An IEEE 1451 Standard-based Plug-and-Play Architecture to Empower the Internet of Things on Smart Environments

Edelberto Franco Silva[1], Bruno J. Dembogurski[2], Francisco Henrique[1], Eduardo Pagani[1], Gustavo S. Semaan[3], Joao Costa[1], Igor Junqueira[1], Melissa Santos Aguiar[1], Christian Scoralich[1], Luiz Filipe P. M. Araujo[1]

[1]Universidade Federal de Juiz de Fora - UFJF
[2]Universidade Federal Rural do Rio de Janeiro - UFRRJ
[3]Universidade Federal Fluminense - INFES e EEIMVR/UFF

***Abstract*—In this paper, we introduce a new solution for the Internet of Things (IoT) paradigm based on Plug-and-Play (PnP) architecture, aiming to stimulate new paths set by the concept of the Smart Environment. In this sense, the IoT ecosystem is considered and the layers are presented since the most high level of software to the lowest level of sensors and actuators. Unlike the state-of-the-art solutions, our proposal features are transparent in both hardware and software domains, supporting the identification of sensors through an easy and intuitive tool responsible to create and manage all resources and logic programming in an IoT's environment. To show the main contributions of this research will be introduced an architecture, respecting the IEEE 1451 standard and based on IoT protocols, as well-defined message protocols. It is interest to highlight the contribution of this work on all PnP levels in IoT architecture, since the platform to user interface, different of all another proposals. Moreover, a report of the first results about the hardware and software implementations will be showed through a use case in a real testbed, proving PnP facilities.**

***Index Terms*—IoT, Plug-and-Play, IEEE 1451, MQTT, IEEE 802.15.4**

I. Introduction

Nowadays, Internet of Things (IoT) are becoming more and more present in daily life through applications such as smart building, health care, automotive, manufacturing, industrial automation, green vehicle, etc [1]. The basic concepts which we have in this area are about the transducer, sensor, and actuator. The transducer is a device that converts one form of energy into another. The sensor is a device that converts a physical parameter to an electrical output and the actuator is a device that converts an electrical signal to a physical output. Thus, the IoT is regarded as a technology and economic wave in the global information industry after the Internet. The IoT is an intelligent network which connects all "things" to the Internet for the purpose of exchanging information and communicating through the information sensing devices in accordance with agreed protocols. It achieves the goal of intelligent identifying, locating, tracking, monitoring, and managing things [2] [3] [4].

Contact: Edelberto Franco Silva - edelberto@ice.ufjf.br.

Industrial areas and facilities are benefiting from this trend, known as Industrial Internet or Industry 4.0. The Industrial Internet of Things (IIoT) uses smart sensors and actuators to enhance manufacturing and industrial processes. This leverages the power of smart machines and real-time analytics to take advantage of the data produced by many machines used in different processes. Considering that smart machines are better than humans to capture, analyze and communicate important data is the moving force behind IIoT. Thus, this stream of data can be used to direct business decisions more accurately, reducing risks and increasing speed.

In the IoT paradigm, many objects surrounding us will be connected into networks in one form or another. RF identification (RFID), sensor technology, and other smart technologies will be embedded in a variety of applications. For the standardization of this environment and its development, the IEEE 1451 standard must be taken into account.

The IEEE 1451 has been around for almost 20 years and in that time it has seen many changes in the world of smart sensors intelligent systems have many advantages such as real-time monitoring controlling, plug and play facility, easy interface to server based application, etc. The National Institute of Standards and Technology (NIST) has promoted the development of the IEEE 1451.X standard series for smart sensors.

These standards establish a foundation for development of "intelligent" networked sensors, and under the IEEE 1451.X are different subgroups [1], [5].

The IEEE 1451.0 defines the functions that are performed by a smart transducer interface module (STIM) and the common characteristics for all devices that implement the STIM. IEEE 1451.1 defines a common object model and programming paradigm for smart transducer systems. IEEE 1451.2 characterizes a transducer to network capable application processor (NCAP) interface and TEDS for a point-to-point configuration [1]. IEEE 1451.3 defines a transducer-to-NCAP interface and TEDS for multi-drop transducers using the HPNA (Home Phoneline Networking Alliance) communication protocol [6]. IEEE 1451.4 defines a mixed-mode interface for analog transducers with analog and digital operating modes. IEEE 1451.5 defines a transducer-to-NCAP interface and TEDS for wireless transducers, which can be understood by standards such as 802.11 (WiFi), 802.15.1 (Bluetooth), 802.15.4 (LR-WPAN *a.k.a* "ZigBee"), and 6LoWPAN. The IEEE P1451.6 defines a transducer-to-NCAP interface and TEDS using the high-speed CAN open network interface. The IEEE P1451.7 defines an interface and communication protocol between transducers and RFID systems.

