A Performance Evaluation of Raspberry Pi Zero W Based Gateway Running MQTT Broker for IoT

Diana Bezerra Correia Lima
Department of Electrical Engineering
Federal University of Paraíba
João Pessoa, Brazil
diana.lima@cear.ufpb.br

Rubens Matheus Brasil da Silva Lima
Department of Electrical Engineering
Federal University of Paraíba
João Pessoa, Brazil
rubens.lima@cear.ufpb.br

Douglas de Farias Medeiros
Department of Electrical Engineering
Federal University of Paraíba
João Pessoa, Brazil
douglas.medeiros@cear.ufpb.br

Renata Imaculada Soares Pereira
PNPD-PPG-Electrical Engineering
Federal University of Paraíba
João Pessoa, Brazil
renata@dee.ufc.br

Cleonilson Protasio de Souza
Department of Electrical Engineering
Federal University of Paraíba
João Pessoa, Brazil
protasio@cear.ufpb.br

Orlando Baiocchi
School of Engineering and Technology
University of Washington Tacoma
Tacoma, WA, USA
baiocchi@uw.edu

***Abstract*—The Internet of Things (IoT) has become widely used in recent years in a wide range of applications, such as, weather condition monitoring, transportation, smart homes, smart cities, smart farm, etc. The ecosystem of the IoT is also vast, including from sensor and hardware devices up to cloud-computing. An approach that is getting more and more attention in the IoT ecosystem is the edge-computing and one of its fundamental pieces of equipment is the edge-computing gateway (GTW), which can working as a data-processing device nearer to the things and as a bridge to the Internet, as well. The most important features for these GTWs must be robustness and efficiency and a very popular solution is to use low-cost Raspberry Pi card-size computers. Considering protocol solution, Message Queue Telemetry Transport (MQTT) communication protocol has been considered one of the most applicable to IoT because of its low-power capability. In this context, this paper describes a study about the performance evaluation of a low-power member of the Raspberry Pi family, the Raspberry Pi Zero W, working as an IoT gateway and running MQTT. The experimental results show its performance using as metrics: the processor temperature, the CPU usage level, and rate of MQTT received messages under different Quality of Services (QoS).**

***Index Terms*—Internet of Things, Protocol, Machine-to-machine communications, Performance evaluation, Raspberry Pi, MQTT.**

I. INTRODUCTION

The Internet of Things (IoT) is a technological revolution that can represent the future of computing and communications, and its development depends on dynamic technical innovation in a number of important fields, from wireless sensors to nanotechnology [1]. The IoT revolution has as its premise that any object or thing (like a light lamp, a door, a refrigerator, a garment, etc.) can directly originate and send data to the Internet without any human interaction [2].

The IoT aims to connect everyday objects (things) to large databases through the Internet embedding smart device in the things themselves that, according to the embedded information processing capabilities, can further enhance the power of the IoT network to its edges [1]. Currently, IoT technological advances are providing an increasingly connected society generating data for problem solving and about 500 billion devices are expected to be internet-connected by 2030 [3].

According to [4], an ecosystem of the IoT is composed of:

- the connected devices and gateways, including both hardware platforms,
- the connectivity between devices and the Internet,
- the application and supporting services.

As it can be observed, an IoT research goes through several domains include the wireless sensor/hardware development, communication, and cloud-computing. Furthermore, IoT-related technologies have been applied in many application domains, varying from weather condition monitoring and transportation to smart homes, smart cities, smart farm, and consumer electronics. As a consequence, the IoT represents a convergence of multiple domains, and can be seen as an umbrella term uniting these underlying technologies [4].

As just described, the IoT ecosystem is very vast coming from hardware platforms up to cloud-computing domain. An approach that is getting more and more attention in the IoT ecosystem is the edge-computing and one of its fundamental pieces of equipment is the edge-computing gateway (GTW), which can working as a data-processing device nearer to the things and as a bridge to the Internet, as well.

