A Multi-Protocol IoT Gateway and WiFi/BLE Sensor Nodes for Smart Home and Building Automation: Design and Implementation

Kanitkorn Khanchuea[1] Rawat Siripokarpirom[2]
Department of Electrical and Computer Engineering
Faculty of Engineering, KMUTNB
Bangkok, Thailand
e-mail: kanitkon.k@gmail.com [1]
e-mail: rawat.s@eng.kmutnb.ac.th [2]

***Abstract*—This paper presents the design and implementation of a multi-protocol gateway that relies on a multi-hop wireless network for smart home and building automation applications. This paper describes how to construct such a multi-hop tree-based wireless network to coexist with ZigBee mesh networks, using low-cost commodity 2.4GHz WiFi/BLE SoC modules. The gateway supports both wired and wireless connectivities to other sensor nodes, including the RS485/Modbus fieldbus and wireless networks based on two different protocols, namely the ESP-Now peer-to-peer wireless protocol developed by Espressif Systems and the ZigBee protocol. In this work, the ESP-NOW protocol was utilized to construct the core low-power multi-hop wireless network, whereas the ZigBee standard was used to build subnetworks of sensor nodes. A single-board computer (SBC), running an embedded Linux operating system, was chosen as a platform for implementing the proposed IoT gateway which utilized the MQTT protocol for message delivery. Field experiments were conducted to evaluate the performance of the ESP-Now multi-hop wireless network, comprised of up to 5 hops. To illustrate the functionality of the proposed gateway, a use case was presented, in which a building automation system prototype was developed for control and management of air-conditioners and AC power meter units.**

Keywords—IoT, Multi-Hop Wireless Networking, Multi-Protocol Gateway, Smart Home and Building Automation.

I. INTRODUCTION

One of the advantages of wireless networks over wired networks is that they have reduced installation costs and increased flexibility in relocation of network devices. In addition, connectivity range can be extended by using a multi-hop wireless network. However, when using wireless networks there are some issues to be concerned with, such as reliability, radio interference with other wireless networks, and security.

Recently, there are a growing number of IoT (Internet of Things) consumer products or devices existing in the market for smart home or home automation. Many of these devices rely on the WiFi standard such as IEEE 802.11b/g/n for infrastructure-based wireless networking; they can act as WiFi stations (STA mode) or access points (AP mode). They usually come with associated mobile apps or web apps so that the user can use a smartphone for device configuration or control. Due to its star (single-hop) network topology, the connectivity range is rather limited.

To overcome the limited connectivity range of a star (single-hop) network topology, multi-hop wireless networks can be used, e.g. ZigBee [1] and Z-Wave [2], which are mesh network protocols and utilize the IEEE 802.15.4 [3] personal-area network standard. Alternatives to both standards include 6LowPAN [4], Thread [5] and Bluetooth Mesh [6], which also support IPv6 over mesh networks.

An IoT gateway is a device or software component that bridges communication between sensing or controllable things and cloud-based systems over the Internet. There are likely to be a number of competing network standards for smart home or building automation applications. Therefore, IoT gateways should support coexisting network standards and protocols, whether they be legacy, mainstream or emerging solutions.

Over the past few years, the families of WiFi SoCs such as ESP8266 and ESP32 from Espressif Systems [7] have gained widespread popularity in IoT applications, thanks to their low cost, high performance microprocessors, low power design. ESP32 is an improved version of ESP8266, supporting both WiFi and Bluetooth LE (BLE) 4.2. ESP32 can be programmed using the open source ESP-IDF toolchain and is also Arduino-programmable, thanks to the development of Arduino core for ESP32. The ESP32 chip supports needed security features, like secure boot and flash encryption. There are various ESP32-based boards or modules available in the market. ESP32 is Cloud ready as it is supported by Microsoft Azure IoT and Amazon AWS FreeRTOS. Thus, ESP32 has its potential to be a good choice for industrial IoT applications.

