Development Of A Freertos Based Novel Gateway For Iot Applications

Giang Truong Le, Thang Viet Tran, Phuc Thinh Doan, Trong Hai Nguyen

Abstract: A novel radio frequency (RF) gateway is designed and implemented by using an open-source real-time operating system for wireless communication between sensor node and server over the internet network. The proposed RF gateway consists of a Dual-core microcontroller unit, a wireless transceiver module that operates at a frequency of 433 MHz and a local user interface website configuration. By using three low-power wireless sensor nodes, we can experiment with the designed RF gateway, that is, connected to commercial environment sensors and collecting data such as water temperature sensors. The proposed RF gateway can simultaneously execute multitasking with a set of different priorities and in particular, it can collect and store measured sensor data in 30 days, and as a server transferring measurement results from nodes to an IoT dashboard by using Websocket protocols. In addition, a friendly portable interface website is designed to manage and illustrate parameters and as control wireless actuator node in offline mode. This system was implemented by using an open-source real-time operation system (FreeRTOS) and a single high-speed multicore microcontroller unit that is a promising option for most of the applications in the Internet of Things.

Index Terms: FreeRTOS, Wireless sensor network, Internet of Things, IoT application, Sensors, Microcontroller, Web IoT.

1 INTRODUCTION

MANY billions of applications run on the internet network that the conduct of electronic devices, sensors and a microcontroller unit, and most of them are used to monitor ambient parameters and in a number of applications. In general, there are studies show that existing architectures in IoT applications in two types 1) IoT devices connect directly to cloud server via internet network; 2) another, this is, illustration in a real-time IoT system [1], in which by using an RF gateway that allows exchange IoT information between a wireless network and the internet network, and which one we focused on the experiment in this paper. Existing gateway architectures are not easily adaptable to real-time communication message requirements [2], [3], [4] and even though a server-side has been supported to engineers, developers a lot such as friendly interface configuration [1], [4] and without requirement advance backend setting, network programming skills at server side. In basically, the wireless sensor network (WSN) is based on physical sensors, a microcontroller unit and a wireless communication standard. With the rapid development of many wired and wireless existing technologies to transfer data from the multiple sensor nodes that designed for IoT applications (using 3G/4G networks, Bluetooth, Zigbee, or cable), these methods are associated with prohibitively high costs, making them impractical for research projects or real-life applications. Others, most of these technologies are often difficult to configure advanced parameters such as indicate receive signal (RSSI), power RF transmission, data packets collision, encryption/decryption algorithms as well as these solution increases the cost of the network infrastructure. In particular, the area of buildings, homes, small farms may not need these technologies.

- *Giang Truong Le is currently working as an executive researcher in Department of Science and Technology, Nguyen Tat Thanh University, Ho Chi Minh City, 70000, Vietnam,PH-(+84)969875864. E-mail: letruonggiang2211@gmail.com*
- *Trong Hai Nguyen is a lecturer at HUTECH Institute of Engineering Ho Chi Minh City University of Technology (HUTECH), Ho Chi Minh City, Vietnam, PH-(+84)913728234. E-mail: nt.hai75@hutech.edu.vn*

