Educational platform for communications using the MQTT protocol

D. Matabuena, F.J. Bellido-Outeiriño, A. Moreno-Muñoz, A. Gil-de-Castro, J.M. Flores-Arias
Departamento de Ingeniería Electrónica y de Computadores,
Escuela Politécnica Superior de Córdoba, Universidad de Córdoba,
Campus de Rabanales, Edificio Leonardo Da Vinci, E-14071 Córdoba, España.
Email: {p22magod, fjbellido, amoreno, agil, jmflores}@uco.es

***Abstract*— With the emergence of Internet of Things (IoT), appears the necessity of a protocol to handle the data flow of all the smart devices. HTTP (Hypertext Transfer Protocol) is the most used protocol. But, in recent years, new protocols are emerging. Nowadays, the Message Queue Telemetry Transport (MQTT) is one of the most used protocol within IoT, due to its low power and bandwidth consumption. This work presents the implementation of an MQTT protocol into LabVIEW platform providing a powerful tool to assess students training into novel protocols and applications under the IoT paradigm.**

Keywords— Educational platform; Internet of Things, MQTT;.

I. Introduction

The term Internet of Things (IoT) is increasingly being used in many applications, aiming to connect everyday objects to Internet. This collection of information generates a massive amount of data that should be managed in an efficient way to develop an optimal system [1]. Internet of Energy (IoE) has arisen from the combination of Smart Grids (SGs) and IoT systems. This concept offers many advantages, such as the interaction between different devices. Communication of different machines (M2M) supplies the term of decentralized systems and its introduction into the Wireless Sensor Network (WSN) philosophy, which consists of distributed nodes realizing measurement and control tasks and sending this information between them or with a central node [2], [3]. In [4], an example of the integration of IoT in Demand Response (DR) systems is shown. By using wireless technology, a platform that allows home load control is achieved.

The arrival of the Industry 4.0 creates the necessity to introduce students to IoT systems. However, teaching instrumentation and IoT concept simultaneously is difficult due to the high number of students, the large amount of content of the discipline and the short time available. Thus, educators need to redesign the subject and are obligated to find new ways of integrating students into IoT systems. Therefore, this paper aims to develop a practical lesson that have been used with students enrolled in the Master´s Degree in Distributed Renewable Energies from the University of Cordoba (https://www.uco.es/estudios/idep/masteres/energias-renovables-distribuidas). More specifically in the subject Instrumentation and Metrology, where the students learn the fundaments of measurement techniques and error correction in instrumentation.

This research is supported by the Spanish Ministry of Economy and Competitiveness under Project TEC2016 77632-C3-2-R

The master's degree is focused in Distributed Energy Resources. DER not only refer to small generation nodes located close to consumption areas but includes other resources as energy storage system or even load management resources. Consequently, the distribution network is turning into a complex model that is hard to be controlled [5]. The Smart Grid's (SG) paradigm has emerged as the only viable solution for the seamless integration of these DERs in today power grids. This concept provides a better position for the knowledge of the different issues that presently challenge the electrical network [6]. There are in fact many studies focused on improving the grid characteristics [7], [8].

In [9], the author elaborates an educational platform which reproduces the behavior of a power generation system associated with active loads. This system integrates measurement, protection, stability, analysis and Power Quality (PQ) control features of the distributed system. Furthermore, the ability to create events makes it easier to test the power generation system itself. In this paper, the extension of the Smart Grid Test Bench (SGTB) is proposed. The main objective of this paper is to integrate this system into the IoT philosophy. It will be possible due to the addition of an MQTT protocol library developed in LabVIEW Software (Laboratory Visual Instrument Engineering Workbench). This library has been used to realize a practical lesson with master´s students.

The rest of the paper is as follows. Section 2 describes the laboratory structure and configuration for IoT communication. Subsequently, Section 3 describes the methodology implemented to achieve the integration of the SGTB into an IoT platform, making an extensive description of the communication protocol used. In addition, the created Graphic User Interface is shown. Section 4 presents different techniques that students use in the practical lesson. Finally, Section 5 contains the conclusions.