The use of sensors, actuators, and other devices in the IoT has substantially increased in the last few years. Every day your components become smarter and the communication becomes informative. While the IoT is still seeking its own shape, its effects have already established solutions for the connected scenario. Aiming stimulates new paths set by the concept of the Smart Environment, this work introduces a new solution based on Plug-and-Play (PnP) architecture, which the IoT ecosystem is considered. This way, we believe that the PnP paradigm brings simplicity in the use and integration of new devices in the IoT network. Thus, directly or indirectly, this architecture can be used to solve several real-life problems [3] [4] [7].

From the literature survey, it is observed that the existing systems suffer three lacks, corresponding to the graphical user interface (GUI) for easy operations and configurations of its IoT environment, the necessity to have a transparent PnP for IoT platforms respecting the IEEE 1451 standard, and the requirement to decrease the processing and resource consumption in a IoT node. Minding in these gaps, the main contribution of this paper is to introduce an easy, quick and intuitive architecture where the user can configure as PnP its IoT's sensors and actuators that can be applied to any kind of smart environments, mainly in industry. Thus, this article presents a proposal to fill the boxes of IEEE 1451 standard with real IoT applications and protocols. Complete architecture is introduced and a proof-of-concept implemented in a real-world testbed. In addition, the proposal stimulate new ways to solve several specific real-life problems, enjoying and improving the power of IoT.

The rest of the article is organized by: Section II introduces the architecture proposal; Section III evaluate and validate the solution; Section IV introduces the state-of-the-art of proposals and solutions related to PnP; Section V concludes the paper.

II. Architecture Proposal

The proposed architecture can recognize automatically more than 100 sensors as plug and play devices, allowing our solution to be applied on diverse IoT and IIoT's fields. The communication among the microcontroller is made through the I^2C (Inter-Integrated Circuit)[1], allowing to standardize the addresses and decrease the use of cables. Our microcontroller is totally generic, its function is to identify the sensors and send its data to our server through an asynchronous message exchange protocol. The NodeRED [8] is our server, in it we have all the data identification process and, if necessary, it is possible to control the actuators. The NodeRED is generic for all existing microcontrollers, allowing the control of several boxes simultaneously.

The architecture is presented in Figure 1. It shows all layers and entities involved in the IEEE 1451 standard family *i.e.* STIM, TEDS, and NCAP. The Section I have introduced all subgroups of IEEE 1451 and its possible to highlight all 3 (three) layers to which this proposal fits plus the application layer and its GUI (NodeRED).

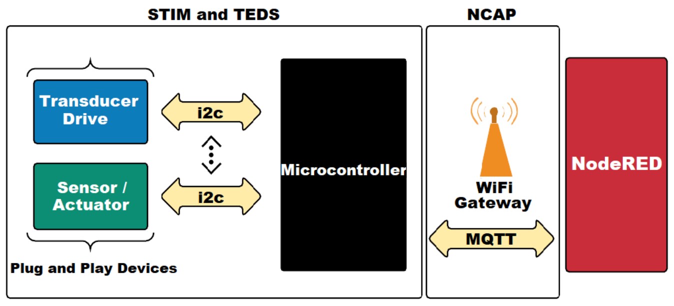

Fig. 1: Our architecture proposal in high-level.

The STIM and TEDS are responsible to export to the user a high-level interface to interact with sensors and actuators. In this layer, occurs the data acquisition

[1] I^2C is a serial protocol for two-wire interface to connect low-speed devices like microcontrollers, EEPROMs, A/D and D/A converters, I/O interfaces and other similar peripherals in embedded systems, and it is used by almost all major integrated circuit manufacturers.

and the PnP functionality. From the transducer drive is possible to interact with the sensors/actuators using an I^2C bus from the microcontroller (the main platform board). The NCAP is represented by a WiFi gateway responsible to collect and send data from STIM and TEDS to the user interface through a message queue protocol (*i.e.* a broker) supported by IEEE 1451. On the right side of the figure is possible to see the NodeRED application, a user interface responsible to allow the user to interact with sensors/actuators as PnP, create its programming logic, get the telemetry, etc, keeping all hard processing on the application level.