To reduce the complexity of these issues, the experimented system has been used a wireless transceiver module to collect the measured result from sensor nodes and manage wireless actuator nodes at a frequency of 433MHz. The wireless transceiver will be detailed in section 2. Nowadays, the establishment of wireless data communication is very common and easy to implement with the new available technologies. The rapid development of the IoT as above mentioned and the development of the open hardware platforms such as Arduino boards, Raspberry, ESP32, etc., provides low-cost LoRa, Wi-Fi, Bluetooth, or Zigbee modules to setup wireless data devices all in one with multiple functionalities not only engineers, researchers but also hobbyists, or even college students count on many electronic devices to create wireless networks easily. However, the main challenge is not only to establish a wireless connection but to make it works under special environmental conditions for years, without any maintenance [5], [6]. In addition, the wireless connection has to work in the best but also in the worst conditions of line of sight. As the number of sensors in an IoT system works, however, the issue of how to transfer data between those devices and a wireless gateway becomes more complex and in special control data transfer needs must be ensured with operating considerations and infrastructure costs. In some specific requirements, a real-time protocol may become important than ever seeing (e.g: self-driving cars, robots, airplane monitoring systems, and video conferencing). Therefore, a number of researchers are still focusing on the speed of data rate [4], [7], [19] and real-time operating systems (RTOS). Unlike RFID protocols that presented in [8], the wireless data packets collision because many sensor nodes can be required simultaneously to exchange data. Thus, several topologies on the wireless network can be used to solve this issue. Among those existing wireless technology, Lora has been become an excellent candidate for a small application in IoT with its advantages as such low-power transceiver, supporting mesh, ring, and star topology which we will present in the experiment part [9], [10]. In contrast to existing wireless communication technologies currently used for indoor IoT sensors, several techniques have been proposed for providing long-range and outdoor wireless communication, such as Sigfox [11], LoRa/LoRaWAN [12], NB-IoT, and LTE-M [13], [14]. Among these, LoRaWAN, which is based on LoRa physical layer (PHY) [15] implementation, is a network standard for telecom operators. LoRaWAN uses a star-network topology for communication between LoRa gateways and IoT devices; only one hop is allowed between a gateway and a LoRa device. Recent, in an experiment and the design, carried out drawbacks of LoRaWAN are similar to those of the ALOHA [15, 16, 17] protocol, as there is no mechanism to arbitrate access to a shared wireless frequency. Transmissions by more than one node (transmitter) at the same time may result in data collision. LoRaWAN uses this approach to simplify the design of the media access control (MAC) layer for a battery-powered LoRa device, which is required for saving valuable energy, thus prolonging its lifetime. Therefore, a LoRa device can transmit data to the gateway at any time, causing significant packet collision if many devices transmit the signal simultaneously. Typical RF gateways have been used application scenarios are not only seen in smart homes but only in agricultural data collection applications and healthcare. In specially, an RF gateway can be used in places where multiple sensors, groups of devices, vehicles are on a monitor network. The gateway on the IoT architecture can be a fixed, dedicated device, such as the Samsung Smart Things Hub [11]. In the alternative, it can be a mobile device, shared device, like the functions of a smartphone. In this case, data can be collected either from the mobile device's internal sensors, for example, fingerprint sensor, microphone, built-in cameras (front and back cameras), GPS (Global Positioning System), accelerometer, magnetometer, gyroscope, proximity sensor, or light sensor) or through external sensor devices, which can be connected to the mobile device using wired (e.g., USB cable) or wireless technologies (e.g., Bluetooth, RFID, Infrared, etc.). Besides, some real-time systems in hospitals in form of measurement machines, medical instrument equipment, imaging system, sensors may be placed on the user's body (e.g., EGG, heart rate sensors, pressure sensors [4], [9], [10], [12]) or in the environment (e.g., air – temperature and – humidity sensors) [1]. There are also a large number of machines based on real-time systems such as self-driving cars, video conferencing, and industrial robots but never less are still important in our daily life. It is clearest that in recent real-time systems have become ubiquitous and have permeated a large number of application areas. The main objective of the study presented hereby is to have a reliable data communication solution that allows us to remotely have collected the measured data and develop an RTOS application. Presently, Amazon Group has been developed and contributed an open-source Amazon RTOS [3] for IoT applications that provide extra software libraries to the easy program to most of the small microcontroller units, which have resource-constrained. In the studies [18] and [19], the authors presented requirements for real-time system analysis of data gathered by IoT devices to develop further knowledge on the observed process. Such a processing step can also be integrated into adaptive systems to control the system characteristics based on patterns observed in real-time. However, the authors state that the requirements for a network supporting such analytics are currently not considered by most networking-oriented research. In addition, the limited documentation available on the internet and the implementation process of classes inside the package are not obvious. To archive and solve these issues, this paper illustrates a different wireless gateway for wireless IoT applications by deploying a star topology sensor network at a frequency of 433 MHz and with a Dual-core microcontroller based on an open-source RTOS library. This paper is organized as the following construction. In section 2, we introduce the system design of the proposed elements with five important sub-sections 2.1, 2.2, 2.3, 2.4, and 2.5. The portable website interface and implementation diagrams will illustrate in section 3. Next, the experiment results and discussion of this work will present in section 4. Finally, the conclusion will discuss in section 5.

