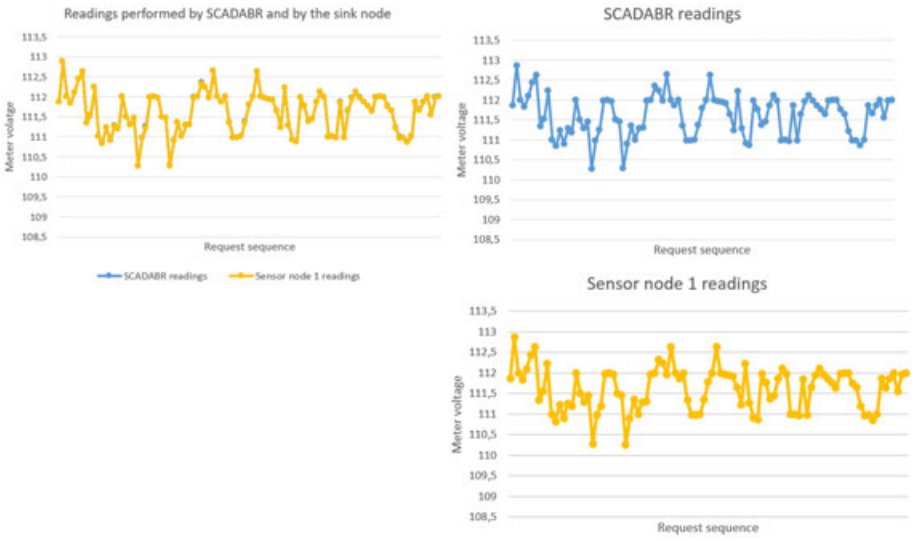

FIGURE 10 Comparison between the readings performed by ScadaBR and the sink node on the wireless sensor network (WSN)

was the phase voltage (111.47 V), which was measured by the power meter connected to the sensor node.

In Figure 9, the upper part presents screens of the voltage readings performed by the ScadaBR software on the WSN sensor nodes. The reading action was performed by clicking the button ′Ler′ (read). The lower part of Figure 9 presents the voltage readings performed by the sink node using a Python program. For each voltage-reading request of the ScadaBR software that arrives at the sink node, the program records the response to that request in an XML file. Thus, before the middleware sends the response to the ScadaBR software, this value is stored in a file. This experiment was performed to compare the values received by ScadaBR with the values obtained directly from the sensor nodes by the sink node. This comparison, which is detailed by the graphs shown in Figure 10, allows us to evaluate whether the converter layer correctly converts the voltage value read from the meter to the value of the PDU data field.

In Figure 10, the upper left-hand chart presents a sample of 100 voltage readings performed by ScadaBR and by the sink node on sensor node 1. Because the values are almost equal, the difference between the 2 reading graphs is negligible. Thus, 2 graphs are presented on the right side of the figure to show the similarity of the values of the 2 reading sequences. A few values of the sample exhibited differences on the order of hundredths or thousandths, which are related to the value rounding of ScadaBR. Similar results were obtained for sensor nodes 2 and 3.

At the top of Figure 11, a screenshot of the DNP3 master configuration designed using a free version of the DTM software is shown. We used this software because ScadaBR does not implement the DNP3 protocol. The DNP3 channel editor is shown on the right side of the screenshot. The computer running DTM was configured as the master (behaviour); the connection was TCP/IP (connection type); the DTM request application was configured as the client (mode); the remote address was the IP address of the sink node (200.129.11.14); and the port was 20000, which is the standard TCP port for DNP3. At the bottom of Figure 11, the DNP3 response received by the DTM client is shown. The response on line 11733 represents the parameters of the application response header, which has an object type of 30 (current value of the point); a point type of analog input; and a variation of 5, which represents a floating-point value. On line 11744, the voltage value of the electronic meter is presented. This value was measured by an electronic meter connected to a voltage stabiliser with a standard output of 110 V.