From the IoT ecosystem [4], taking in consideration just the "connected devices and gateways" and "the connectivity

This work was partially funded by the National Institute of Science and Technology of Micro and Nanoelectronics Systems (Proc. 573738/2008-4 INCT NAMITEC), by Maranhão Research Foundation (FAPEMA, Grant: 01550/16) and the PNPD CAPES-Brazil (Grant Procs. PNPD 2878/2011). C.P. Souza would like to acknowledge the Research Productivity Scholarship – CNPq (Brazilian National Council for Scientific and Technological Development, Proc. 311794/20165 DT). In addition, the authors also acknowledge the CNPq and the Fundação de Apoio à Pesquisa do Estado da Paraíba - Brazil (FAPESQ) for financial support (PRONEX).

978-1-7281-2530-5/19/$31.00 ©2019 IEEE

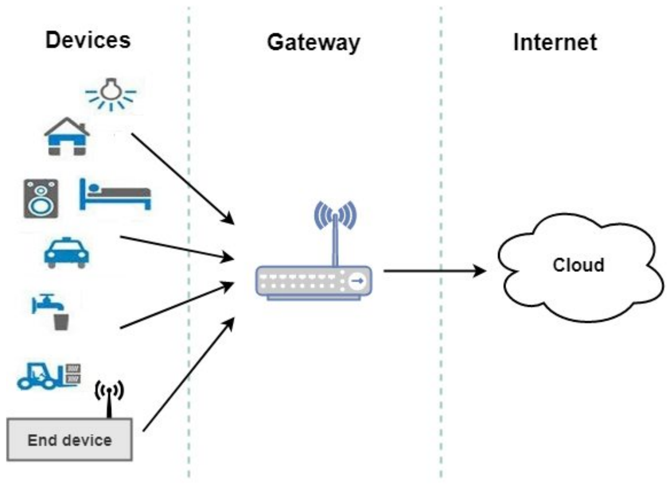

Figure 1. Network architecture using gateway.

between devices and the Internet", an IoT application can have a lot of built-in sensors working in a spacial and temporal distributed way sharing a large amount of measured data into a cloud solution. In some scenarios, a particular IoT enabling-network has to deal with some constraints such as power consumption, low cost, low latency, network congestion and low flexibility. In such scenarios, gateway (GTW) for edge-computing can be used as a local data concentration device that receives data from the sensors and transfers them to a desired destination. In general, sensor data are packed into messages that travel from the local network until reach a gateway which routes them to the Internet [5]. This gateway-based architecture is shown in Fig. 1.

The most important features of edge-computing GTWs must be robustness and efficiency and, when low cost and low power are take in account, a very popular solution is to implement them using some member of the Raspberry Pi (RPi) family. The RPi family is a set of low-cost credit-card-sized computers that contain processor, memory, and various peripheral connectors [6] and, when it is connected to a computer screen, a keyboard, and a mouse, runs rather similar to a personal computer (PC), but with some limitations.

A comparison of a RPi with other architectures such as MICAz, TelosB and Iris is provided in [6] where the authors evaluated parameters like size, price, processing power, memory, flexibility, communication, and operating system and concluded that RPi presented many advantages over the others, such as processing power and operating system. It can be concluded that a RPi is a low-cost computer suitable for using in an IoT solution.

For example, an IoT application for water level monitoring using LoRa radio wireless communication technology to send sensor data to a gateway was developed and described in [7]. The RPi was chosen to be used as a gateway, performing the interface between the sensors and the server and, according to the authors, it was a complete embedded system solution featuring also an SPI serial port communicating to a radio module based on a modulation technique, called LoRA.

Figure 2. A Picture of a Raspberry Pi Zero W.

Other application to monitor variables such as body temperature and heart rate in patients was proposed using also a RPi as a gateway [8]. The obtained data are sent from the sensors to the RPi that are then forward to the doctors. In the work proposed and described in [9], the authors implemented an wireless sensor network using RPi, also running as a gateway, to measure soil parameters, such as electrical conductivity and pH.

Although the RPi is used in many applications, the potential of the hardware is not deeply analyzed, which leaves a gap for those who want to use it for more complex tasks.Thus, a goal of this work is to set up a RPi as a gateway in order to map its limits.

Currently, a very low-cost IoT hardware solution for Edge-GTW is the Raspberry Pi (RPi) Zero W (RBpi0W). The RBpi0W is shown in Fig. 2 and is composed of a single-core ARM processor.

On the context of IoT protocol, the Message Queue Telemetry Transport (MQTT) communication protocol has been considered one of the most applicable to the IoT because of its low-power feature and its possibility to share data among machines with limited memory and processing power [10].