II. Laboratory background

As explained in the introduction, this article aims to expand previous research [6] and introduce the SGTB into an IoT environment. The SGTB consists of three main parts: hardware system, communication (MQTT Protocol) and monitoring system and software. Every part is described below.

A. Hardware System

The selected hardware platform to implement the broker has been an NI CompactRIO (cRIO) [10] which is at the top level of the developed system. Its architecture is a rugged control embedded system, including a real-time controller, a FPGA module and IO modules. In addition, its Ethernet connection makes it adequated for high efficiency wireless industrial projects. cRIO equipment is treated as Smart Grid's nodes. This equipment allows the addition of input/output modules, which makes it a good option for high-speed control and measurement tasks. Several models have been used as a control system. On the one hand, the cRIOs 907x models use VxWorks operating systems. On the other hand, cRIOs 9030 and 9063 contain a Linux RTOS operating system.

B. Message Queue Telemetry Transport Protocol (MQTT)

A wide variety of communication protocols can be applied, such as AMQP, STOMP or MQTT. The main advantage of Advanced Message Queuing Protocol (AMQP) is its reliability and functionality, currently used by Google and NASA. Nevertheless, in comparison to MQTT protocol, it has the disadvantage of high data overload. The Streaming Text Oriented Messaging Protocol (STOMP) is lightweight and text-based [11]. However, it does not offer a reliable exchange. Because of the above reasons, MQTT protocol has been selected to communicate all the nodes of the network. MQTT is a client-server publish/subscribe protocol based on TCP/IP communication. Nowadays, it is used in communication within the Internet of Things (IoT), due to its low power and bandwidth consumption. In MQTT, a client sends a topic subscription request or a publish message to a broker (server). This broker manages the data flow and replies the requests according to the protocol rules. The client's actions are defined below.

On the one hand, the client submits a membership request to a topic with a *subscribe message*. If the broker accepts this petition, the client will receive the publications sent by other MQTT devices, for that topic. On the other hand, the customer can send a *publish message* to a topic, with a defined level of Quality of Service (QoS), that are techniques used to ensure that a receiver captures the message. When the broker receives this communication, redistributes it to every subscribed to the above-mentioned. The user-configured QoS level gives the system a great flexibility, and it is composed of three levels. At level 0, called *At most once delivery*, the sender delivers the publish message just once. As can be seen in Fig 1, it is not possible to know if the subscriber has ever received the message. Greater number of message exchange exists in level 1 (*At least once delivery)* because a recognition response is needed. If the receiver does not reply with an acknowledge message, the transmitter duplicates and resends the delivered message (Fig 2). At *Exactly once delivery* level (QoS 2), the receiver acquires the published message just once. As shown in Fig 3, the sender has an acknowledge message when the broker delivers the message to the subscribers, thus, the publisher assures that listener receives the message. To sum up, the Quality of Service inclusion makes the protocol to be a flexible tool. This allows different level of reliable package reception. However, QoS levels increase the number of exchanged packages. Moreover, a higher QoS level involves an increase of the communication delay. As a consequence, systems with a significant number of nodes connected imply a low quality of service level.

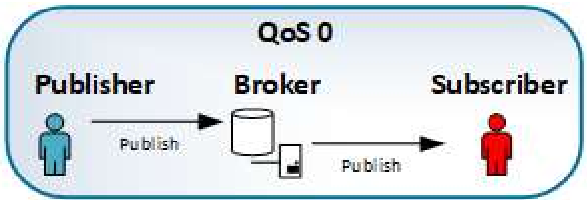


Fig. 1. Message exchange diagram of QoS 0.

