A LoRa Enabled Building Automation Architecture Based on MQTT

Susanna Spinsante*, Gianluca Ciattaglia, Antonio Del Campo, Davide Perla,
Danny Pigini*, Giovanni Cancellieri*, Ennio Gambi
Dipartimento di Ingegneria dell'Informazione, *AEIT Sezione Adriatica
Universita' Politecnica delle Marche
Ancona, Italy 60131
Email: s.spinsante@univpm.it

***Abstract*—The current and future trend in building and home automation sees an increasing pervasiveness of Internet of Things technologies, that allow to implement networks of low power sensors and actuators, and, at the same time, to enable advanced services and applications. In this paper we propose the use of a common protocol in the IoT domain, the MQTT, over a LoRa physical communication link, to jointly exploit the lightweight pub-sub paradigm offered by MQTT and the long range, low power wireless capabilities offered by LoRa, to deliver building automation services. Experiments show that the proposed architecture ensures commands transmission and actuation in less than 200 ms.**

***Index Terms*—LoRa, building automation, sensor, MQTT.**

I. Introduction

Internet of Things (IoT) technologies, with networks of low power sensors and actuators, and advanced services and applications supported by cloud-based platforms [1], are applied in smart buildings, to dynamically adapt to changes in indoor and outdoor environment, without the need of human intervention. Typical use cases refer to energy and water metering [2], appliances monitoring and control [3], residential security monitoring (e.g. access control), automatic management of systems and devices (e.g. lighting, air conditioning or heating).

In this paper we propose the use of the Message Queue Telemetry Transfer protocol (MQTT) over a Long Range (LoRa) communication link, to jointly exploit the MQTT lightweight pub-sub paradigm, and the long range, low power wireless capabilities offered by LoRa. In [4], an MQTT broker is used to monitor and control room temperature. Wi-Fi is chosen as the communication technology; with respect to LoRa, it requires a careful network planning in huge buildings, to offer uninterrupted radio coverage in different areas, by deploying several APs. On the contrary, thanks to its robustness to deep indoor signal attenuation [5], LoRa minimizes the number of gateways needed to cover wide areas, and simplifies the system deployment. In [6], authors design a home automation system based on MQTT, but the end devices (sensors and actuators) used are equipped with a ZigBee transceiver. In our proposal, on the contrary, native LoRa enabled end nodes are used, avoiding the need for additional gateways (potential points of failures in automation solutions). About the choice of MQTT, it is able to support reliable message transmission [7], with timely alarm notification [8], so it is deemed suitable for the purpose of message exchange in building automation scenarios, with real-time constraints.

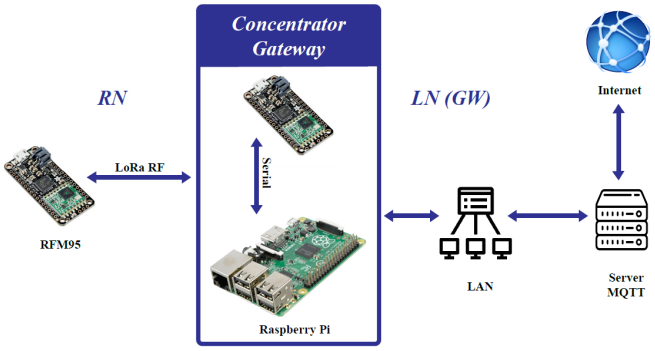

Fig. 1. Main components of the proposed system architecture.

The paper is organized as follows: Section II shortly describes the main HW and SW components of the proposed architecture; Section III presents the experimental evaluation, finally Section IV concludes the paper.

II. System Architecture and Components

Fig. 1 shows the system architecture implemented within our project. From left to right: the remote node (RN), i.e. the sensor or actuator, equipped with LoRa transceiver (a Semtech SX1276 IC [9]); the concentrator/gateway (GW), with two main elements: the LoRa radio frequency (RF) module [9] (named local node - LN), and a Raspberry Pi3 acting as the Micro Controller Unit (MCU). The concentrator connects to RN's over LoRa, and to an MQTT broker via a generic data connection (e.g. LAN-Ethernet, 4G). Finally, the backend server hosts a database and an MQTT broker. Any other MQTT client (e.g. a smartphone) may reach the broker via an internet connection. The concentrator receives frames formatted according to LoRa specifications from the RN, provides their conversion into MQTT messages, and publishes them on specific topics managed by the MQTT broker, over the available data connection.