To provide network connectivity, both ESP8266 and ESP32 are usually programmed to operate either in STA mode or SoftAP mode. Developed as an alternative to the standard WiFi, the ESP-NOW protocol offers a low-level adhoc network over 802.11 frames and enables multiple devices to communicate with one another without using a WiFi infrastructure (neither STA nor SoftAP), thus reducing power

978-1-7281-1921-2/19/$31.00 ©2019 IEEE

consumption of the system. ESP-NOW was first released in September 2015 and supported only the ESP8266, and was ported to support the ESP32 family later.

This paper seeks to address the following questions: Is it possible to use commodity WiFi/BLE SoCs such as the ESP32 family to build an infrastructure for wireless multi-hop or mesh networking ? How can BLE be utilized to help construct a self-forming multi-hop network ? How to integrate existing low data-rate, low-power wireless mesh sensor networks such as ZigBee into such a network ?

The remainder of this paper is organized as follows. Related work is presented in Section II. The system architecture and components are displayed in Section III. Implementation details are given in section IV. Section V describes the experimental setup and results. Section VI represents and discusses the conclusions and future work.

II. Related Work

WiFi and ZigBee are two preferred communication technologies for smart homes. Froiz-Míguez, et al. [8] provided a review of recent wireless technologies, hardware and software solutions for smart homes and presented a low-cost IoT Fog computing Home Automation System (HAS) that allows for carrying out seamless communications among ZigBee and WiFi devices for use in smart homes. They used Raspberry Pi SBCs as smart home controller and Fog gateways. The ESP8266-based NodeMCU boards equipped with XBee modules were as the ZigBee-to-WiFi gateway for wireless mesh sensor nodes.

Mikhaylov and Tervonen [9] proposed mechanisms for enabling multi-hop data transfers between the nodes in BLE networks. The BLE advertising packets were used in discovering neighboring nodes or routers. The proposed method was implemented on the top of the Texas Instruments' BLE stack targeting at CC2540 chips supporting the Bluetooth 4.0 standard.

S.Y. Ameen et al. [10] used the IEEE 802.15.4/ZigBee based wireless sensor network and investigated the effect of three different network topologies on the coordinator load, end to end delay, throughput and number of hops. According to the experimental results, it was concluded that tree routing was more suitable for WSN due to higher throughput, lower end-to-end delay and acceptable coordinator load when compared with star routing; and mesh routing caused lower delay and load, but lower throughput.

S.Soijoyo and A.Ashari [11] investigated the network metrics such as the average value of delay, throughput and packet loss when using three different network topologies of a ZigBee network comprised of XBee modules.

III. System Architecture

A. The System Components

The proposed architecture consists of the following components as shown in Fig. 1

1) MQTT broker and Web Server: A single-board computer (SBC) is used to implement an MQTT broker and a Web server. It provides the user with Web-based control panel, which is accessible within the corporate network. It is also responsible for message delivery and protocol conversion, communicating with the gateway within the same network. Outgoing messages from the wireless sensor network are forwarded by the gateway to the MQTT broker. Incoming messages sent from the web app via WebSocket to the MQTT broker, are forwarded to the gateway.

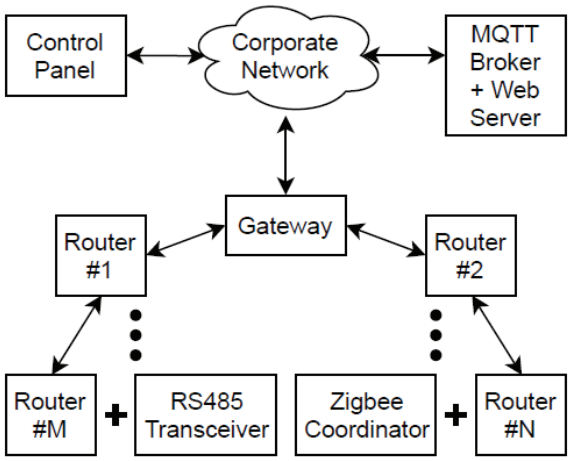

Fig. 1. System Architecture.