Figure 12 presents a screenshot of the IEC 61850 client configuration designed using a free version of the DTM software. We used this software because ScadaBR does not implement the IEC 61850 protocol. The IEC 61850 channel editor is shown on the right side of the screenshot. The computer running DTM was configured as the client; the connection was TCP/IP; the server address was the IP address of the sink node (200.129.11.14); and the port was 102, which is the standard TCP port for the IEC 61850 service.

Figure 13 shows a screenshot of the confirmation of the DTM request, which was performed using the IEC 61850 protocol and captured by the pcap (packet capture) application. This screenshot was selected because it clearly shows the service that was used in the request (read service) and the data object implemented for the measurement unit. In Figure 13, the first orange row contains information such as the type of the message protocol (MMS), the IP address of the destination node (virtual IP of the WSN), and the message information (confirmation of a request). Information regarding the source TCP port is presented in the line called ′Transmission Control Protocol′. This port was set to

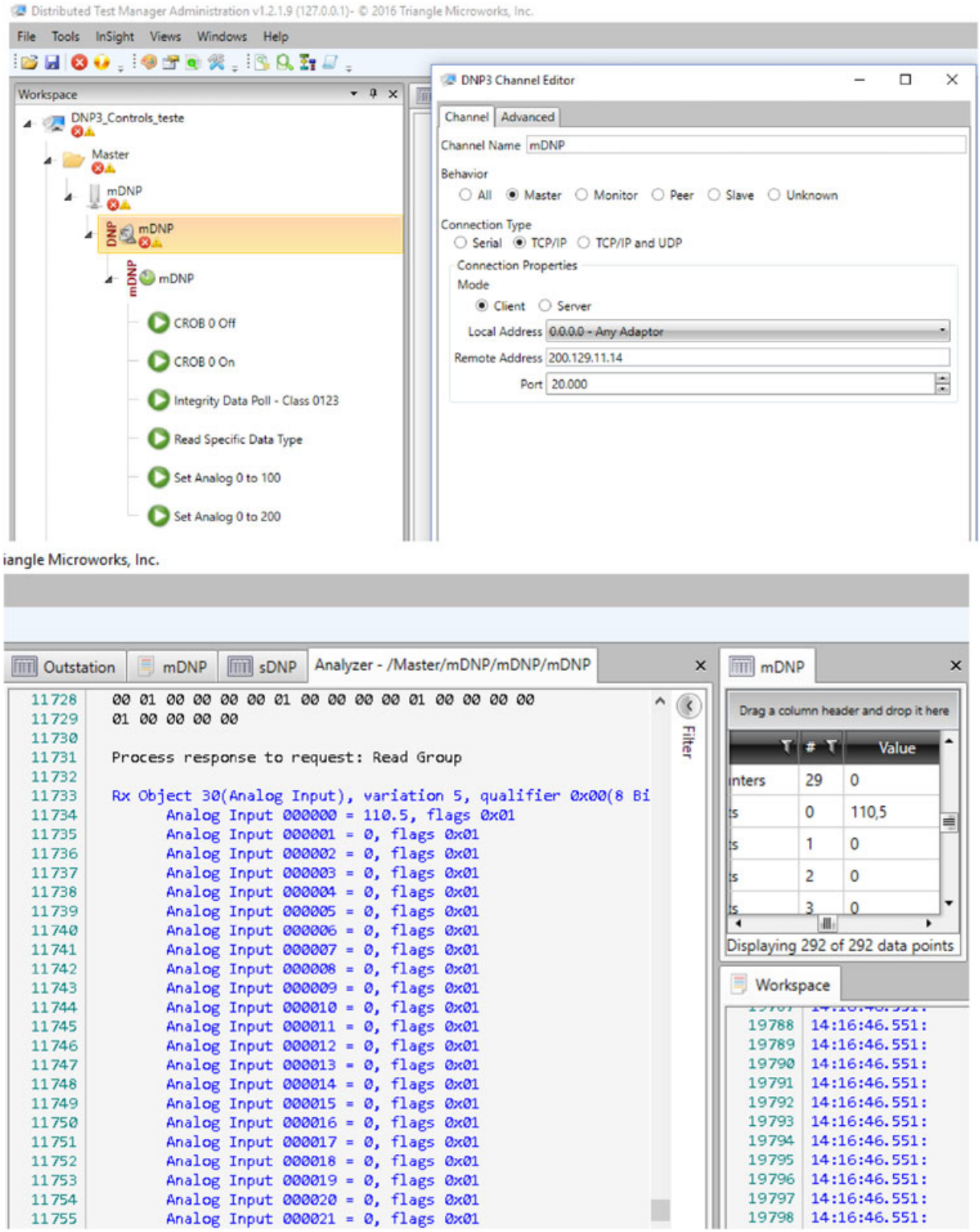

FIGURE 11 Screenshot of the DNP3 master configuration using Distributed Test Manager (DTM) software

102, which is the standard TCP port for IEC 61850/TCP communication. In the IEC 61850 protocol, the measurement information of an electrical device is represented by a data object, and its anatomy is indicated by the ′idemID′ of the ′domain-specific′ item. In the ′idemID′, the expression ′f′ denotes the *type description* attribute, which is floating-point. ′mag′ is the *variable-name* attribute for the phase voltage magnitude. ′AnIn′ represents the *named variable* object, which is related to the voltage value of the meter, and ′MX′ is the *functional restriction* attribute. Finally, the expression ′GGIO′ is related to the logical node and represents the VMD object.

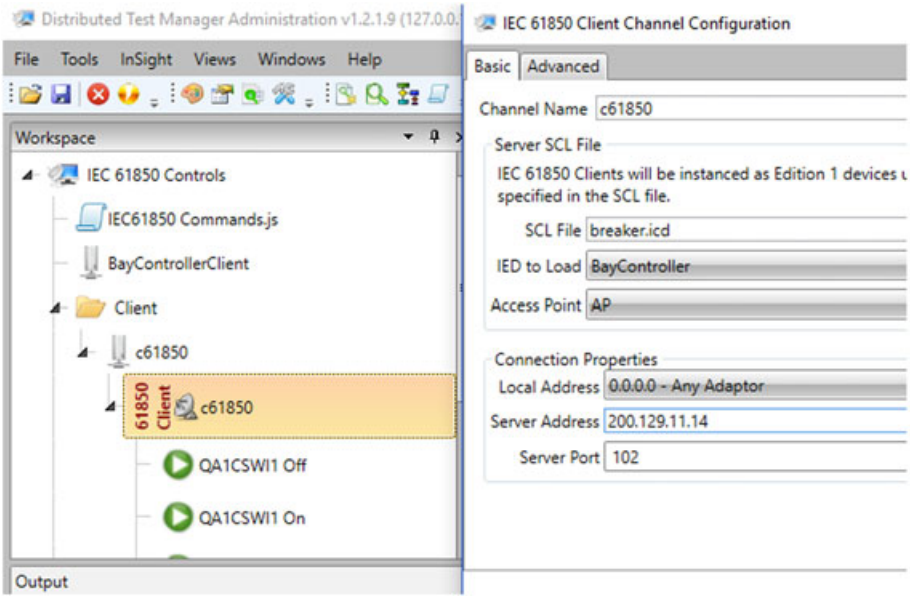

FIGURE 12 Screenshot of the IEC 61850 client configuration using Distributed Test Manager (DTM) software