This paper describes an experimental study about the performance evaluation of a Raspberry Pi Zero W working as an IoT gateway running the MQTT. The main objective is to find the processing limitations of the RBpi0W and evaluate its performance in terms of processor temperature, CPU use level, and maximum number of received messages considering different MQTT Quality of Services (QoS) levels by an experimental study.

II. MQTT Protocol

In IoT, a network protocol is required for devices communication in order to define the rules of the syntax, semantics and synchronization of the communication process [11]. In general, the communication protocol models can be classified as based on **request/reply** and based on **publish/subscribe**.

The **request/reply** communication model provides a synchronous communication and uses client/server architecture. It operation is based on three basic steps:

- A client requests information sending a message to a server;
- The server receives the message and processes it;
- A reply message is sent back to the client from the server.

Some protocols based on request/reply model are, for example, REST HTTP and CoAP (Constrained Application Protocol) [12].

Otherwise, the **publish/subscribe** model provides an asynchronous communication and consists in three different parts:

- publishers,
- subscribers, and
- broker (server).

In this model, the sensors, as a data generator, are **publishers** that connect to a broker using TCP and publish their data to an address created in the broker, called **topic**. Subscribers must subscribe to some topics and can receive data posted by publishers to these topics.

MQTT (Message Queuing Telemetry Transport) and AMQP (Advanced Message Queuing Protocol) are some examples of protocols based on publish/subscribe model [13].

Developed by IBM, MQTT is very popular because of it is a lightweight publish/subscribe messaging protocol and is useful for use with low power sensors [14]. An open source MQTT broker is the Eclipse Mosquitto that is also lightweight and is suitable for use on all devices from low power single board computers to full servers makes it suitable for IoT messaging [15].

A machine running MQTT/Mosquitto can act as a gateway in an IoT solution. An important point is that Mosquitto is very flexible allowing deployment in a lot of different machines, including the ones with limited memory and processing power [10]. As shown in Fig. 3, when sensors publish data (messages) to some topic, the devices that are subscribed in this topic, receive the message. In this way, there is no need to carried out a request procedure to obtain the data, as it is needed in a request/reply model.

The MQTT defines three levels of Quality of Service (QoS): QoS_0, QoS_1 and QoS_2. The QoS is the confidence level of receiving a message. According to [14], the specifications of each QoS level are:

- QoS_0: the broker or the client receives a message at most once, without any confirmation.
- QoS_1: the broker or client receives the message at least once, with confirmation.
- QoS_2: the broker or client receives the message exactly once according to a four-steps procedure.

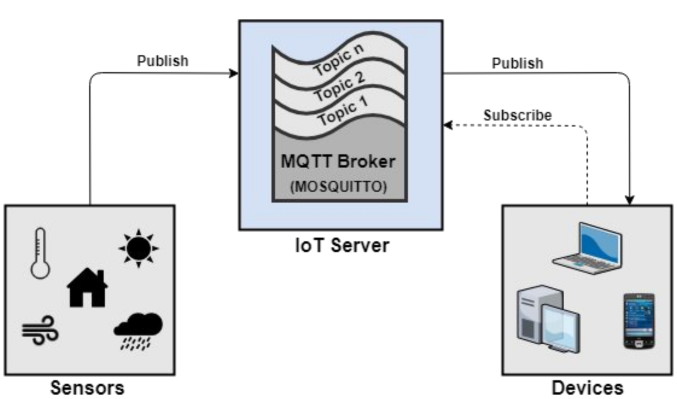

Figure 3. Basic architecture of MQTT protocol.

Figure 4. Thermal image obtaining setup using FLIR I7.

A home automation system based on MQTT using a RPi as the network gateway is described in [16] where the MQTT implementation is compared with CoAP as well as its architecture, used ports, power, and security.

III. Methodology

The operating system used on the RBpi0W-under-Test (RUT) was the Rasbian Lite, since it has no graphical interface and other memory/computation-power consumption applications, so the RUT just uses the maximum possible memory/CPU-processing capacity of the system. The RUT runs the MQTT Mosquitto broker and the remote access to the RUT is done via SSH [17].