A. The remote node

The RN is implemented on an M0 Adafruit Feather equipped with SX1276 IC [9]. The RFM95 transceiver of

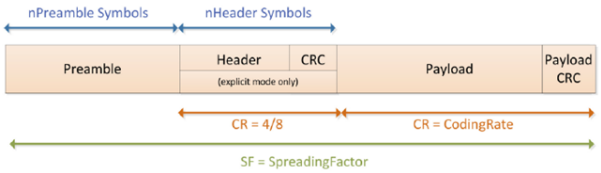

Fig. 2. A LoRa frame typical structure.

the radio interface uses a 868/915 MHz LoRa modem: the working frequency selected for use is 868 MHz. The maximum duty-cycle of the EU 868 ISM band is 1% and it results in a maximum transmission time of 36 sec/hour in each sub-band, for each end-device. Any sensor used for building automation purposes (e.g. light or temperature sensor), and connected to the RN, generates data carried into the payload of a LoRa frame, with a size limited to 255 bytes, more than enough for the purposes of our project. The general structure of a LoRa frame is shown in Fig. 2. The preamble field allows the receiver to synchronize to the incoming data. The header, in the pre-defined explicit mode, contains the packet length information (in bytes), and indicates whether a 16-bit Cyclic Redundancy Check (CRC) for the payload is present at the end of the frame. Additionally, it includes a CRC to allow the receiver to discard packets with invalid headers.

In the specific architecture developed within the project, the LoRa frame payload formatted by the nodes has this syntax: "msgID=* * * *&value=*&nodeID=*&time=* * * * * * * * * *", where each * denotes a decimal. So, four decimals are used to encode the message identifier, one for the value, one for the node (sensor or actuator) identifier, and ten decimals for the message timestamp.

B. The concentrator/gateway

As stated in the Introduction, two main components are included in the GW, i.e. the LN that provides the physical LoRa interface, and a RaspberryPi acting as the MCU. The MCU is equipped with a Linux-based Raspbian O.S.; message formatting (from LoRa on the serial port to MQTT, and viceversa) is implemented by a Python script exploiting the Paho MQTT library [10]. A background thread processes the received MQTT messages, whereas other functions in the script monitor the serial port to check for possible new messages provided by the LN. The concentrator collects LoRa frames through the LN's radio interface, and provides protocol conversion to MQTT Control Packets (CPs) [11]. The message to publish by the MQTT broker has a variable length, and it may be any type of data (binary, textual, XML or JSON). MQTT supports three levels of Quality of Service (QoS), depending on the way it manages message retransmission. In QoS level 0, MQTT operates in a "fire and forget" fashion, meaning that a best effort delivery is provided, with no guarantee about the real message reception. In QoS level 1, MQTT guarantees that a message is received at least once (but it may even be received more than once); finally, in QoS level 2 a unique reception of the message is guaranteed.

By defining the *publishers*, the *subscribers* and the *topics* managed by the MQTT broker, and setting the proper QoS level, it is possible to implement the rules governing the building automation functionalities. As an example: to switch on/off lamp A according to the light intensity value measured by sensor B, the system will request actuator A to *subscribe* to the *B light intensity value* topic, onto which sensor B is going to *publish* its data. This way, A will receive timely notification about any value transmitted by B, and will act upon accordingly. A QoS level 2 is adequate for this type of communication, to avoid unnecessary switch on/off events due to multiple receptions of the same message, or missing actuations due to lost messages.

C. The communication protocol

The data transmission protocol implemented for the project provides a LoRa-to-MQTT and MQTT-to-LoRa format conversion, operated by the RaspberryPi used as the MCU at the GW.

Assuming for the sake of simplicity that the RN acts as a sensor, the communication from it to the GW, and back, takes place according to the following steps:

1) The sensor information is encoded according to the format described in II-A The timestamp is set to a proper value, and the message is sent to the GW as a LoRa frame;
2) Once received by the LN at the GW, the whole payload of the LoRa frame is copied to the serial port connected to the MCU (the RaspberryPi);
3) The MCU copies the LoRa frame payload into an MQTT message, changes the original timestamp value to the local value computed by the MCU, and publishes the MQTT message on the predefined topic;
4) The MQTT broker delivers the message to all the nodes subscribed to that topic, among which the GW itself is listed;
5) Being a subscriber, the RaspberryPi (i.e. the GW) receives the message, changes the timestamp value to the current local one, and sends back the message to the LN on the serial connection;
6) The LN copies the MQTT message with the new timestamp into a LoRa frame payload and sends it back to the RN;
7) Finally, the RN extracts the received LoRa message ID and timestamp. They are compared to the original ones (step 1): if the message ID is the same but the timestamp values are different, this is taken as a proof that the original message generated at step 1 has been delivered by the MQTT broker.