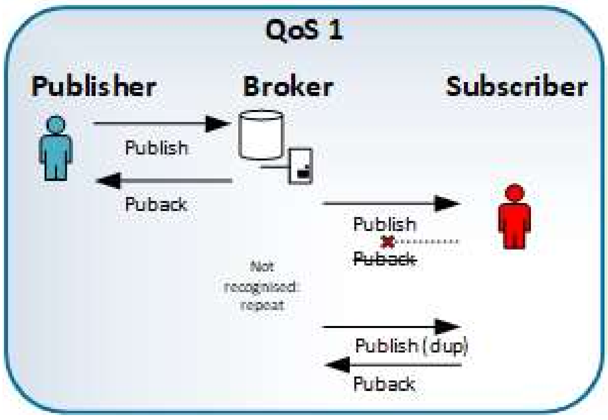


Fig. 2. Message exchange diagram of QoS 1.

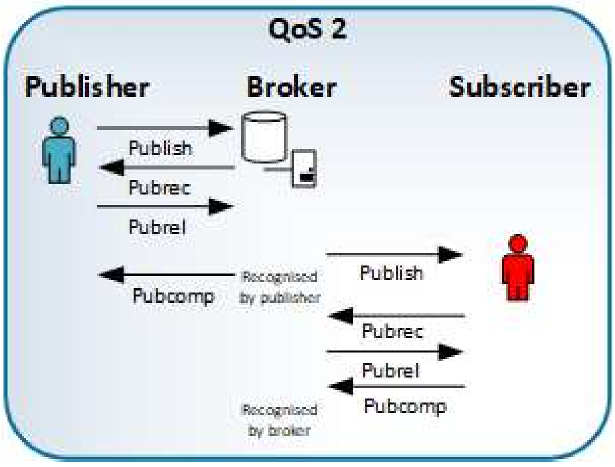


Fig. 3. Message exchange diagram of QoS 2.

C. Monitoring system and Software

Finally, there is a monitoring system, with lab-distributed computers. A graphic user interface (GUI) is available (described in section III). This fact enables the control of the data acquisition systems. In addition, LabVIEW has been the programming tool. This software allows testing and debugging the correct functionality of the developed library. For its development, it is essential to know that LabVIEW follows a data flow model to run Virtual Instruments (VIs). A block diagram node is executed when it receives all the required inputs. As a disadvantage, this software is not open source and a license is required. Despite the above inconveniences, LabVIEW has been used to teach students how to work with technology widely utilized in industrial environments.

III. Final Application

The developed system is included in the laboratory framework. For this reason, cRIO microcontroller linked the reliability assessment platform of power quality and the MQTT platform. The advantage here is that in the meantime, FPGA and I/O modules are able to realize other tasks (i.e. data measurement tasks as explained in Section 2). This software has been used for creating an MQTT protocol-based library. The Application can be summarized in three control loops: Receiving and data processing Control, Transmission Control, and Time Control.

First, the *Receiving and data processing* control loop manages several broker's functions. For this reason, it has been handled with a states machine technique, in which the next state depends on the current state as well as the value of the inputs, as shown in Fig. 4. The possible states are defined in this section. In the Initialize status, the server creates a TCP listener at port 83, specified by MQTT protocol, which is waiting for new connections in the second state called Status Check. While there is no customer's connection, the server is waiting in this subroutine. After detecting a new connection, the control loop advances to Process Data status, which is the most important part of the library. It is composed of several subroutines and functions. To get started, the server expects to receive a new message. After that, it decodes the information encrypted in