A. GUI and Flow-based programming tool

Many common IoT scenarios in areas such as home and industrial automation require integration with online services and real-time sensing and actuation. However, several web-based platforms have emerged to ease the development of interactive or near real-time IoT applications by providing a way to connect things and services together and process the data using a data flow paradigm. As they begin to address interactive IoT scenarios like these we have the NodeRED [8].

NodeRED is an open source browser-based development tool for wiring IoT components, originally developed by IBM's Emerging Technology Services team and now a part of the JS Foundation [8]. It provides flow-based [9] visual programming, which describes an application's behavior as a network of nodes. This encapsulates part of the complexity of creating such complex systems. Each node is basically a self-contained JavaScript function with a well-defined purpose, given an input data it will process it and send it forward through the network flow.

Mainly, the NodeRED operates, as seen in Figure 2, with three different types of nodes: input, process, and output. Input nodes have a grey box on the right side, indicating that it will only send information. Processing nodes have two grey boxes, one for input and a second one for outputting the result of its process. Finally, Output nodes have one grey box on the left side to receive information and output it in the correct format.

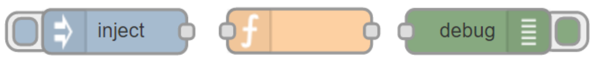

Fig. 2: Three main types of nodes in NodeRED. From left to right: input, process and output.

Also, NodeRED has a variety of predefined nodes which are ready to use. It encourages users to extend the palette of nodes creating new ones, new flows and sharing those with the community as JSON files.

B. Communication

The message exchange between STIM and user's applications are only possible because of communication protocols. In this section are introduced two kinds of communication protocols: the first one is responsible to physically connect (*e.g.* wired or wireless) the objects, and then another one which is responsible to sensing and acting on the transducers from user's applications in a standard fashion.

1) Application Level:

The MQTT[2] (*Message Queue Telemetry Transport*) protocol is a lightweight messaging protocol, that employs the *publish/subscribe* paradigm, specifically designed for machine-to-machine (M2M) and Internet of Things [10]. The idea of this protocol is that devices can publish somewhere, so that other devices can obtain this published information. In [11] it is introduced how MQTT can be incorporated as part of IEEE 1451 and in [12] its modification to support constrained nodes.

The protocol entered into a standardization process by OASIS (*Organization for the Advancement of Structured Information Standards*) [13], where it had the specification openly published with a license without *royalties* for many years, and companies such as Eurotech have implemented this protocol in their products. The use of this protocol considers three basic communication elements: the publisher, the registered user, and the *broker*. The publisher is a device responsible for collecting information and send it to *broker*. The *broker* receives the message and send it to all registered users so they can access the information. Alternatively, there are clients who can be registered in one or more topics, according to their needs and availability. The users receive information from the *broker* and use according to their own wish. The figure 3 shows the operation of the protocol.

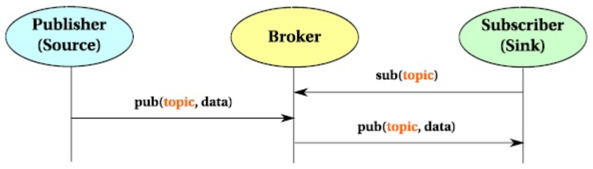

Fig. 3: Topic-based Pub/Sub Communication Model using the MQTT protocol [14].

Embodying all aspects of the *publish/subscribe* paradigm, MQTT decouples these parts spatially, so to publish or receive messages, publishers and subscribers only need to know the hostname/IP and port of the broker. Also, it decouples time, allowing the

[2]http://mqtt.org/

broker to store messages for clients that are not online [15]. Lastly, MQTT works asynchronously, even though in some cases synchronization is possible and needed. Nevertheless, this is due to the fact that most client libraries are based on callbacks and work asynchronously, thus tasks are not blocked while waiting for a message.

2) Link Layer: The link layer is responsible to make possible the network connection and communication between STIM and NCAP.