According to the flow of messages described by the aforementioned steps, the communication protocol implemented is a bidirectional one. It enables some kind of acknowledgment about the message delivery, but it is not able to prove the success of communication. In fact, what the LoRa RN gets back from the GW is a proof that the MQTT broker has delivered its original message to the clients subscribed to

that specific topic. However, this does not ensure that the target client has received the message: it could be temporarily switched off, and no feedback about this condition is available for the RN originating the message, nor for the MQTT broker, or for the GW. It is left to upper layer-2 protocols the execution of procedures able to guarantee the message delivery to the destination node.

A similar flow of steps, as the one described above, happens when an MQTT client (like a software app running on a smartphone to let the user control the home or building automation facilities) needs to send a message to a LoRa RN, for example an actuator:

1) The MQTT client publishes a message on a specific topic;
2) The MQTT broker delivers the message to all the clients subscribed to that specific topic, among which the GW is included;
3) The GW MCU extracts the message ID and stores it locally; the message timestamp is updated to the actual value set by the GW, and the new message is copied onto the serial port connecting the MCU to the LoRa LN;
4) The LoRa LN copies the message from the serial port into a LoRa message and transmits it to the RN over a LoRa link;
5) The RN compares the node ID value of the message to its own ID. If the message is accepted, its ID value is compared to the last stored one, for freshness verification. The payload is copied to the local memory, then the message is re-transmitted, as is, back to the LoRa GW;
6) At the GW, the message goes through all the steps described previously;
7) The message, published onto a topic the GW is subscribed to, gets back to the GW. Here, after checking the message ID is the same already stored within the GW at step 3), the message is deleted to avoid further loops.

Steps 6) and 7) in the aforementioned flow ensure the possibility to check that the message originated from an MQTT client has reached the destination node (actuator) through LoRa.

When addressing the selection of a wireless communication technology for applications related to building and home automation, it is important to take into account aspects related to security, in order to avoid the risk the system gets controlled by unauthorized entities. According to the specifications, LoRa uses two AES (Advanced Encryption Standard) security keys, as shown in Fig. 3, that are unique to each LoRa device: the *NwkSKey*, and the *AppSkey*. Both keys have a length of 128 bits. The network session key (*NwkSKey*) is used for interaction between the node and the network to check the validity of messages (MIC - Message Integrity Check). The application session key (*AppSKey*) is used for encryption and decryption of the payload. There are two ways for LoRa devices to join the network: Activation by Personalization (ABP) and Over-the-Air Activation (OTAA). In the ABP case, the session keys are stored in the LoRa nodes. Dynamically activated devices (OTAA) use a third 128 bit AES application key (named *AppKey*) to derive the two session keys during the activation procedure. The choice of one of these two options is tipically based on implementation constraints.

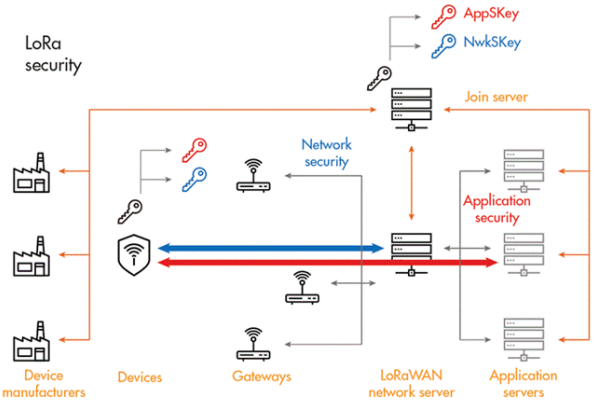

Fig. 3. Key management in LoRa [12].