A Task Manager for Linux [18], called HTOP, was used to obtain information about the RUT, such as memory usage, processing level of the processor, as well as the consumption of each process. The surface temperature of the RUT processor was measured by a thermal imaging camera, model FLIR I7 [19], that features a sensibility lower than 0.1^oC at 25^oC, and a range of -25^oC to 250^oC. A picture of the measurement procedure is seen in Fig. 4 and a thermal image in Fig. 5.

A. Experimental Procedures

The experimental procedures have been based on two Python scripts developed using the PAHO-MQTT library, in which enables implementations of MQTT and MQTT-SN (MQTT for Sensor Networks) messaging protocols [15].

One of the scripts was used to send messages from a PC Desktop (machine M) to the RUT ($M \longrightarrow$ RUT) via MQTT through Wi-Fi. It also receives the correspond QoS parameters of the transaction, computes the messages transfer rate (MTR), and registers the maximum MTR per second at a time instant. The other script counts the messages received of a PC Desktop M from the RUT ($M \longleftarrow$ RUT) getting data like the sent

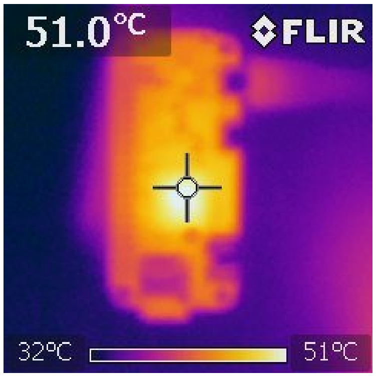

Figure 5. An obtained thermal image of the RUT.

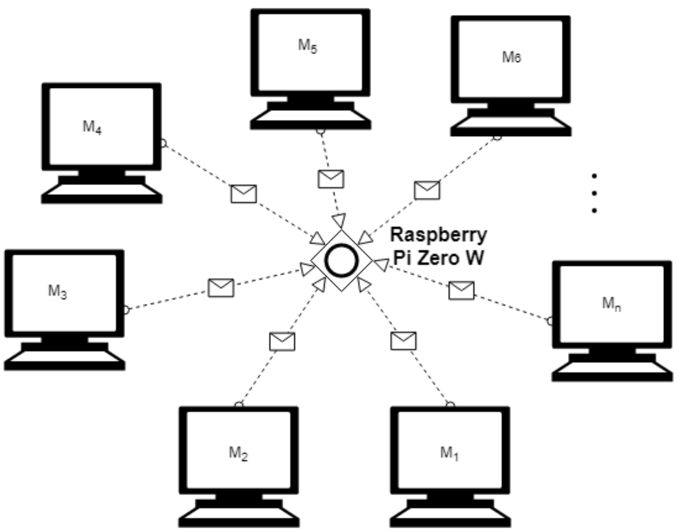

Figure 6. Experimental Setup.

messages and the MTR per second. The block diagram of the experimental setup is shown in Fig. 6.

*1) **Experimental Procedure 1**:* In the first experiment, in order to evaluate the use of different QoS, the maximum number of messages that a given machine (M_1 activated) can send to the RUT according to the chosen QoS was evaluated. After a predefined time period (10 $seconds$), a second machine (M_2 activated) begins to send messages at its maximum rate with the same QoS, and so on up to 10 machines (..., M_9, M_{10} activated). Next, the number of machines are decreased one-by-one at the same time period. In this experiment, the CPU temperature and the CPU usage level of the RUT has been also measured.

*2) **Experimental Procedure 2**:* In the second experiment in order to observe the influence of the message size (number of sent bytes), the same procedure used in Experimental Procedure 1 was repeated. However, the message payload changed from the character “1” to the char string “LABORATORIO DE MICROENGENHARIA”.

*3) **Experimental Procedure 3**:* In order to analyze the publishing capacity of the RUT, 6 machines were simultaneously used, where each of them runs 10 SHELL terminals, totaling 60 clients publishing messages to the RUT at their maximum MTR. The aim of this experiment is to obtain the rate of the number of messages per machine and the RUT processor usage level.

IV. Experimental Results

Fig. 7 shows the temperature variation of the RUT processor according to the number of active machines taking in consideration different QoS. As expected, the temperature increases when the number of machine increases. It is important to highlight that the RUT CPU temperature shown in Fig. 7 does not decreases immediately as the number of machine decreases, due to the thermal inertia, but it is possible to observe that it decrease its thermal gradient.