2) Gateway: The gateway, capable of both Ethernet and WiFi connectivity, acts as an MQTT client by subscribing itself to the MQTT broker on the SBC. In addition, it operates as the coordinator for the wireless multi-hop sensor network.

3) Router Node: Router nodes are used to form a self-organizing wireless multi-hop network. A router node may be equipped with either a RS485 transceiver module or a ZigBee coordinator module, both using the serial interface for data communication. Thus, router nodes provide access to other subnetworks.

4) ZigBee Sensor Node: ZigBee sensor nodes, configured as ZigBee routers, are deployed to form a self-organizing low-power wireless mesh sensor network built on top of the IEEE 802.15.4 standard. Each node can be equipped various types of sensors and actuators.

5) Control Panel: The control panel is a Web app running in a Web Browser such as Google Chrome. The user can view status and control the remote devices (e.g. air-conditioning units and power meter units). By using the WebSocket JavaScript library, the Web app can access the MQTT brokeron the remote machine.

B. ESP-NOW Peer-to-Peer Networking

The ESP-NOW protocol allows for sending and receiving payload data of length up to 250 bytes between two paired WiFi devices, by using vendor-specific action frames. According to the IEEE 802.11-2007, vendor-specific action frames are 802.11 management-type MAC frames, which are

used to perform supervisory functions (e.g. when joining and leaving wireless networks or access points). Before sending data using the ESP-NOW protocol, devices must be paired first. A device can manage a list of up to 20 peer devices. If data encryption is enabled, the number of paired devices is reduced to 10.

The implementation of the ESP-NOW protocol is supported by the ESP-IDF library. Examples of API functions for ESP-NOW are listed in Table 1.

TABLE I. ESP-NOW FUNCTION

Function	Description
esp_now_init()	Initializes the ESP-NOW driver.
esp_now_add_peer()	adds a device (specified by its MAC address) to the paired device list.
esp_now_send()	Sends payload data to a specific paired device.
esp_now_register_send_cb()	Registers a send callback function. After ESP-NOW data have been sent, the send callback function is called. The sending status can be checked for success or failure.
esp_now_register_recv_cb()	Registers a receive callback function. When receiving ESP-NOW data, the receive callback function is called.

IV. SYSTEM IMPLEMENTATION

The overview of the system implemented is shown in Fig.2

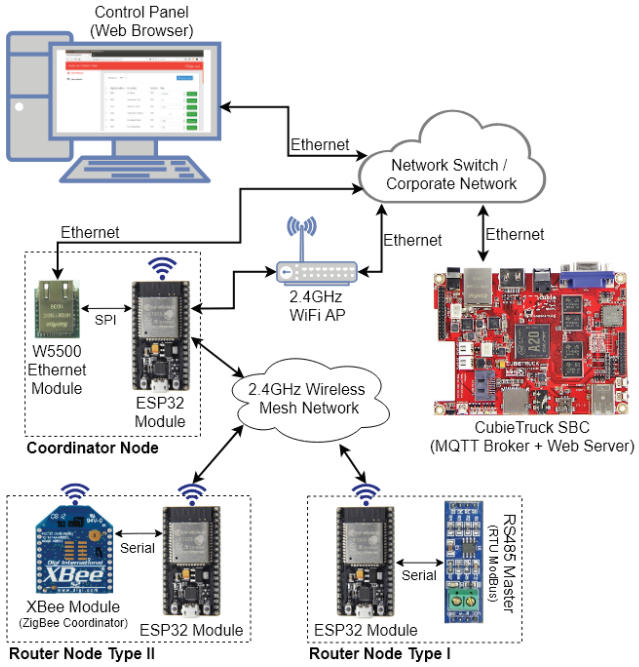

Fig. 2. Detailed System Implementation.

A. Detailed Implementation of System Components

1) MQTT broker and Web Server: There are many SBCs available in the market, e.g. Raspberry Pi 3, Orange Pi, and Asus Tinker. In this work, the CubieTruck SBC [12] was chosen to implement an MQTT broker (Mosquitto) [13] and a Web server. The CubieTruck SBC was equipped with a solid-state storage drive (250GB, SATA), installed with embedded Linux (Armbian-Ubuntu) as its operating system. It connects the network using the 1Gbps Ethernet interface.