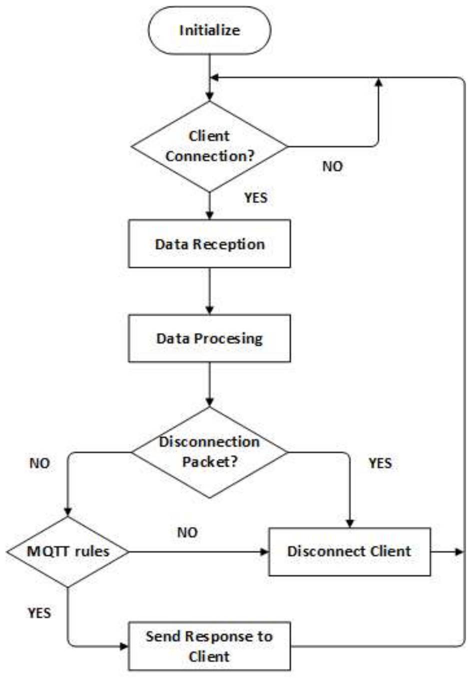


Fig. 4. MQTT communication flow chart

UTF-8 (8-bit Unicode Transformation Format), that is a long variable ISO 10646 standard [12]. The information is saved in LabVIEW class data types (LV Object), allowing its access at every moment.

In addition, diversified types of messages are distinguished according to the literature [13]. Each of them is associated with a subroutine with properties, different from each other. Nevertheless, the SubVIs agree on several aspects. On the one hand, every subroutine must verify that the received packet agrees with the protocol specifications. If not, the client communication is automatically closed. On the other hand, the control loop must insert information into a buffer, called *Enqueue Keep-Alive*, which communicates the *Process Data* control loop with the *Time Control* loop as explained later. Subsequently, if a response to client is needed, the broker assembles a response, in UTF-8, and send it through the transmission loop. The building of a publish packet message is composed of four parts. First, the Fixed Header contains the message type and the QoS level, among other data. Second, Variable Header is composed of the Topic name and the message identification (ID). Payload, which contains the rest of the message, is shown in part C. Afterward, every package is concatenated and sent to the Transmission Control Loop, in part D.

Second, the *Transmission Control* loop is in charge of sending data packages to client(s). It receives codified information from the process control loop by a queue's technique. Later on, in the Transmission subVI, the packet is sent to client, using the TCP-IP port that has previously been opened. Furthermore, to display the number of sent messages, the number of times that this subVI is executed is count. It belongs to a different loop because it was necessary to achieve an independence between reception and transmission. It allows using different sampling frequency in reception and delivering.

Finally, *Time Control* loop accomplishes an exhaustive time verification, necessary in communication. If there is not information from a client for a time called Keep-alive, the broker must close the connection. In the developed library, this concept is managed as a back account. When the value of a time counter reaches the value of a customer's keep alive time, the client is automatically disconnected. The control loop is executed every 10 ms, even if no customer's message is received. The information is received from the process control loop in a buffer.

Whenever the server receives a message, this Time Control Loop performs. Then, data flow reaches Keep Alive SubVI, which is made up of four main parts. First, the status of the time counter is calculated with the difference between the current time and the time of the last execution of this SubVI. Part 2, is a function in charge of updating the information of the keep-alive time of customers sending a message. The execution of this part of the code depends on the reception of a message. The third part looks for customers whose time count has been finished and return as output the client connection reference. In that case the customer reference is inserted in a disconnect queue. This queue is linked to the process loop, where the connection will be closed. Part 4 execution depends on the existence of the above conditions by a customer.

A. *Graphic user interface and network manager*

The educational system developed in LabVIEW is constituted as an interface to be provided to students. The GUI is helpful to perform the practical training, as a consequence of the created library, which can act as a server too. From this interface, the student can observe the broker redistributing all the received messages and also how it controls the connections of all the customers. The GUI is subdivided into different groups, as shown in Fig 5.