Fig. 8 shows the RUT processor usage according to the number of external machines. In Fig. 7 and Fig. 8, it is possible to observe that the RUT's CPU temperature and CPU usage increase when the QoS decreases, concluding that the quantity of message is more relevant than QoS. It can be observed that the lower the QoS, the greater the number of messages sent per machine, that is, its responsiveness is greater when QoS is lower.

Considering the experimental procedure 2, it was observed that the length of the message payload does not influence significantly the RUT's processor, as shown in Fig. 9. And, considering the RUT's CPU temperature, the same can be concluded, as shown in Fig. 10.

In the last experiment, when the limit of the RUT was evaluated by the use of 60 clients at the same time, it is possible to analyze how the RUT behaves with larger numbers of message/terminal reaching its limits. The number of messages shared with all 60 sending terminals can be seen in Fig. 11.

It can be observed that a maximum number of received messages was reached, even if the number of machines are very high, the number of messages, in average, stagnated about 250 messages per second as shown in Fig. 11, even having

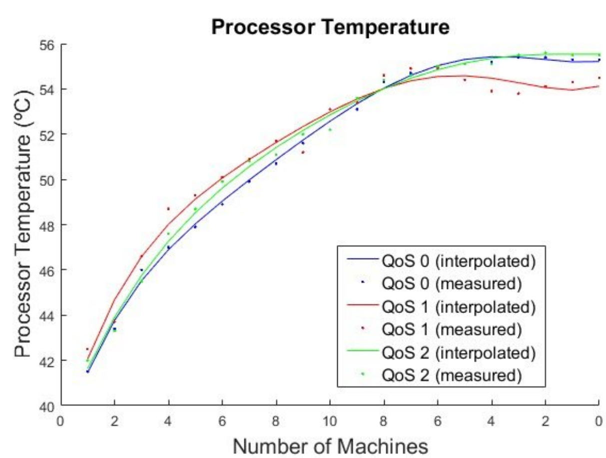

Figure 7. RUT processor temperature for the different QoS. The ”approximated” labels mean that a curve fitting was carried out.

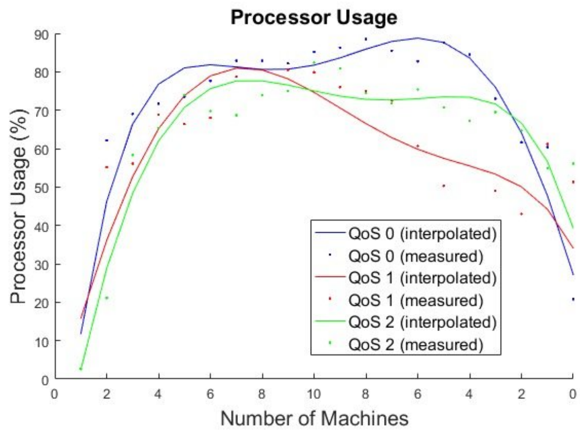

Figure 8. Processor usage for different QoS.

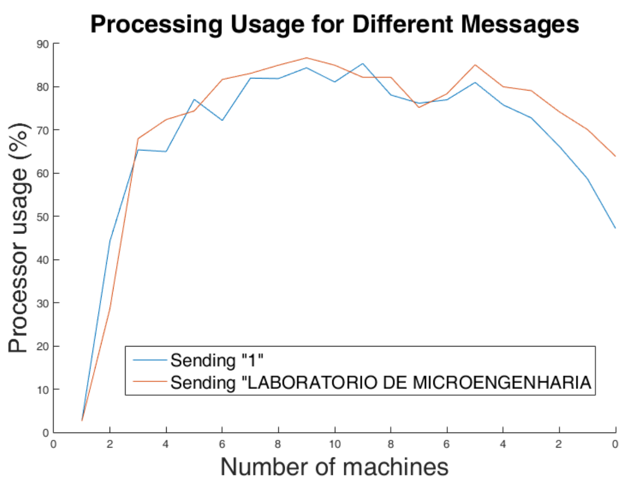

Figure 9. Processing usage for different size messages.