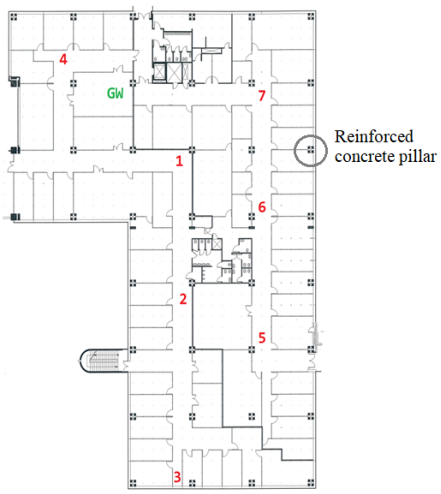

Fig. 4. PoMs and GW location for LoRa RSSI measurements on the same floor.

III. Experimental Results and Discussion

A. Radio coverage evaluation

As a first experimental evaluation, we conducted several measurement campaigns to collect data about the indoor radio coverage provided by LoRa inside a huge facility (in our University campus), to possibly replicate a realistic building automation scenario. Fig. 4 shows the location of the LoRa GW (inside the Telecommunications Lab) and the selected Points of Measurement (PoM).

Received Signal Strength Indicator (RSSI) values were collected in 7 different PoMs on the same floor, by transmitting LoRa packets of 20 bytes payload, with a Spreading Factor (SF) equal to 7. At each position, the transmitter was switched on for a total time of 30 minutes. The LoRa enabled RN was programmed to transmit around 4 packets per minute, in order to collect 120 packets per PoM. In reality, due

TABLE I
AMOUNT OF PACKETS RECEIVED AT EACH POM OVER 30 MINUTES TRANSMISSION

PoM	no. of packets received
1	109
2	111
3	111
4	110
5	112
6	101
7	110

TABLE II
AVERAGE RSSI VALUES IN INDOOR LORA PROPAGATION (SF = 7)

PoM	Avg RSSI (dB_m)	distance (m)	no. of walls
1	-61	13	3
2	-79	29	7
3	-83	53	13
4	-42	9	1
5	-95	40	11
6	-75	27	6
7	-60	22	1

to environmental conditions affecting the transmissions, the effective number of packets received at the GW from each PoM was smaller, as detailed in Table I.

Fig. 5 shows the distribution of the number of packets received for each RSSI value at each PoM, during a 30-minute-long measurement campaign. The privileged position of PoM 4 is quite evident from the graph, as it provides the packets with the highest RSSI values over all the positions tested. PoM 7 and PoM 1 are almost equivalent from the point of view of the distribution of received packets with similar RSSI values, whereas PoM 3 results more favorable than PoM 5, despite the greater distance from the GW. We can explain this behavior by looking at Fig. 4: even if PoM 3 is farther than PoM 5 from the GW, and the number of walls crossed is higher, however in order to reach PoM 5 the LoRa signal interacts with reinforced concrete pillars and gets more attenuated.

Distances between each PoM and the GW are reported in Table II, that also summarizes the average RSSI values obtained at each position, depending on the different distances and the number of walls the LoRa signal went through. It is possible to verify that the shortest distance (9 m) joint with the smallest number of walls (1) provides the highest RSSI (-42 dB_m). The variation of the minimum, maximum, and average RSSI at each PoM (according to the RSSI distributions presented in Fig. 5), with respect to the distance from the GW, is reported in Fig. 6.

B. System performance

First of all, we can state that the physical layer technology chosen for the proposed architecture, namely LoRa, is adequate to provide the necessary signal strength for the indoor enviroment considered. In fact, as shown in Fig. 6, in any case,

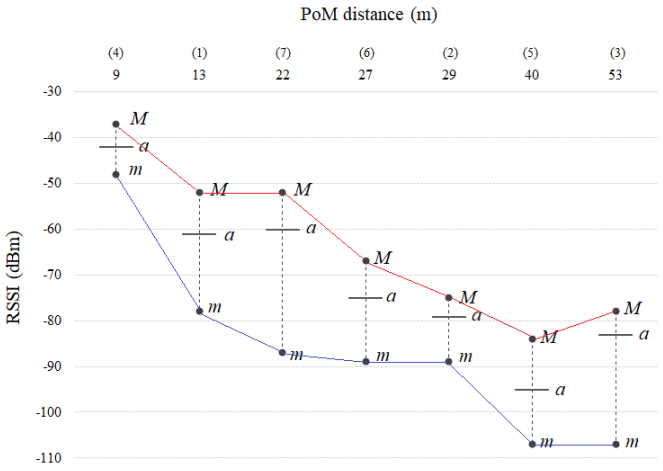

Fig. 6. Minimum (m), maximum (M), and average (a) RSSI value at each PoM, as a function of its distance from the GW. The PoM identifiers are given in brackets.

the RSSI is far above the receiver sensitivity of -142 dB_m, so the LoRa radio coverage on the floor is adequate to support the transmission of data and commands related to the building automation system.