2) Gateway: The ESP32 chip was chosen as the main processing and communicating unit. ESP32 is a WiFi/BLE chip with Xtensa dual-core 32-bit LX6 microprocessor, operating up to 240 MHz, and integrated with on-chip memory (448 KB for ROM and 520 KB for SRAM). The ESP32 chip has an additional hardware-serial interface, which can be used to access other modules at high baudrates without using soft-serial programming technique. The ESP32 boards with ESP-WROOM32 modules (ESP32-D0WD chip, 4MB on-board Flash memory) were chosen in this work.

There are two modes for interfacing the ESP32-based gateway to the local network.

Standalone gateway device: In this mode, the ESP32 module uses either the built-in WiFi transceiver to connect to an wireless access point or an external Ethernet module to access the network. The W5500 SPI-Ethernet (MAC+PHY) module or the LAN8720 RMII-based Ethernet (PHY) module can be used for this purpose. Note that the ESP32 has an integrated Ethernet MAC unit. The software network stack for ESP32 supports network programming based on the TCP/IP, HTTP/HTTPS and MQTT protocols.

USB-attached gateway device: In this mode, the ESP32 module is attached to a host computer or a single-board computer via the USB-to-serial port, operating at baud rates of up to 1Mbps. However, an additional software program (e.g. written in Python or Nodejs) is required in order to handle messages sent over the USB-to-serial port and forward them to the MQTT broker, and vice versa.

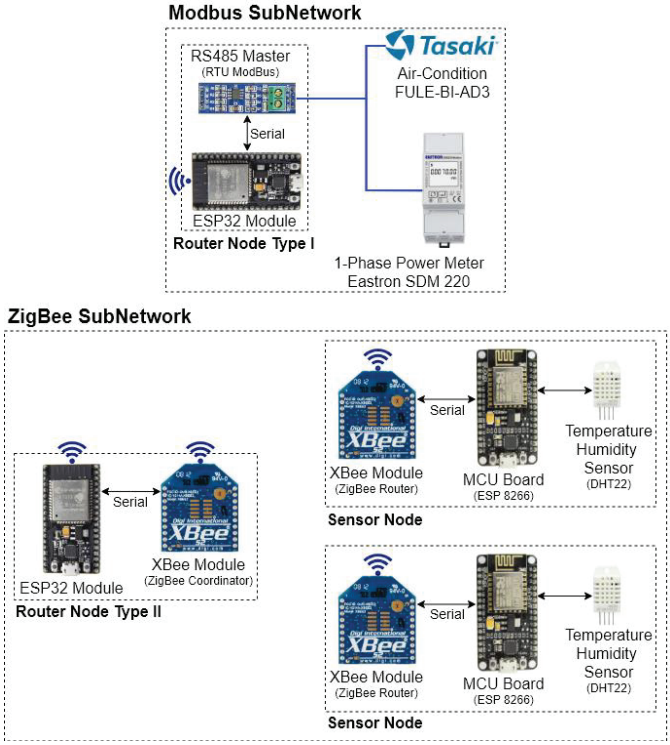

Fig. 3. Router Nodes.

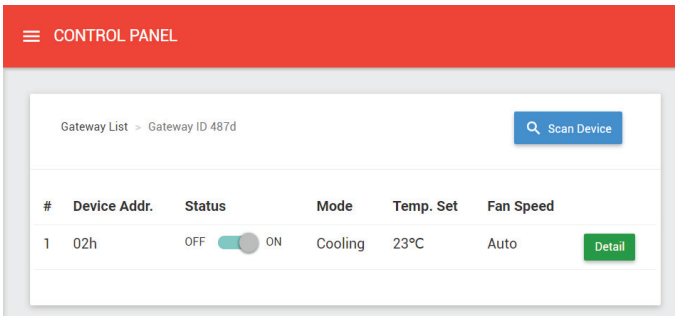

Fig. 4. Control panel web application list of Air condition devices.