In the part 'A' of Fig. 5, the received and transmitted packets can be observed. As packets are codified in UTF-8, each box corresponds to a byte of information. Part 'B' in Fig. 5 corresponds to general network information. The number of connections and a timer are located. In the middle of the region B, client identifications and *keep alive* information are shown. *Keep alive* is the time the client can spend without sending information to the broker. After this time, the broker must close the connection with this client. *Retain message* information is kept and shown in region B.3. When a client sends a publish package with the retain flag enabled, the broker must save this message-topic combination, and send it when a new subscriber connects to this topic. The part 'C' in Fig. 5 shows important information from each client. TCP/IP connection reference, the client identification (ID) and its *keep alive* time is located in region C.1. Kept data about the client's subscriptions is shown in region C.2, which is composed of the client's topic(s) and their respective Quality of Service. A single client can be subscribed to several topics with various QoS levels. The region C.3 contains information about the 'will message'. This message is optional and is automatically sent when the client turns off the connection with no reason and without receiving a disconnect package. On the contrary, 'will message' is not sent if the broker has decided to disconnect the client because do not comply with the protocol rules. The last region (C.4) kept QoS messages (topic and message), which can´t be deleted before the message acknowledgement, as explained in Section II. When the broker recognizes the message packages, it deletes the QoS package.

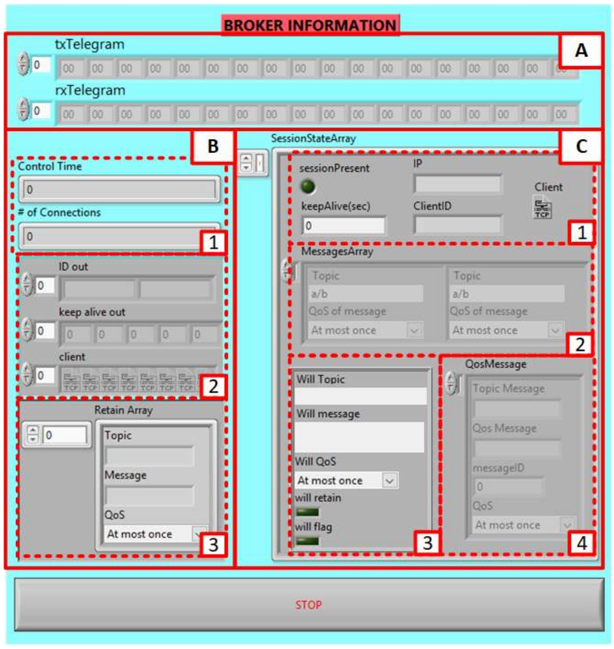


Fig. 5. LabVIEW GUI Broker Information interface

IV. FINAL TESTS

Clients and servers (brokers) are needed to stablish a communication in the MQTT protocol. For this reason, the paper presents a practical lesson in which students can test different platforms for both parts. For the practical session, each student is assigned to a work position. In Fig. 6 is presented the material that will be used. At first, an Arduino Uno WiFi board is connected to a Light Depending Resistor (LDR) and a Negative Temperature Depending (NTC). Furthermore, students are all provided of a Raspberry Pi Board and a Windows Computer, and there is a cRIO system to all of them. The functionality and connection of each of the devices is explained below.

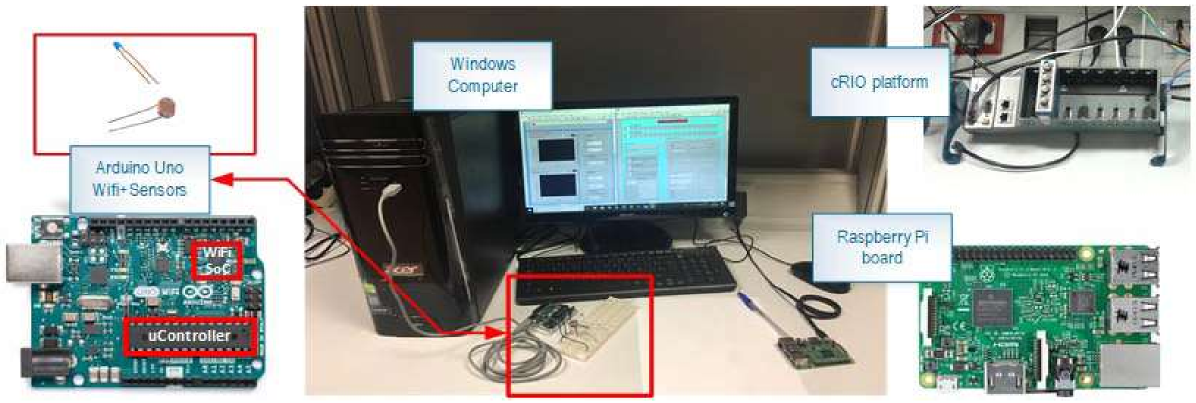


Fig. 6. Overview of the work position.