First of all, it is necessary to remember the benefits of IPv6 in IoT environments and the requirements to use it on constrained nodes. As one may know, the IP protocol was considered heavy for use in LR-WPAN – Low-Rate Wireless Personal Area Networks –, but IPv6 could be adapted to be possible the communication between the Internet and sensors network. The efforts of the 6LoWPAN [16] standard have focused mainly on the compression and fragmentation techniques, being possible the exchange of IPv6 packets from the Internet to the low-power networks without damaging the size available for payload. The 6LoWPAN adaptation layer was created respecting the maximum frame size of LR-WPAN/IEEE 802.15.4 standard (*i.e.* 127 bytes).

Trying to be a comprehensive standard, IEEE 802.15.4[3] concentrate its effort to define the working of the link and the physical layer, delegating the operation of the higher levels to the different kinds of applications that the standard will be used.

In scenarios where nodes have more resources, it is possible to get the benefits of adopting IEEE 802.11 WiFi standard-based solutions. For example, in the case of an IIoT scenario where the nodes are in the same hangar where a WiFi infrastructure pre-exists, it is natural to think of ways to leverage this infrastructure to create an intelligent environment.

C. Component Details

As the lowest level, we present our prototype board-/platform. In order to allow a simple connection with any sensor or actuator type, it was developed a pair of boards with a common connection interface. These common interfaces can be any kind of serial plug, being only necessary to respect the order of connection wires. As mentioned, the Transducer Drive is responsible to provide unique identification for each sensor/actuator, allowing the data flow between the Microcontroller and NodeRED. The format of the data is a JSON[4] representing the node and its sensor/actuator.

To illustrate our Microntroller (main board), introduced by Figure 1, two boards are presented (front and back view) in Figure 4, both designed with the concept of modularity in mind. The main board is able to accept up to eight devices through a simple cable interface. It uses an integrated NodeMCU[5], so each device connection is mapped to the microcontroller pins. Since the NodeMCU has only one ADC pin, only one device with an analog input is supported, at the same time all connections, including the analog, supports digital input and output devices.

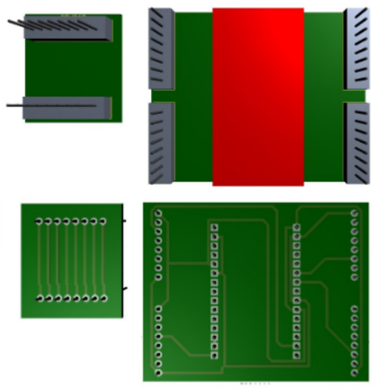

Fig. 4: Model of the platform boards.

Along with the main board, it was developed a sensor board. It is a simple board which maps any device pins to the common connection interface used in the main board. These abstract the device connection, who has the responsibility to treat the collected data and create all the logic functions is the upper layer, *i.e.* GUI.

With the serial cable adopted by our proposal, sensors and actuators can be used from up the 2 meters from the main board, allowing the development of multi-device solutions which can cover a whole indoor space like a room or even a small outdoor space like a light post.

III. Validation

In the context of IoT, a product with a wide range of applications is paramount. Thus, the main goal of the architecture developed in this work was to enable the user to apply it in many different scenarios *e.g.* Smart Homes, Industry etc. Also, it is noteworthy that any sensor or actuator can be used, defining, on the application-level the plug-and-play's logic, improving or changing the solution.

The purpose of this section is to demonstrate the interaction among all components of our architecture proposal. In Figure 5, steps 1 through 4 are only data acquisition and data processing steps, the user only

[3]https://standards.ieee.org/standard/802_15_4-2015.html
[4]https://www.json.org/
[5]https://www.nodemcu.com/

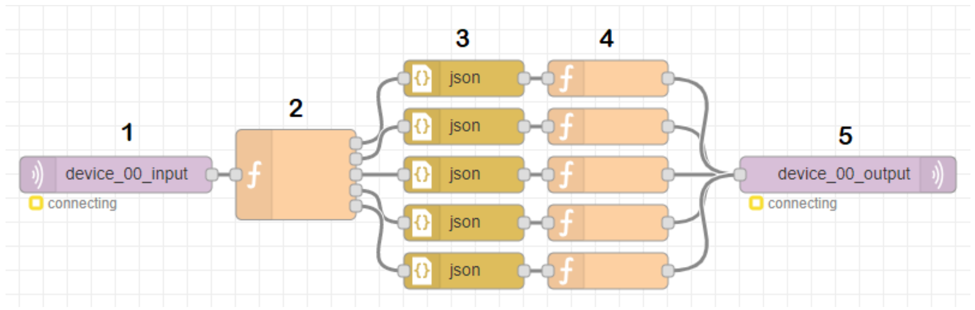

Fig. 5: Flow-based programming for the use case.

connect the nodes and create the logic functions to treat them.