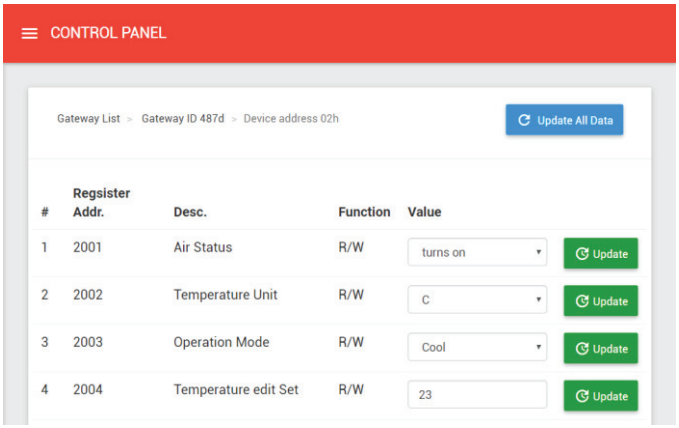

Fig. 5. Control panel web application list of regsiter values.

3) Router Node: The prototypes of router nodes were implemented using ESP32 modules. As shown in Fig.3, there are two types of router nodes proposed in this work. To support RS485/ Modbus RTU, router nodes can be equipped with an RS485 transceiver module, referred to as Type I. Router nodes of Type I were implemented to read or control Tasaki single-phase air-condition units and 1-phase power meter units (Eastron SDM 220 model) using the Modbus RTU protocol, as shown in Fig 6. Alternatively, to support ZigBee sensor network, router nodes can be equipped with a ZigBee module, referred to as Type II. XBee Series 2 (S2) modules [14] were chosen for this purposes. For both types, the hardware serial interface of the ESP32 was used for serial data communication. In case of ZigBee, each XBee module attached to a router node was configured as a ZigBee coordinator node for a specific ZigBee subnetwork (with a unique PAN ID).

4) ZigBee Sensor Node: The XBee S2 modules were configured using the Digi International's XCTU software, capable of operating in data transparent mode and self-forming a wireless mesh network. All XBee modules were configured as ZigBee mesh routers, except the XBee coordinator module attached to a router node. All ZigBee mesh routers communicate only with the ZigBee coordinator over other relaying ZigBee router nodes. For demonstration purposes, each XBee module was attached to an ESP8266 microcontroller module, allowing to read data sensors (e.g. DHT22, SHT31 temperature and relative humidity sensors) or to control actuators such as power relays.

5) Control Panel: The Web app for control panel was written using HTML5, CSS, JavaScript, Bootstrap, and Socket.io for the front-end development. The web server was written using in Node.js 6.x for back-end web development. Fig.4 and Fig.5 display screenshot photos of the control panel. The user can scan for devices and send a command message or update the status of the target device, which was an air conditioning system in this demo case.

Fig. 6. Router Node Type I.

B. *Multihop Tree-Based Network Model*

The multi-hop network topology used in this work is restricted to a tree-based model. The coordinator or the gateway is the root node, whereas routers are intermediate and end nodes in the network. The coordinator plays the master role in which it initiates the communication with a specific node in the network as the target. Each node in the network has a unique, pre-programmed node ID (16-bit), which is used to address the target node by the coordinator. The target node must send a response message within a specific time interval, back to the coordinator after it has received the request message and performed the required action.

To construct a self-organizing multi-hop network using the ESP-NOW protocol, neighboring devices must be discovered first and their MAC addresses must be determined so that they can be addressed to the peer list. One possible way is to configure all devices to operate in SoftAP mode, each with a specific SSID name that contains a prefix and its MAC address as a hex string. After scanning existing APs, the SSID names of found APs are analyzed to filter the MAC addresses of devices to be paired with. In addition, the received signal strength value (RSSI) of each found device is known. This method has a drawback in that the devices must operate in SoftAP mode, leading to higher power consumption. Another simple way is to broadcast a message to neighboring node using the ESP-NOW send method. However, in this method it is not possible to determine the RSSI value.