2 SYSTEM DESIGN

This section describes the proposed system used to monitor temperature from sensor nodes in more detail and the flow chart of the proposed elements.

2.1 Overview of the Implemented System Architecture

The main idea of the proposed system is to enable wireless Web-based interactions on RF gateway FreeRTOS based and use Websocket-server for monitoring and controlling the remote applications in IoT. In Fig. 1 shows the proposed implementation for monitoring system by using wireless sensor network (WSN). The low-power wireless transceiver module RFM69HCW is used to communicate data in the proposed system and a star topology is used to monitor and control radio frequency node (RF). The implemented system that FreeRTOS-based RF gateway for executing tasks with fixed difference of priority levels. The components of experimenting system consists of three sensor nodes that communicate as a wireless sensor network at a frequency of 433MHz a wireless gateway, and a remote server. On the remote sever side, an IoT dashboard is also developed for management sensor nodes and as well as the RF gateways. However, we will briefly introduce this in section 4, but not in deeply. Seeing in the Fig. 1 that the RF gateway has important role in which it collects data from sensor node and forward data to a remote sever through the internet network. On the other hand, RF gateway receives command from remote server and then broadcast those commands to control node (actuator node). The designed RF gateway is not only connect to a commonly router WiFi at a frequency of 2.4 GHz, but it also connect to internet network by using an internet cable via physical LAN port on a router WiFi. To exchange the data to a remote server the RF gateway used a specific bi-direction protocol of Websocket, named SocketIO. In this paper, we focus on the RF gateway FreeRTOS-based, therefore SocketIO will be briefly introduction in sub-section 2.5 Websocket and explained the flow chart of the SocketIO protocol. For the local web interface on the proposed gateway is to log the data sensor in offline mode and as for real-time monitoring data sensor in online mode (connected to internet network success). Next, we will show detailed to main flow chart execution of the proposed RF gateway in sub-sections.

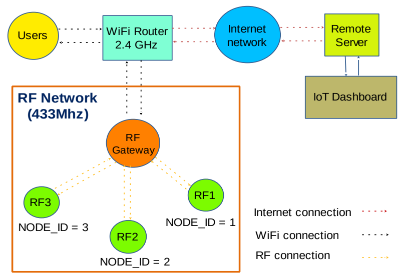


Fig. 1. The system proposed and implemented for monitoring environment data and wireless control actuators; the proposed RF1 and RF2 are sensor nodes, RF3 is a actuator node (RF3 has 4 relays for controlling high-voltage actuator such as lights, motors)

2.2 The Proposed Gateway Communicates with RF nodes
In Fig. 2, we show the block diagram of the design of the RF gateway. There are two main components of the system are consists of an ESP32-WROOM-32U unit [27] that is a low-cost system on a chip with built-in antenna Wi-Fi (WLAN network) and a transceiver module RFM69HCW. ESP32 is a powerful dual-core processor and it may be an excellent candidate with its features as mentioned in the introduction section and it has been provided 3.3Vdc volts to GPIOs the most important feature to adopt with the wireless transceiver. Another reason to choose ESP32 is not to integrate multiple modules but also provides two SPI ports with high speed and allows communication with many IoT devices.

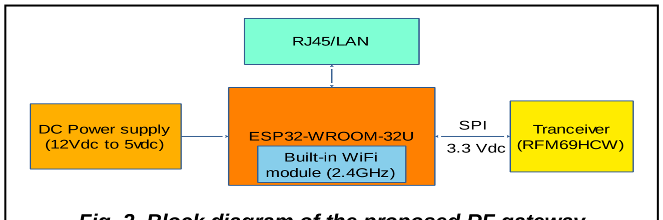

RJ45/LAN
DC Power supply (12Vdc to 5vdc)
ESP32-WROOM-32U
Built-in WiFi module (2.4GHz)
SPI
3.3 Vdc
Tranceiver (RFM69HCW)

Fig. 2. Block diagram of the proposed RF gateway

In [9], the authors have studied and compared several topologies of WSN. The analysis results carried out that how the choice of communication topology and has strongly recommended that the network should be designed as a star network with a data flow to the gateway. In Fig. 3, we introduce the general flow chart of the proposed RF gateway with setup functions and creation tasks necessary.