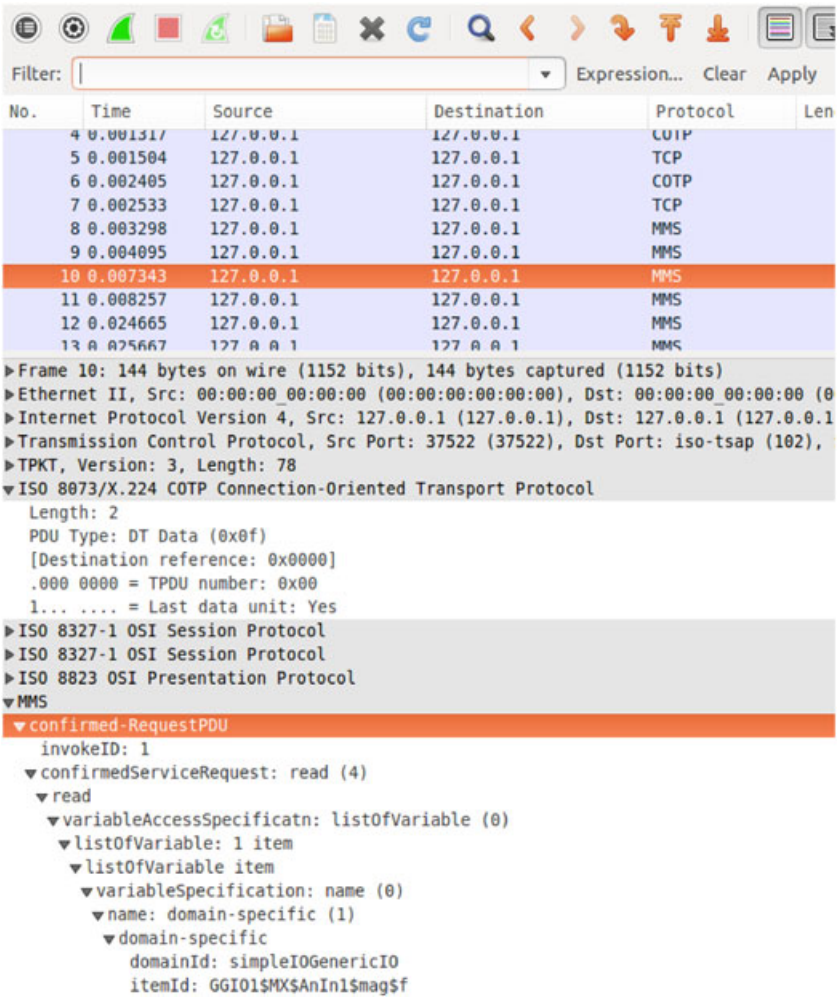

FIGURE 13 Screenshot of the confirmed IEC 61850 request captured by the pcap (packet capture) application

The aforementioned examples show that the proposed middleware can perform queries on the WSN by using 1 of the 3 protocols for smart grids: DNP3, IEC 61850, or MODBUS. Tests demonstrated that the converter layer is capable of translating requests performed by the SCADA or DTM software, which run in a control centre computer, using one of these protocols.

5 | CONCLUSION AND FUTURE WORK

A middleware that runs on the sink node of a ZigBee-based WSN is proposed for solving problems related to the integration of legacy electrical equipment into a smart grid that uses a standardised communication protocol (DNP3, Modbus, or IEC 61850). This middleware allows, as its main feature, the translation of the messages communicated between the PSCC and the electrical equipment of the smart grid.

A middleware model was proposed to guide WSN-based system developers in integrating devices or electrical equipment into the smart grid. This model was divided into layers, each represented by modules and their interfaces. Using this model, all the features, requirements, and interactions needed by the middleware to allow the integration were defined.

Because the converter layer of the middleware must receive the message via the commercial protocols for smart grid and translate it into a structure recognised by the WSN, a memory mapping was proposed to facilitate this translation. This mapping is based on an address relationship in the memory area called the smart-grid poll.

The results presented in Section 5 show that the infrastructure allows the monitoring of the electrical equipment using commercial standards for smart grids, such as DNP3, Modbus, and IEC 61850.

In future work, to improve this solution, the characteristics of the WSN such as the scalability, response time, and reliability will be evaluated via simulations using the Castalia software. Additionally, experiments will be performed in a real environment using a network with many electronic meters to validate the middleware.

ACKNOWLEDGEMENTS

This work was partially supported by National Funding from the FCT—Fundação para a Ciência e a Tecnologia through the UID/EEA/500008/2013 Project, by the Government of the Russian Federation, grant 074-U01, and by Finep, with resources from Funttel, grant no. 01.14.0231.00, under the *Centro de Referência em Radiocomunicações*—CRR project of the *Instituto Nacional de Telecomunicações* (Inatel), Brazil. It was also partially supported by National Funding from the CNPQ—Conselho Nacional de Desenvolvimento Científico e Tecnológico through the APQ/468898/2014-0 project, by the federal government of Brazil.