This paper proposes a better method for discovering neighboring devices by utilizing BLE advertising beacons. A beacon consists of the manufacturer ID (2 bytes), UUID data, service data, major and minor data. Each device can be programed to operate a BLE server sending BLE advertising beacons periodically. When receiving a BLE beacon message, the Bluetooth address which is related to the MAC address of the WiFi interface, and the RSSI value of the sending device can be determined. The RSSI values can be used to estimate the distance to neighboring nodes.

In addition, the BLE beacon sent from a router node merge this with the previous paragraph includes the number of active child nodes (*#ChildNodes*) and the number of hop levels (*#Hops*) in network tree of that node.

To start constructing a network, the coordinator sends broadcast messages periodically to neighboring nodes, using BLE beacons. If a node wants to join the coordinator, then it sends a join-request message to the coordinator. If accepted, the coordinator sends a join-accept message to the join node, which becomes a child node of the coordinator. Otherwise, a join-reject message is sent; this happens when the number of joined child nodes has reached the maximum value (set to 10). Any router node can have only one parent node and can later be the parent node of other nodes joining the network. When a router node is not associated with any parent node at initial state or when its parent node is not reachable, it starts BLE scanning process to receive information about existing router nodes in the neighborhood. A priority (p) value is assigned to each of parent candidates found and calculated using the formula below, where w_0, w_1 and w_2 are weight numbers and set equal to 1/3 as example. The maximum number of child nodes ($\#ChildNodes_{max}$) is set to 10 and the maximum number of hop levels ($\#Hops_{max}$) is limited to 8 as example. The candidate node with the highest priority value will be selected first.

$$p = w_0\left(\frac{\#ChildNodes_{max} - \#ChildNodes}{\#ChildNodes_{max}}\right) + w_1\left(\frac{RSSI_{max} + RSSI}{RSSI_{max}}\right) + w_2\left(\frac{\#Hops_{max} - \#Hops}{\#Hops_{max}}\right) \quad (1)$$

Routers are intermediate nodes which are responsible for relaying or forwarding messages from the coordinator to the target node. When a router has received a message from its parent node, it must forward the received message to all child nodes. If a parent node cannot send the message to a child node after a specific number of retries, that child node will be removed from the list. In other direction towards the coordinator, if a child node cannot send the message to its parent node after a specific number of retries, it will attempt to join another node.

C. The Message-Level Communication Protocol

Fig.7 shows the structure of the messages which are sent from the coordinator to a specific router node in the network, and vice versa. There are three types of messages, namely the network management, the ZigBee and the RS485/Modbus types. The first byte specifies the message type. The next two bytes specify the unique ID (16-bit) of the target router node. In case of ZigBee, this node ID is the same as the PAN ID of the associated ZigBee subnetwork. For the Modbus RTU, the first three bytes of the message will be removed at the target router node. The rest of the message is format-converted and then sent to the Slave device, starting with the slave address byte, followed by the function code, value type and access type according to the Modbus RTU protocol.

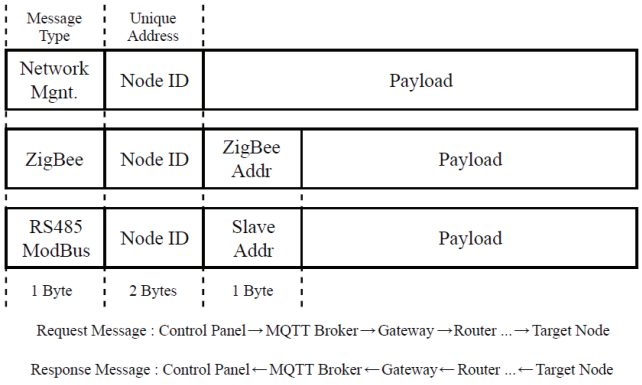

Fig. 7. Structure of Messages.