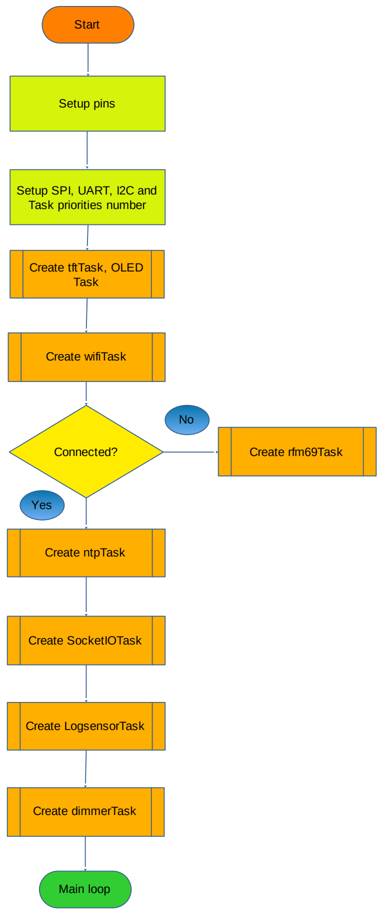


Fig. 3. Flow chart of the proposed RF gateway. After power-up, this is only run once at the start-up state.

In the setup function, the ESP32 microcontroller checks the connection to a router WiFi. Then it will create tasks that they depend on the internet connection and if the WiFi connection is available. Such as NTPTask requires exchange information to NTP server and then synchronous starting time, SocketIOTask will establish a connection to a remote server over the internet network. On the opposite, if the WiFi connection is a failure, the ESP32 microcontroller then gives up those tasks relevant to the internet network. In this situation, the rfm69Task is still created and run in an infinitive loop with its priority.

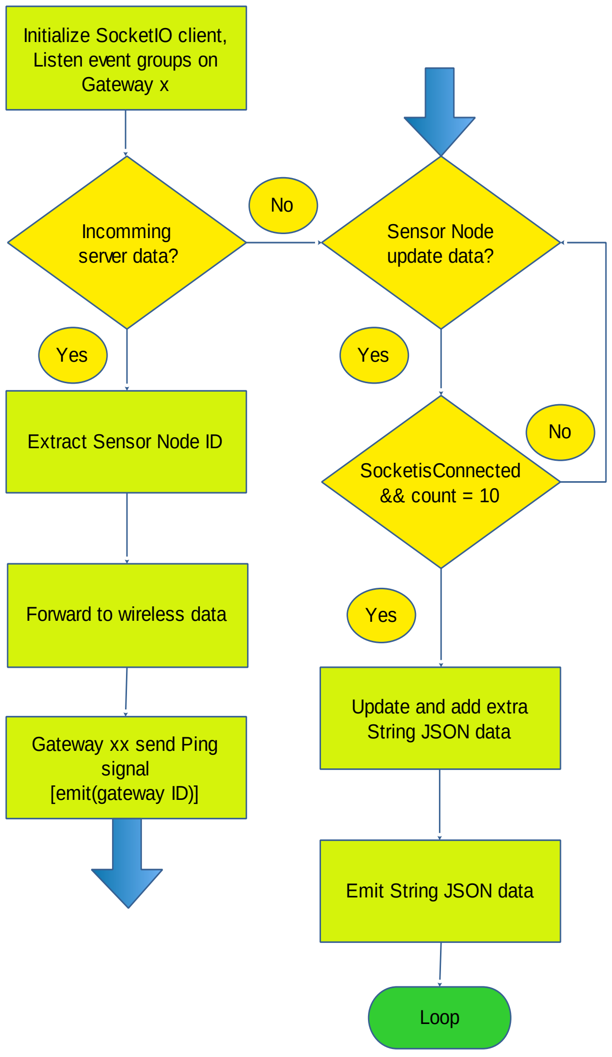

Initialize SocketIO client, Listen event groups on Gateway x
Incomming server data?
No
Sensor Node update data?
Yes
Yes
Extract Sensor Node ID
No
SocketisConnected && count = 10
Forward to wireless data
Yes
Gateway xx send Ping signal [emit(gateway ID)]
Update and add extra String JSON data
Emit String JSON data
Loop