ORCID

Joel J. P. C. Rodrigues http://orcid.org/0000-0001-8657-3800

REFERENCES

1. Weerathunga EP, Cioraca A. The importance of testing smart grid IEDs against security vulnerabilities. In: Paper presented at: 69th Annual Conference for Protective Relay Enginers (CPRE). College Station: Texas, USA; 2016:1-21.
2. Bi YB, Jiang L, Wang XJ, Cui LZ. Mapping of IEC 61850 to data distribute service for digital substation communication. In: Paper presented at: IEEE Power and Energy Society General Meeting. Vancouver: British Columbia, Canadá; 2013.
3. Zhang F, Bhatt N. Next-generation monitoring, analysis, and control for the future smart control center. *IEEE Trans Smart Grid*. 2010;1(2):186-192.
4. Agrawal S, Das M. Internet of Things: a paradigm shift of future internet applications. In: Paper presented at: Nirma University International Conference on Engineering. Ahmedabad, Gujarat, India; 2011.
5. Abid M, Khallaayoun A, Harroud H, Lghoul R, Boulmalf M, Benhaddou D. A wireless mesh architecture for the advanced metering infrastructure in residential smart grids. In: Paper presented at: IEEE Green Technologies Conference. Denver, CO, USA; 2013.
6. Castellani A, Ministeri G, Rotoloni M, Vangelista L, Zorzi M. Interoperable and globally interconnected smart grid using ipv6 and 6lowpan. In: Paper presented at: 3rd IEEE International Workshop on SmArt COmmunications in NEtwork Technologies. Ottawa, ON, Canada; 2012.
7. Jain A, Mishra R. Changes & challenges in smart grid towards smarter grid. In: Paper presented at: International Conference on Electrical Power and Energy Systems (ICEPES). Near Mata Mandir, Bhopal, India; 2016.
8. Meloni A, Atzori L. The role of satellite communications in the smart grid. *IEEE Wirel Commun*. 2017;24(2):50-56.
9. Sayed M, Tsiftsis TA, Al-Dhahir N. On the diversity of hybrid narrowband-PLC/wireless communications for smart grids. *IEEE Trans Wirel Commun*. 2017;PP(99):1-1. https://doi.org/10.1109/TWC.2017.2697384. IEEE early access articles.
10. Zhu C, Shu L, Hara T, Wang L, Nishio S, Yang LT. A survey on communication and data management issues in mobile sensor networks. *Wirel Commun Mob Comput*. January 2014;14(1):1936.
11. Zhu C, Yang LT, Shu L, Leung VCM, Rodrigues JJPC, Wang L. Sleep scheduling for geographic routing in Duty-Cycled mobile sensor networks. *IEEE Trans Ind Electron*. November 2014;61(11):6346-6355.
12. Zhu C, Yang LT, Shu L, Leung VCM, Hara T, Nishio S. Insights of top-k query in duty-cycled wireless sensor networks. *IEEE Trans Ind Electron*. February 2015;62(2):1317-1328.
13. Zhou L, Hu RQ, Qian Y, Chen H-H. Energy-spectrum efficiency tradeoff for video streaming over mobile ad hoc networks. *IEEE J Sel Areas Commun*. May 2013;31(5):981-991.
14. Zhou L. Mobile device-to-device video distribution: theory and application. *ACM Trans Multimed Comput Commun Appl*. 2015;12(3):1253-1271.