A. Clients

To be in accordance with a WSN philosophy, each client will act as a network node. For each of these clients, the student will send messages to a server whether it is a text message only or data from a sensor-based measurement. The proposed nodes are defined as below.

1) MQTT.fx

Mqtt.fx is a MQTT client written in Java. This program is supported by MacOS, Windows, and Linux. The greatest benefit of using this software is providing a graphic user interface, which allows a simple and intuitive way of introducing the IoT philosophy to students [13]. To set up the application, students just have to define the server's IP address and specify the topic they want to publish data on. Other protocol configurations can also be modified, such as 'keep alive' time, 'clean session' flag and the MQTT version. The objective of this practical work's section is character strings messages with Mqtt.fx program and observe them with the IoT platform shown in Fig 7. Messages published to the topic *IoTplatform/mqttfx* are displayed at the top of Fig. 8.

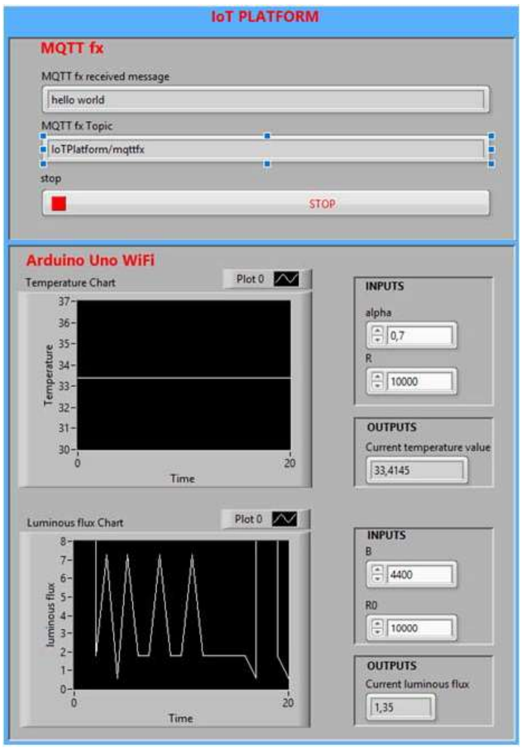
2) Mosquitto App

Mosquitto is a lightweight open-code implementation of the broker features. In addition, its installation on the PC provides a library for deploying MQTT brokers and clients [14], [15]. In the practical lesson, the students have a guide explaining how to access the program through the Windows command prompt and explaining the controls to use for sending messages between clients. A publisher posts information on two topics called *IoTplatform/test* and *IoTplatform/test2*. Another client subscribed to the above-mentioned topics, displays the published package when it is received.

3) Arduino Uno WiFi

Arduino Uno WiFi [16] board is based on the ATmega328 microcontroller and integrates WiFi co-processor module ESP8266. As a free hardware platform, Arduino offers multiple advantages. For instance, a free software development environment in C language that allows easy programming. Therefore, Arduino is broadly extended in student day life. In addition, there is literature and forums to help the student learning about this platform. These features provide an easy starting environment for IoT applications by means of the specific purpose library Ciao, which could be effortlessly configured and connected to the broker through MQTT protocol.

In the practical lesson, the students are provided with a LDR and NTC. They are connected to Arduino Uno WiFi board via I2C protocol and they are periodically measuring the illuminance and temperature in the room. Furthermore, data is sent to an IoT server as explained in subsection B. Sensors are connected to a 10 kΩ resistance and powered with 5 V provided by the Arduino board, as shown in Fig 8. In addition, a user interface to display the results is shown in Fig 7. The interface allows to modify the parameters required to convert voltage to the value unit of the measurement (ºC or Lux). It also shows a graph of the values collected in the last 20 seconds.