V. Expreimental Setup And Results

To evaluate the performance of the multi-hop wireless network using the ESP-NOW protocol, two experiments were conducted, using a linear multi-hop network topology. All experiments were carried out in office environment and the Tx power levels of all router nodes were set to 19.5 dBm.

In the first experiment, the distance between two adjacent nodes was fixed to 4 meters, but the number of hops was varied, from 1 to 5 router nodes. Network metrics such as throughput, round-trip time (RTT) and delay variations (jitter) were investigated when testing with different packet sizes (30, 60, 120, 180 and 240 bytes long) as shown on the x-axis in Fig.8 and 9. For each test run, 100 packets were sent from the coordinator and forwarded by intermediate nodes to the end node, and sent back in reverse order back to the coordinator.

In the second experiment, the number of hops was fixed to 3 router nodes, separated by an equal distance from each other. In order to study the impact of the node distance on network metrics as mentioned before, the distance between two adjacent nodes was varied, starting from 2m, 5m, 10m and 15m. The same Tx power level was used for all nodes.

The experimental results in Fig.8 show that, as the number of hops increased, both average round-trip time and jitter increased (measured in milliseconds), whereas the throughput (kbps) decreased. Fig.9 visualizes the results of the second experiment. Increasing the node distance led to a decrease in the throughput and an increase in the round-trip time. The experimental results in Fig.8 show that, as the number of hops increased, both average round-trip time and jitter increased

(measured in milliseconds), whereas the throughput (kbps) decreased. Fig.9 visualizes the results of the second experiment. Increasing the node distance led to a decrease in the throughput and an increase in the round-trip time.

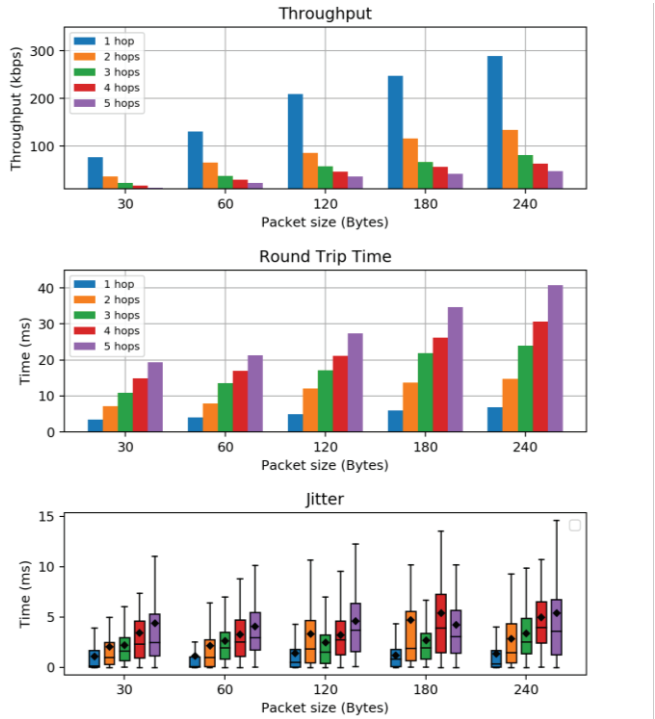

Fig. 8. ESP-NOW Mulit-Hop First Experiment results.