15. Al-Anbagi I, Erol-Kantarci M, Mouftah H. A delay mitigation scheme for wsn-based smart grid substation monitoring. In: Paper presented at: 9th International Wireless Communications and Mobile Computing Conference (IWCMC), Sardinia, Italy; 2013.
16. Azzara A, Bocchino S, Pagano P, Pellerano G, Petracca M. Middleware solutions in WSN: the IoT oriented approach in the ICSI project. In: Paper presented at: 21st International Conference on Software Telecommunications and Computer Networks SoftCOM. Split-Primosten, Croatia; 2013.
17. Ferrari P, Flammini A, Rinaldi S, Sisinni E, Vezzoli A. Toward smart metering application exploiting ipv6 over wm-bus. In: Paper presented at: IEEE Workshop on Environmental Energy and Structural Monitoring Systems. Trento, Italy; 2013.
18. Joel H. Using a 6lowpan smart meter mesh network for event-driven monitoring of power quality. In: Paper presented at: IEEE SmartGridComm Symposium. Tainan City, Taiwan; 2012.
19. Zhou L, Yang Z, Wang H, Guizani M. Impact of execution time on adaptive wireless video scheduling. *IEEE J Sel Areas Commun*. 2014;32(4):760-772.
20. Zhou L, Rodrigues JJ, Oliveira L. Qoe-driven power scheduling in smart grid: architecture, strategy, and methodology. *IEEE Commun Mag*. 2012;50(5):136-141.
21. Popovic Z, Cackovic V. Advanced metering infrastructure in the context of smart grids. In: Energycon. Cavtat, Croatia; 2014.
22. B.S. Inc. Beyond SCADA. http://scada.com; 2016.
23. M. Sistemas. ScadaBR. http://www.scadabr.com.br; 2017.
24. Texas Instruments Org. Arm cortex-m4f based mcu tm4c123g. http://www.ti.com/tool/ek-tm4c123gxl; 2016.
25. Microchip Org. Mrf24j40mc data sheet. http://ww1.microchip.com/downloads/en/DeviceDoc/75002A.pdf; 2016.
26. Eletra Energy Solutions. Cronos 7023. http://www.eletraenergy.com/index.php?q=node/27; 2015.
27. Associação Brasileira de Normas Técnicas. Intercâmbio de informações para sistemas de medição de energia elétrica. http://www.abntcatalogo.com.br/norma.aspx?ID=822; 2008.
28. Bhela S, Kekatos V, Veeramachaneni S. Enhancing observability in distribution grids using smart meter data. *IEEE Trans Smart Grid*. 2017;PP(99):1-1. https://doi.org/10.1109/TSG.2017.2699939. IEEE early access articles.
29. Moghaddass R, Wang J. A hierarchical framework for smart grid anomaly detection using large-scale smart meter data. *IEEE Trans Smart Grid*. 2017;PP(99):1-1. https://doi.org/10.1109/TSG.2017.2697440. IEEE early access articles.
30. Wang K, Wang Y, Hu X, et al. Wireless big data computing in smart grid. *IEEE Wirel Commun*. 2017;24(2):58-64.
31. Raspberry Org. Raspberry pi model b. http://www.raspberrypi.org/products/model-b/; 2016.
32. Araújo PRC, Holanda R, Rodrigues AW, Arajo AC, Aguiar J, Oliveira JP. A middleware for the integration of smart grid elements with WSN based solutions.*International Journal of Distributed Sensor Networks (IJDSN)*. 2017;2014:15. Article ID: 506203. https://doi.org/10.1155/2014/506203.
33. Wagenknecht MA, Braun T. Snomc: an overlay multicast protocol for wireless sensor networks. In: Paper presented at: 9th Annual Conference on Wireless Ondemand Network Systems and Services (WONS'12). Courmayeur, Italy; 2012.
34. Modbus Organization. Modbus specifications and implementation guides. http://www.modbus.org/specs.php; 2017.
35. Distributed Network Protocol. A DNP3 protocol primer. https://www.dnp.org/AboutUs/DNP3%20Primer%20Rev%20A.pdf; 2005.
36. Triangle MicroWorks. Distributed test manager (dtm). http://www.trianglemicroworks.com/products/testing-and-configurationtools/dtm-pages; 2017.

How to cite this article: Araújo PRC, Filho RH, Rodrigues JJPC, Oliveira JPCM, Braga SA. Middleware for integration of legacy electrical equipment into smart grid infrastructure using wireless sensor networks. *Int J Commun Syst*. 2017;e3380. https://doi.org/10.1002/dac.3380