Fig. 4. Flow chart of SocketIoClient function of the proposed RF gateway to communicate with a remote server by using JSON data package

First of all, the design of flow chart multi-tasking supports for our RF gateway is illustrated in Fig 3. In general, the first working phase of the program is executed when a +12Vdc (as shown in Fig. 2.) power supply to the RF gateway, or a reset event of the proposed gateway is carried out. At this state, after configuration GPIOs and UART baud rate to communicate with Arduino IDE terminal, the following tasks are created: display task (tftTask, OLEDTask), WiFiTask. Then if the establishment of the proposed RF gateway to router WiFi is a success, it will create four different tasks that they consist of ntpTask, SocketIOTask, LogsensorTask and dimmerTask. In case, if the establishment of the designed RF gateway to a router is not. It will only create the rfm69Task (see Fig. 3) that the designed RF gateway will fall into offline mode. Once the first phase is done, the microcontroller has to perform cyclically with their priority levels. The consideration of priority levels of sub-tasks will be introduced as in Table 1 and discuss in a later section. In addition, the microcontroller will check the internet connection every one minute by using a ping function to a remote server. If the internet connection is available, the microcontroller will disable offline mode to switch into online mode. The online mode means the RF gateway will establish a connection to the server by using SocketIO protocols. The SocketIO protocols will discuss in sub-section 2.5. Next, in Fig. 3, some of the blocks are simple tasks such as tftTask and OLEDTask that transfer information to those display devices and it is not difficult to deploy. Therefore, we will focus on and introduce two of the important tasks included SocketIoTask and rfm69Task with priorities of 8 and 7, respectively. In similarly, others tasks are commonly in hardware programming and we will do not deeply introduction in this paper.

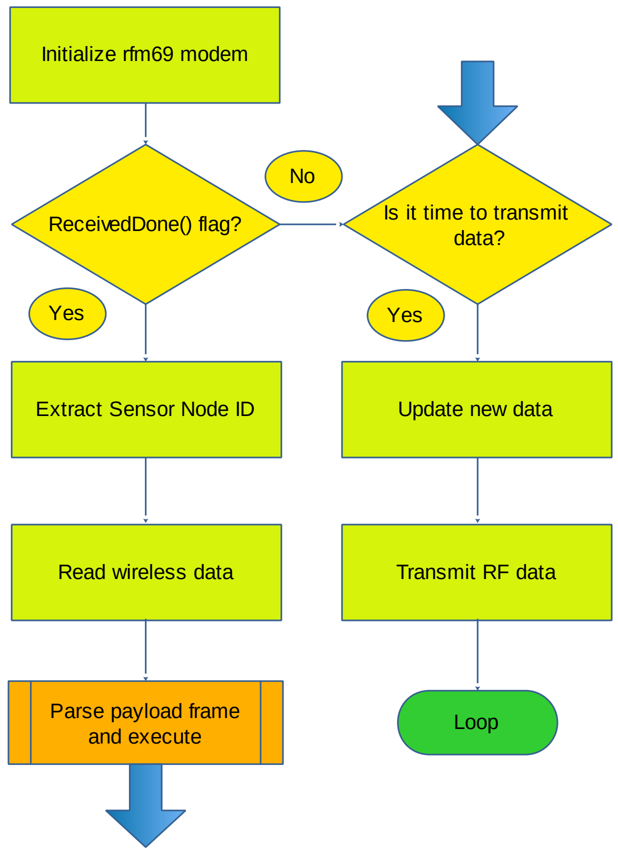


Fig. 5. Flow chart of rfm69Task of the proposed RF gateway; this is used to send command to actuator node and monitor the measured sensor data from the sensor node.