As a bottom-up presentation, a digital sensor sends raw data and its unique identification to the application level through the middle layers, i.e. network, NCAP and application (NodeRED).

The main reason for using NodeRED is the possibility of transferring libraries to the internet server atmosphere, transforming a high-level NodeMCU processing to lower processing.

Following the steps highlighted in Figure 5 is possible to see a block called "device" indicated by 1. This block is responsible to receive the raw data in JSON format from MQTT broker. All platforms with the same kind/model of sensors/actuators that have been identified publish in the same topic and throw its unique ID[6]. The step 2 (block called "f") represent the functions responsible to identify the type of element – as sensor or actuator – and redirect to the correct flow on the next step. The step 3 (block called "json") is responsible to convert the raw data in objects accessible to the logic functions. Thus, step 4 – called "f" – is a function, responsible to get the values and make decisions as send alerts, store data in databases, reply MQTT messages to actuators, etc. In the last step, number 5 – called "device_*" – is possible to see all topics associated to any sensor/actuator. This step is only started if step 3 want to communicate with sensors/actuators.

Therefore, the NodeMCU is responsible only to send data (unique identification and data collected) from one specific port to a specific topic. All logic, identification and how this data need to be treated, is handled on an application-level – at NodeRED. In this way, sensors can be plugged in and unplugged from the NodeMCU platform transparently from the architecture, and the user is only responsible to create the function in flow logic-programming, treating the data get by sensors.

In this example, the main logic is to get the data from the flame sensor ($flame$ variable) and check if it is true, and if it is, an alert is sent to the user. Another step of this process is the verification of the increasing possibility of a fire. This is done verifying the $temperature$ and $humidity$ by the DHT11' sensor. The algorithm shows only a simple experiment, but a relevant one, showing how users can create programming functions and embed it on block "f" - step 4 Figure 5.

In our validation, one platform was connected to an MQTT broker via WiFi, and one sensor was plugged and unplugged to validate the PnP functionality. The user interface was capable of collecting the data and process it, resulting in a validation success. Another validation was conduct with more sensors and actuators – 3 sensors and 1 actuator – plugging and unplugging them, validating again our expectations of PnP features and application flow.

IV. Related Work

In this section will be present the main works and solutions related to our field of research, from PnP interfaces to market (and black box) solutions.

The first one is [17], when was propose a set of actions, as well as a set of state variables called Universal Plug and Play Smart Transducer - uPnP. One of these variables provides a list of available channels and another the list of reachable transducers. The uPnP protocol aims at exchanging services between devices attached to a TCP/IP network. The PnP on uPnP is based on the Control Point and tuples of channel responsible to implement the transducer and to identify the sensors connected to the control network. This solution follows the IEEE 1451 architecture recommendation, having the NCAP (Network Capable Application Processor) as a gateway to allow the connection of a SM (Sensor Manager) - i.e. TIM (Transducer Interface Module) element at IEEE 1451 - through a WiFi network. The main benefits of uPnP is to allow the addressing, discovery, description, control, and eventing for sensors attached to the network.

[6]An ID number is a unique sequence number.

The MicroPnP platform [18] is a commercial solution developed by VersaSense. Its PnP is composed by three-simultaneous sensors/actuators plugged to the board, and these sensors/actuators need to be part of a supported elements list. The details of MicroPnP and its PnP are not showed, but it is possible to see the limitation of only plug three sensors/actuators at the same time. Although the platform currently offers more than twenty five peripherals that can be associated to the device via PnP and fit into a variety of everyday situations, the central device is still limited to only serving three peripherals, greatly limiting its applications.

Another initiative is from Microsoft, called Iot Plug and Play from Azure[7]. It proposes to turn easy the registration of an IoT device on Azure platform, creating the possibilities of a developer to build its IoT solution. But, it is necessary to know the use of PnP concept is more thinked on application level, did not any kind of automatic recognition of an IoT device in any level.

In contrast to all related work, our proposal is a complete architecture for PnP respecting the IEEE 1451 standards. Creating a transparent integration and identification of a sensor/actuator to the user interface, allowing him/her to create its own application to monitor and actuate on the IoT environment. Another differential of other proposals and ours is the scalability. Each platform based on our solution can support approximately 127 sensors/actuators – limitation imposed by I^2C bus – and the number of platforms running simultaneously is greater than one hundred.