IoT PLATFORM

MQTT fx

MQTT fx received message: hello world

MQTT fx Topic: IoTPlatform/mqttfx

stop: STOP

Arduino Uno WiFi

INPUTS
alpha: 0,7
R: 10000

OUTPUTS
Current temperature value: 33,4145

INPUTS
B: 4400
R0: 10000

OUTPUTS
Current luminous flux: 1,35

Fig. 7. IoT Educational Platform

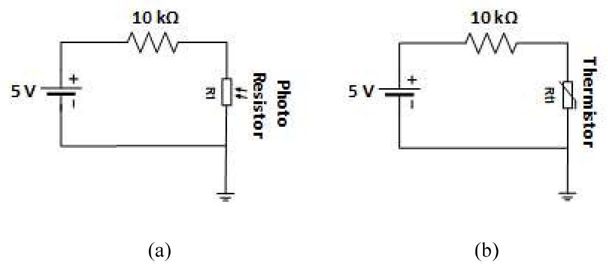


Fig. 8. Sensor connection circuit: (a) LDR sensor; (b) NTC sensor.

B. Broker

To show students the inclusion of the IoT in different environments such as domestic area or an industrial one, in this part of the practical lesson, the students test the developed schedule in Section III, in several devices. They have been classified according to increasing level of industrialization. To check the correct performance of the implemented library in the different proposed devices, students are allowed to use either devices explained above as network clients.

1) Windows computer testing

The basic programming of the library has been done on a Windows computer, attempting to isolate the program from the computer hardware. Only the use of port TCP/IP 83 was necessary. The student in the practical training has a computer, therefore, the aim of the lesson is to test the computer as a server/broker of MQTT. This is accomplished with the LabVIEW libraries provided to the trainee. To check that it works properly, any of the clients described above can be used.

2) *Raspberry Pi board*

Raspberry Pi is a small, lightweight and high-performance ARM based computer. It is composed of a development environment around the microcontroller [17]. The aim of executing the developed library in LabVIEW into this device is to create a portable, rugged, and a low-cost system. To achieve that, it is needed to install LabVIEW 2014 version. Afterward, students can check the correct operation of the Raspberry Pi acting like a server.

3) *CRIO testing*

To provide an industrial point of view to the practical work, several cRIO control embedded systems have been used, with different Operating systems, as described in Section II. Introducing Compact-RIO system into the IoT environment offers the advantage of being equipped with a microcontroller, a FPGA and I/O modules. Consequently, in combination with being the network core, cRIO can perform other tasks in parallels, such as data measurement or control system. During the development of the practical lesson, students test the cRIO working as a server.

V. Conclusion

The paper presents an implemented educational platform for MQTT Protocol. As can been explained during the paper, after the creation of the platform different tests were carried out. The program was run with different LabVIEW compatible devices, such as a Windows computer, Raspberry board or high performance embedded control systems such as Compact-RIO. It was successfully achieved, so it can be said that the program has been isolated from the hardware of the computer where the program works.

As a final conclusion, the practical work provides the student with the ability to create a WSN network in a residential and industrial environment. In future works, a studio and integration into the practical work of an IoT cloud, can be performed. In addition, the developed GUI teaches the trendy forms of communication. Students can evaluate each package information and check the different parts of the communication protocol, from the physical layer, through TCP/IP protocol, to the control framework (MQTT). At the end of the subject, master's students complete a satisfaction questionnaire. The IoT platform was evaluated in a positive way by the students.

References

[1] M. Jaradat, M. Jarrah, A. Bousselham, Y. Jararweh, and M. Al-Ayyoub, "The Internet of Energy: Smart Sensor Networks and Big Data Management for Smart Grid," *Procedia Comput. Sci.*, vol. 56, pp. 592–597, Jan. 2015.