In [13] a similar scenario is proposed with the use of Zigbee technology. Due to the short coverage range, the system architecture is more complex because in order to achieve the same coverage more nodes are needed. In fact for covering two floors, a network consisting of 16 routers is needed. By using Wi-Fi technology, as described in [14], the ESP8266 Wi-Fi module can draw spikes of 250 mA in transmission, while the RFM95 LoRa module used for this project reaches 120 mA at the maximum, during a +20 dB_m transmission. This means a shorter battery lifetime for Wi-Fi based networks, so higher maintenance costs.

We can show another important result of the proposed solution, i.e. the time requested by a message to cause a state change in the actuator node is compatible with the delay requirements posed by real-time home and building automation applications, i.e. 200 ms for delay-sensitive use cases [15]. Fig. 7 shows a screenshot of the oscilloscope displaying a delay of 174 ms between a button-pressure event and light activation. In [16] it is shown that by using the ZigBee technology, the latency for one hop is about 7.35 ms, so for the building automation scenario this leads to a total latency of 170 ms. Such a technology provides low latency but requires networks with more elements and hence higher costs and even more complex procedures to add or remove nodes to/from the network itself.

IV. CONCLUSION

This paper presented an MQTT/LoRa architecture to support real-time building automation services. Extensive experimental measurement campaigns proved LoRa is well suited to ensure adequate radio coverage in indoor scenarios, even in big and bulky buildings, resulting in a received signal strength

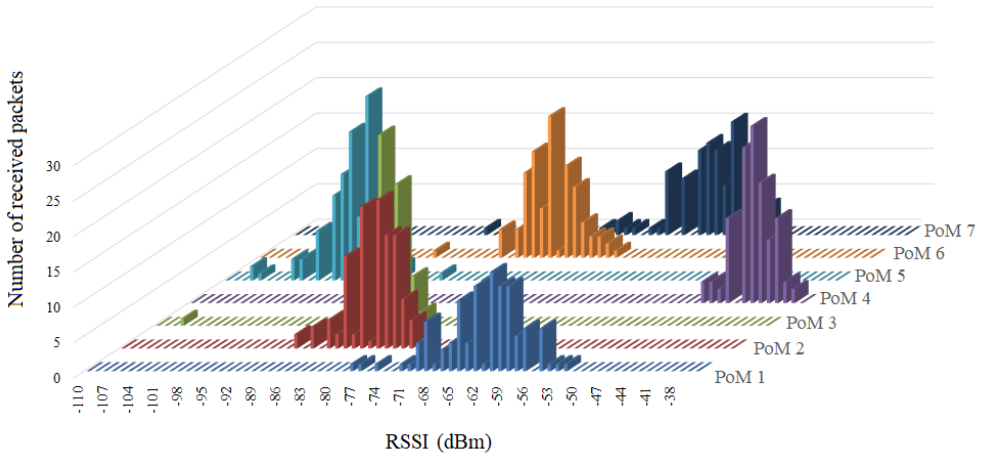

Fig. 5. Distribution of the number of packets received for each RSSI value at each PoM.

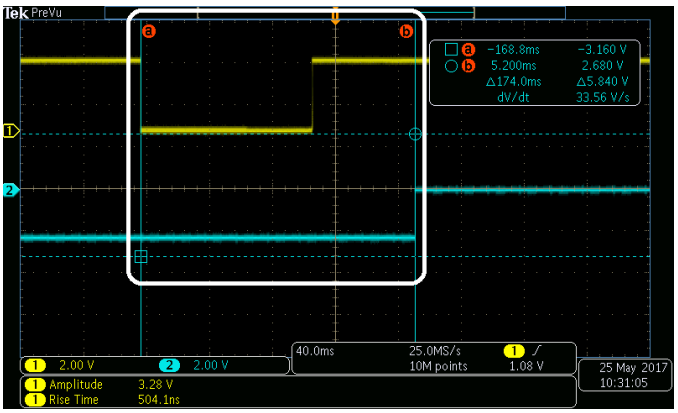

Fig. 7. Oscilloscope screenshot showing a delay of 174 ms between command transmission and execution.

well over the receiver sensitivity. An ad hoc communication protocol to ensure a basic acknowledgment mechanism to the messages exchanged among nodes of the building automation system, and between them and any third-party MQTT client, has been designed and its performance tested, thus verifying the real-time requirements of the building automation service are fulfilled. Future works will concern the extension of radio coverage characterization to outdoor areas, and tests to verify the protocol performance when the number of nodes in the automation system increases.