V. Conclusions

In this work, a new solution for IoT ecosystem based on Plug-and-Play architecture was presented. Its main contribution is the facility to communicate and configure a sensor or actuator. Thus, an architecture based on IEEE 1451 standard family was introduced filling the boxes of that standard with real protocols and standards adopted from the IoT ecosystem. In order to demonstrate such advantages, a use case for a specific scenario has been shown. However, as the work makes clear, it is simple to change any sensor or actuator to create new scenarios and applications.

We believe that the PnP paradigm brings simplicity in the use and integration of new devices in the IoT network. This way, the solution stimulates new ways to solve several specific real-life problems. In future works, the scalability of our proposal and user experience will be measured. Another investigation will be the evaluation of the necessity to use a WiFi's zeroconf.

[7]https://azure.microsoft.com/en-us/updates/iot-plug-and-play/

References

[1] A. Kumar, V. Srivastava, M. K. Singh, and G. P. Hancke, "Current status of the ieee 1451 standard-based sensor applications," *IEEE Sensors Journal*, vol. 15, no. 5, pp. 2505–2513, 2015.

[2] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[3] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the internet of things: A case study," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1275–1284, April 2018.

[4] P. Ray, "A survey on internet of things architectures," *Journal of King Saud University - Computer and Information Sciences*, vol. 30, no. 3, pp. 291 – 319, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1319157816300799

[5] K. B. Lee, "Ieee 1451: A standard in support of smart transducer networking," in *IEEE INSTRUMENTATION AND MEASUREMENT TECHNOLOGY CONFERENCE PROCEEDINGS*, vol. 2. IEEE; 1999, 2000, pp. 525–528.

[6] A. Dutta-Roy, "Networks for homes," *IEEE spectrum*, vol. 36, no. 12, pp. 26–33, 1999.

[7] B. Kang, D. Kim, and H. Choo, "Internet of everything: A large-scale autonomic iot gateway," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 3, no. 3, pp. 206–214, July 2017.

[8] N. O'Leary and D. Conway-Jones, "Node-red," 2013. [Online]. Available: https://nodered.org

[9] J. P. Morrison, *Flow-Based Programming, 2Nd Edition: A New Approach to Application Development*. Paramount, CA: CreateSpace, 2010.

[10] A. S. Clark, "Message queue telemetry transport (MQTT)," 1999. [Online]. Available: http://mqtt.org

[11] J. Velez, R. Trafford, M. Pierce, B. Thomson, E. Jastrzebski, and B. Lau, "Ieee 1451-1-6: Providing common network services over mqtt," in *2018 IEEE Sensors Applications Symposium (SAS)*, March 2018, pp. 1–6.

[12] E. F. Silva, B. J. Dembogurski, A. B. Vieira, and F. H. C. Ferreira, "IEEE P21451-1-7: Providing more efficient network services over mqtt-sn," in *2019 IEEE Sensors Applications Symposium (SAS)*. IEEE, 2019, pp. 1–5.

[13] A. Banks and R. Gupta, "MQTT version 3.1.1. OASIS standard," 2014. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[14] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "Mqtt-s âĂŤ a publish/subscribe protocol for wireless sensor networks," in *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, Jan 2008, pp. 791–798.

[15] J. Dizdarevic, F. Carpio, A. Jukan, and X. Masip-Bruin, "Survey of communication protocols for internet-of-things and related challenges of fog and cloud computing integration," *CoRR*, vol. abs/1804.01747, 2018. [Online]. Available: http://arxiv.org/abs/1804.01747

[16] V. P. Nikshepa and U. K. K. Shenoy, "6lowpanâĂŤperformance analysis on low power networks," in *International Conference on Computer Networks and Communication Technologies: ICCNCT 2018*, vol. 15. Springer, 2018, p. 145.

[17] A. R. Nieves, N. M. Madrid, R. Seepold, J. M. Larrauri, and B. A. Larrinaga, "A upnp service to control and manage ieee 1451 transducers in control networks," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 3, pp. 791–800, 2011.

[18] N. Matthys, F. Yang, W. Daniels, W. Joosen, and D. Hughes, "Demonstration of micropnp: The zero-configuration wireless sensing and actuation platform," in *2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, June 2016, pp. 1–2.