[2] N. Bui, A. P. Castellani, P. Casari, and M. Zorzi, "The internet of energy: A web-enabled smart grid system," *IEEE Netw.*, vol. 26, no. 4, pp. 39–45, 2012.

[3] F. J. Bellido-outeiriño, J. M. Flores-arias, M. Liñan-reyes, E. J. Palacios-garcía, and J. J. Luna-rodríguez, "Wireless Sensor Network and Stochastic Models for Household Power Management," in *IEEE Consumer Electronics Society*, 2013, no. Iec 62386, pp. 483–491.

[4] C. Mahapatra, A. Moharana, and V. Leung, "Energy Management in Smart Cities Based on Internet of Things: Peak Demand Reduction and Energy Savings," *Sensors*, vol. 17, no. 12, p. 2812, 2017.

[5] A. Moreno-Muñoz, J. J. G. De La Rosa, R. J. Real-Calvo, and V. Pallarés, "Embedding measurement in Distribution Automation Systems," *IEEE Int. Symp. Ind. Electron.*, pp. 3722–3727, 2010.

[6] P. Siano, "Demand response and smart grids—A survey," *Renew. Sustain. Energy Rev.*, vol. 30, pp. 461–478, Feb. 2014.

[7] V. Salehi, A. Mohamed, A. Mazloomzadeh, and O. A. Mohammed, "Laboratory-based smart power system, part I: Design and system development," *IEEE Trans. Smart Grid*, vol. 3, no. 3, pp. 1394–1404, 2012.

[8] D. Celeita, M. Hernandez, G. Ramos, N. Penafiel, M. Rangel, and J. D. Bernal, "Implementation of an educational real-time platform for relaying automation on smart grids," *Electr. Power Syst. Res.*, vol. 130, pp. 156–166, 2016.

[9] I. M. Moreno-garcia, A. Gil-de-castro, A. Moreno-munoz, V. Pallares-lopez, R. Medina-gracia, and D. Matabuena, "Educational Platform for Reliability Assessment of Power Quality," in *IEEE Global Engineering Education Conference*, 2018, pp. 1618–1624.

[10] National Instruments, "What is CompactRIO? - National Instruments." [Online]. Available: http://www.ni.com/compactrio/whatis/. [Accessed: 29-Jan-2018].

[11] V. Gazis *et al.*, "A survey of technologies for the internet of things," in *IWCMC 2015 - 11th International Wireless Communications and Mobile Computing Conference*, 2015, pp. 1090–1095.

[12] International Organization for Standardization, "ISO/IEC 10646:2012 - Information technology -- Universal Coded Character Set (UCS)." [Online]. Available: https://www.iso.org/standard/56921.html. [Accessed: 27-Feb-2018].

[13] Jens Deters, "Welcome to the home of MQTT.fx." [Online]. Available: http://mqttfx.jensd.de/. [Accessed: 18-Feb-2018].

[14] F. J. Bellido-Outeiriño, J. M. Flores-Arias, E. J. Palacios-Garcia, V. Pallares-Lopez, and D. Matabuena-Gomez-Limon, "M2M Home Data Interoperable Management System Based on MQTT," *IEEE International Conference on Consumers Electronics (ICCE)*, Berlin, pp. 2–4, 2017.

[15] Eclipse, "Eclipse Mosquitto." [Online]. Available: https://mosquitto.org/. [Accessed: 18-Feb-2018].

[16] Arduino.cc, "Arduino Uno WiFi." [Online]. Available: https://store.arduino.cc/usa/arduino-uno-wifi. [Accessed: 29-Jan-2018].

[17] Raspberry Pi Foundation, "Raspberry Pi - Teach, Learn, and Make with Raspberry Pi." [Online]. Available: https://www.raspberrypi.org/. [Accessed: 18-Feb-2018].