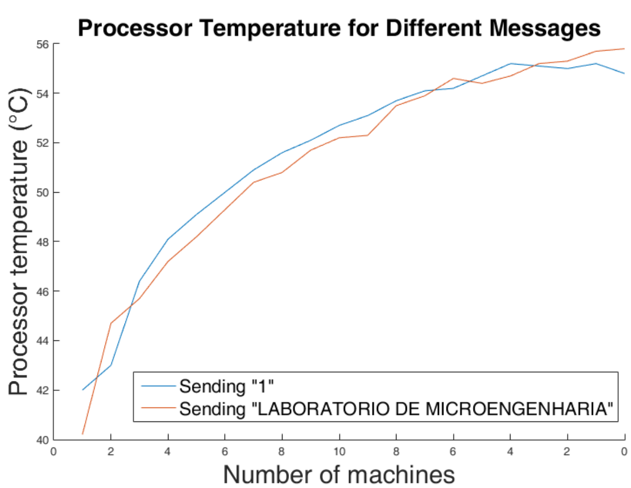

Figure 10. Processor temperature for the different size messages.

peaks above 300 or below 30 messages per second. We can also observe the temperature and percentage of RUT CPU usage in the test when the limit is reached, seen in Fig. 12.

It can be observed that RUT's CPU usage is limited to values close to 95% but never reaching 100% and the temperature follows the expected pattern, but not reaching extreme values achieving about 60^oC.

V. Conclusions

In this paper, a study about the performance evaluation of the Raspberry Pi Zero W working as an IoT gateway running MQTT protocol was described. The experimental results show that the QoS level chosen affects its performance but not in an extreme way. The same behavior pattern was observed for the CPU usage level and temperature. In all cases, the maximum reached temperature was not sufficient to damage the Raspberry-under-Test, that is, its temperature behaviour was very robust. However, it was observed a random

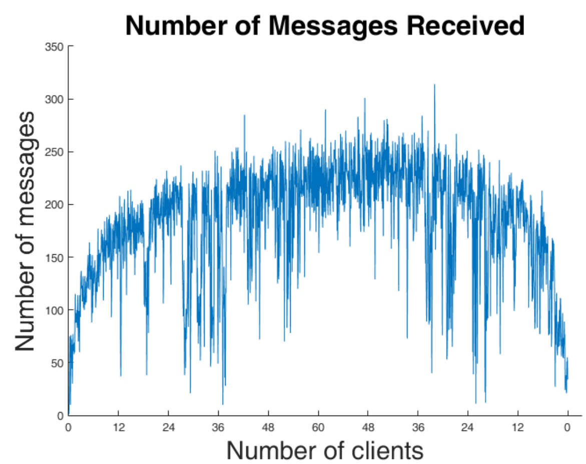

Figure 11. Received messages per client (each shell terminal in a machine).

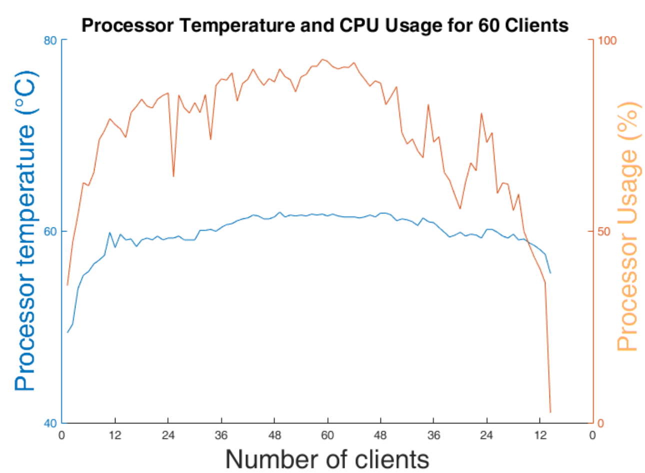

Figure 12. Temperature and CPU usage per client (each shell terminal in a machine).

variation in the number of received messages. Finally, it was observed that the Raspberry-under-Test's CPU usage did not reach the maximum even when a considerably number of machines/terminal have being sent data to it, but it may likely that the internet network used in the experiment limited the number of messages sent per machine, and not the Raspberry itself.