Secondly, rfm69Task is developed to communicate with sensors nodes in wireless link and control nodes by using a customized data frame as illustrated in the below section (see Fig. 10). The flowchart of rfm69Task is shown in Fig. 5. Firstly, SocketIoTask programs are to exchange all data packets between the proposed RF gateway, it can send out JSON data for requesting a connection to the remote server (see Fig. 1.), At the remote server-side, it provides event over Websoceket library and needed parameters (eg. user, password, port connection, domain name or static IP address). To manage the RF gateways, the server has created a list of gateways and it is declared empty in the initial state. This list is held by RF gateways parameters and will store these parameters in structure format on a database of a remote server. The processing of SocketIoTask illustrates as shown in Fig 4. Another demand of this task is forwarding a specific command in JSON format to control the actuator nodes that we will present in sub-section 2.4 with the title Wireless Sensor Node. A customized data package is presented in 2 layers included Layer 1 shows the general wireless data package size of a transceiver RFM69HCW and the more detailed payload message in bytes is shown in every single field of Layer 2. There are a total of 16 bytes in the length of this wireless package and it is used to exchange data between the designed RF gateway and nodes. To collect sensor data from the sensor node is a key function of rfm69Task. The sensor nodes can use the data structure packet to respond to a request from an RF gateway. Notice that the data packet has 4 bytes in the length of the Sensor data field. On the RF gateway side, it received this data package with Node-ID address field and identification. For broadcasting wireless data to control nodes, the RF gateway used a specific wireless data frame as shown in Layer 2 (Sensor commands field, 4 bytes in length). After the RF gateway received a command in JSON format from the server-side through SocketIoTask and then begins parsing the received data, and finally it is broadcast to control nodes with definition node address (Node ID field in Fig. 10, Layer 2). In short, the purpose of rfm69Task is to handle events that receive and broadcast wireless data by the RFM69HCW transceiver. Fig. 5 has been shown more detail about the processing sequence of the rfm69Task that is used to collect measured data from sensor nodes and broadcast wireless data package to control nodes. This task has a lower priority than controlRelayTask and a greater priority than SocketIoTask as illustrated in Table 1.

2.3 Gateway Firmware Design RTOS Based

The main firmware of the proposed RF gateway is illustrated in Fig. 3 that can process multitasking with their preemptive priority levels as illustrated in Table 1. In sub-section 2.3.1, we focused on rfm69Task and SocketIoTask and explained these two sub-tasks with the flowcharts shown in Fig. 4 and Fig. 5. Notice that two main issues need to be considered: the priority levels of sub-tasks and the connection to the internet network of the designed gateway.

Next, the consideration of Task Control and Task Priority Levels which is based on fixed-priority preemptive scheduling will discuss.

1. Task control and task priority levels

A separate task is created for each part of the system that can be identified as being able to exist in isolation, or as having a particular timing requirement.

In the task function, it requires making other system calls related to software events or timing. Many tasks will wait for a particular type of event and do something in response. Some may generate a software event. Others may do something, wait 100ms, and then repeat.

Processor utilization is automatically switched from task to task on a most urgent need basis with no explicit action required within the application source code.

TABLE 1
FUNCTIONS TASKS AND PRIORITIES OF THE RF GATEWAY IMPLEMENTED

Task name	Executive Core id	Priority	Stack size
ControlRelayTask	1	9	1000
rfm69Task	1	8	1500
SocketIoTask	1	7	1500
SensorNodeTask	1	6	1500
LEDTask	1	5	1000
NTPTask	1	3	1500
WiFiConnectionTask	0	2	2000
WebserverTask	1	1	3000

Another common approach to developing the firm of the proposed RF gateway is the period transformation technique [17, 18] in which separates into sub-tasks therefore we can reduce the complexity of tasks and increase the number of sub-tasks.