ACKNOWLEDGMENT

This work was developed within the framework of the Italian National Technology Cluster project SHELL (Shared interoperable Home Ecosystems for a green, comfortabLe and safe Living), project code: CTN01_00128_111357.

REFERENCES

[1] L. Hou, S. Zhao, X. Xiong, K. Zheng, P. Chatzimisios, M. S. Hossain, and W. Xiang, "Internet of things cloud: Architecture and implementation," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 32–39, December 2016.

[2] S. Spinsante, S. Squartini, L. Gabrielli, M. Pizzichini, E. Gambi, and F. Piazza, "Wireless m-bus sensor networks for smart water grids: Analysis and results," *International Journal of Distributed Sensor Networks*, vol. 10, no. 6, p. 579271, 2014.

[3] J. Joshi, V. Rajapriya, S. R. Rahul, P. Kumar, S. Polepally, R. Samineni, and D. G. K. Tej, "Performance enhancement and iot based monitoring for smart home," in *2017 International Conference on Information Networking (ICOIN)*, Jan 2017, pp. 468–473.

[4] D. H. Kang, M. S. Park, H. S. Kim, D. y. Kim, S. H. Kim, H. J. Son, and S. G. Lee, "Room temperature control and fire alarm suppression iot service using mqtt on aws," in *2017 International Conference on Platform Technology and Service (PlatCon)*, Feb 2017, pp. 1–5.

[5] L. Gregora, L. Vojtech, and M. Neruda, "Indoor signal propagation of lora technology," in *2016 17th International Conference on Mechatronics - Mechatronika (ME)*, Dec 2016, pp. 1–4.

[6] Y. Upadhyay, A. Borole, and D. Dileepan, "Mqtt based secured home automation system," in *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*, March 2016, pp. 1–4.

[7] H. C. Hwang, J. Park, and J. G. Shon, "Design and implementation of a reliable message transmission system based on mqtt protocol in iot," *Wireless Personal Communications*, vol. 91, no. 4, pp. 1765–1777, 2016.

[8] A. DelCampo, E. Gambi, L. Montanini, D. Perla, L. Raffaeli, and S. Spinsante, "Mqtt in aal systems for home monitoring of people with dementia," in *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Sept 2016, pp. 1–6.

[9] "SX1276 data sheet," Semtech Corporation. [Online]. Available: http://www.semtech.com/apps/filedown/down.php?file=sx1276.pdf

[10] "Python Client - documentation," Eclipse Foundation. [Online]. Available: https://eclipse.org/paho/clients/python/docs/

[11] "OASIS Standards Track Work Product, MQTT Version 3.1.1," OASIS, 2014. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf

[12] "LoRa Technical Specifications," LoRa Alliance. [Online]. Available: https://www.lora-alliance.org/

[13] R. Hsiao, D. Lin, H. Lin, C. Chung, and S. Cheng, "Integrating zigbee lighting control into existing building automation systems," 2012.

[14] R. K. Kodali and S. Soratkal, "Mqtt based home automation system using esp8266," in *Humanitarian Technology Conference (R10-HTC), 2016 IEEE Region 10*. IEEE, 2016, pp. 1–5.

[15] H. Zhu, Z. Pang, B. Xie, and G. Bag, "Ietf iot based wireless communication for latency-sensitive use cases in building automation," in *2016 IEEE 25th International Symposium on Industrial Electronics (ISIE)*, June 2016, pp. 1168–1173.

[16] J. Zhang, A. Huynh, Q. Ye, and S. Gong, "Reliability and latency enhancements in a zigbee remote sensing system," in *Sensor Technologies and Applications (SENSORCOMM), 2010 Fourth International Conference on*. IEEE, 2010, pp. 196–202.