REFERENCES

[1] International Telecommunication Union, "ITU Internet Reports 2005: The Internet of Things Executive Summary," International Telecommunication Union (ITU), Geneva, Switzerland, Tech. Rep., 2005. [Online]. Available: https://www.itu.int/osg/spu/publications/internetofthings/InternetofThings_summary.pdf [Accessed: 2019-08-05]

[2] C. P. d. Souza and O. Baiocchi, "Energy Resources in Agriculture and Forestry: How to be Prepared for the Internet of Things (IoT) Revolution," in *Energy Systems and Environment*, U. S. o. A. Tsvetkov, Pavel (Texas A&M University, Ed. London: InTech, 9 2018, pp. 1–16.

[3] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, "A vision of IoT: Applications, challenges, and opportunities with China Perspective," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 349–359, 2014.

[4] O. Mazhelis, E. Luoma, and H. Warma, "Defining an Internet-of-Things ecosystem," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7469 LNCS, 2012, pp. 1–14. [Online]. Available: http://link.springer.com/10.1007/978-3-642-32686-8_1 [Accessed: 2019-07-14]

[5] E. Luchian, A. Taut, I. Ivanciu, G. Lazar, and V. Dobrota, "Mobile wireless sensor network gateway: A raspberry pi implementation with a vpn backend to openstack," in *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Sep. 2017, pp. 1–5.

[6] V. Vujović and M. Maksimović, "Raspberry Pi as a Wireless Sensor node: Performances and constraints," in *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2014 - Proceedings*, no. May, 2014, pp. 1013–1018.

[7] Lukas, W. A. Tanumihardja, and E. Gunawan, "On the application of IoT: Monitoring of troughs water level using WSN," *2015 IEEE Conference on Wireless Sensors, ICWiSE 2015*, pp. 58–62, 2016.

[8] S. L. Rohit and B. V. Tank, "Iot Based Health Monitoring System Using Raspberry PI - Review," *Proceedings of the International Conference on Inventive Communication and Computational Technologies, ICICCT 2018*, no. Icicct, pp. 997–1002, 2018.

[9] U. Madhura, P. Akshay, A. J. Bhattad, and G. Nagaraja, "Soil Quality Management Using Wireless Sensor Network," *2nd International Conference on Computational Systems and Information Technology for Sustainable Solutions, CSITSS 2017*, pp. 1–5, 2018.

[10] M. B. Yassein, M. Q. Shatnawi, S. Aljwarneh, and R. Al-Hatmi, "Internet of Things: Survey and open issues of MQTT protocol," in *Proceedings - 2017 International Conference on Engineering and MIS, ICEMIS 2017*, vol. 2018-Janua, 2018, pp. 1–6.

[11] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, feb 2014.

[12] J. Dizdarević, F. Carpio, A. Jukan, and X. Masip-Bruin, "A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration," *ACM Computing Surveys*, vol. 51, no. 6, pp. 1–29, 1 2019. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3303862.3292674 [Accessed: 2019-07-12]

[13] S. Sreeraj, N. S. Kumar, and G. S. Kumar, "A framework for predicting the performance of iot protocols, a use case based approach," in *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)*, Aug 2017, pp. 577–580.

[14] R. Light. MQTT man page — Eclipse Mosquitto. [Online]. Available: https://mosquitto.org/man/mqtt-7.html [Accessed: 2019-08-13]

[15] E. Foundation, "Eclipse Mosquitto." [Online]. Available: https://mosquitto.org/ [Accessed: 2019-07-03]

[16] Y. Upadhyay, A. Borole, and D. Dileepan, "MQTT based secured home automation system," in *2016 Symposium on Colossal Data Analysis and Networking, CDAN 2016*. IEEE, 2016, pp. 1–4.

[17] Secure Shell. Secure Shell Home Page. [Online]. Available: https://www.ssh.com/ssh/ [Accessed: 2019-08-15]

[18] Hisham. Htop - an interactive process viewer for Unix. [Online]. Available: http://hisham.hm/htop/ [Accessed: 2019-08-15]

[19] FLIR, "FLIR i-Series datasheet." [Online]. Available: http://www.ditek.com.mx/files/FICHA-TECNICA-FLIR-SERIE-I.pdf [Accessed: 2019-07-13]