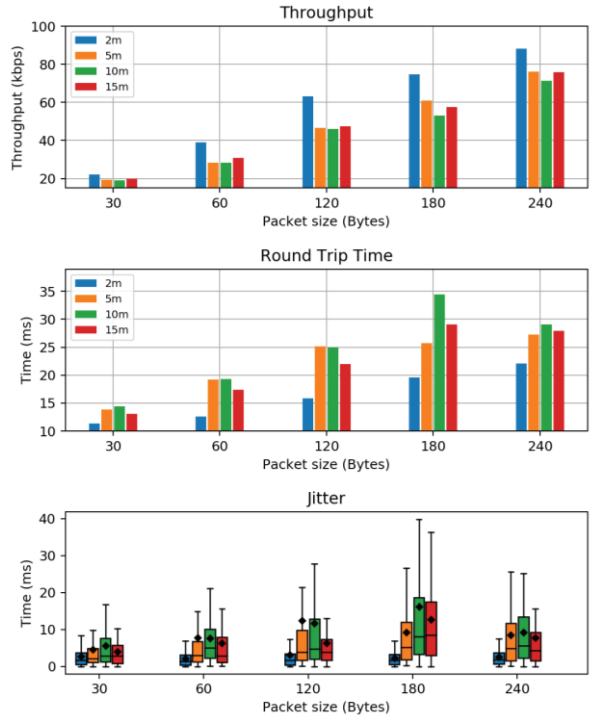

Fig. 9. ESP-NOW Multi-Hop Second Experiment results

VI. Conclusion And Future Work

In this paper we presented an approach to design and implement a multi-hop tree-based wireless sensor network and a multi-protocol gateway for smart home and building automation applications, using the ESP32 family of WiFi/BLE SoC chips and the ESP-NOW IEEE 802.11-based peer-to-peer protocol. We also proposed to utilize the BLE features of the ESP32 chip to help in the process of self-organizing a wireless network. A system prototype was built to demonstrate a use case scenario in which the proposed architecture was deployed to support control and management of air-conditioning units and power meter units. Field tests were conducted to study the impact of the number of hops and distance between hops on network metrics such as average round trip time, throughput and jitter. However, only a simple linear network topology was used and the number of nodes was rather small (limited to 6 nodes). We concluded that the ESP32 WiFi/BLE chip, in combination with the ESP-NOW protocol, represent a good choice for building wireless multi-hop, self-organizing, low-power sensor networks. As future work, it is possible, for example, to extend the proposed architecture with Mesh network routing capability. However, further studies and implementations are necessary and still in progress.

References

[1] Zigbee Alliance, “ZigBee Specification,” ZigBee Document 05347r17, Jan. 2008, pp. 1-576.

[2] Z-Wave Alliance, “Z-Wave Device Class Specification,” Z-Wave Document SDS1042, Mar. 2018, pp. 1-195.

[3] IEEE 802 Working Group, “IEEE Standard for Local and Metropolitan Area Network – Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs),” IEEE Std 802.15.4TM-2011, Jun. 2011, pp. 1-314.

[4] Kushalagarent, N. al. “IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem, Statement and Goals”, IETF RFC 4919, Aug. 2007

[5] Thread Group, https://www.threadgroup.org. (Access : Jan 21 2019)

[6] Bluetooth SIG, “Bluetooth Mesh Networking”, Bluetooth SIG Document, Jul. 2017, pp. 1-28.

[7] Espressif System, https://www.espressif.com/. (Access : Jan 21 2019)

[8] Iván Froiz-Míguez, et al. "Design, Implementation and Practical Evaluation of an IoT Home Automation System for Fog Computing Applications Based on MQTT and ZigBee-WiFi Sensor Nodes", Sensors 2018, 18(8), 2660.

[9] Konstantin Mikhaylov and Jouni Tervonen, "Multihop data transfer service for Bluetooth Low Energy", Proc. of the 13th International Conference on ITS Telecommunications (ITST), Tampere, Finland, 2013, pp. 319-324.

[10] Siddeeq Y. Ameen, et al. "Coordinator and Router Investigation inIEEE802.15.14 ZigBee Wireless Sensor Network", December 2013

[11] Sigit Soijoyo and Ahmad Ashari, "Analysis of Zigbee Data Transmission on Wireless Sensor Network Topology", International Journal of Advanced Computer Science and Applications, 8(9), 2017, 145.

[12] Cubie Board, Cubietruck, http://cubieboard.org/tag/cubietruck/. (Access : Jan. 19 2019)

[13] Mosquitto project. www.mosquitto.org. (Access : Jan 18 2019)

[14] Digi, Digi XBee / XBee-PRO ZigBee Modules (S2), https://www.digi.com/support/productdetail?pid=3430. (Access : Jan. 19 2019)