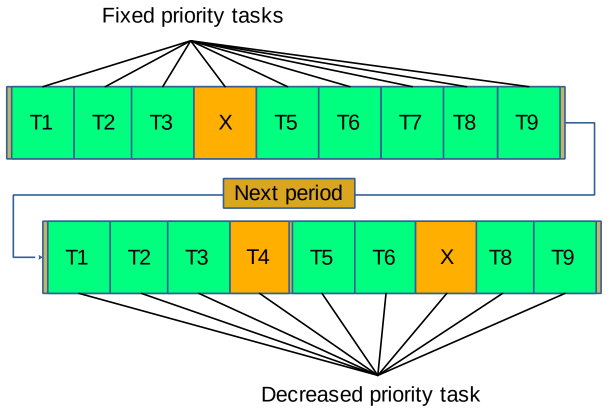


Fig. 6. Experiment method: an example scenario of a deadline missed handling

Task priorities can be set in a variety of ways even randomly and it is not difficult to reason randomly selecting task priority levels. However, a dynamic priority algorithm is an optimal way to ensure key task deadlines are always studied [20] that is focused on the closer a task to its deadline, the higher its priority. Two authors, Liu and Layland are first studied the rate monotonic (RM) algorithm that gives the highest priority to the periodic task with the shortest deadline. Another disadvantage of RM is difficult to support aperiodic and sporadic tasks [17]. Liu and Layland [21], the authors pointed out RM is not optimized when task periods and deadlines differ [17], [20].

Based on the paper of Reinder and Jim [20], the authors provided an extract equation to calculate priority levels and discussed fixed priorities. The worst-case response time of a task occupied that a task used the length of the longest interval from a task's release till its completion.

Seeing in Table 1, the priority level of number 4 is not presented that they can be used to decrease the priority of a task. In order words, a task has missed its deadline it will be decreased by its priority level task. An example scenario of deadline miss handling is described in Fig. 6.

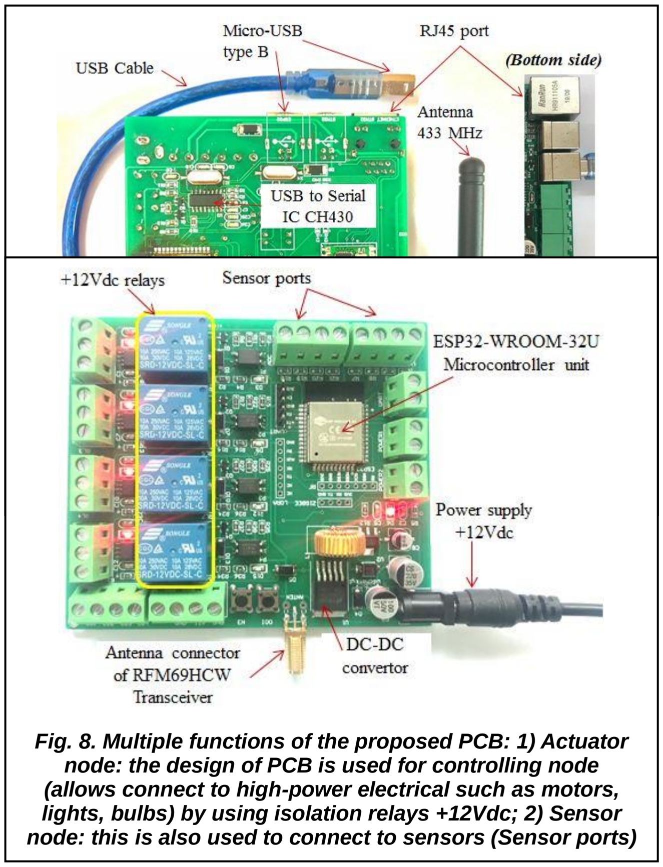


Fig. 8. Multiple functions of the proposed PCB: 1) Actuator node: the design of PCB is used for controlling node (allows connect to high-power electrical such as motors, lights, bulbs) by using isolation relays +12Vdc; 2) Sensor node: this is also used to connect to sensors (Sensor ports)

In Table 1, we illustrate the task's stack size in which based on an experiment of the proposed RF gateway and investigation from some studies [4], [17], [20], [25]. The priority levels considered to the avionic mission of the proposed RF gateway.

2.4 Wireless Sensor Node

In this sub-section, the RFM69HCW module, a wireless transceiver module has been used to exchange data between sensor nodes and the proposed wireless gateway. The sensor nodes are deployed at the fixed place as per their utilization and requirements. A prototype PCB was designed which has both functions as such sensor node and control node and it is shown in Fig. 8. Noticed, sensors can connect to sensor ports and can be easily extended by using a standard serial communication I2C as well as a 1-wire standard in which we experimented in this paper.

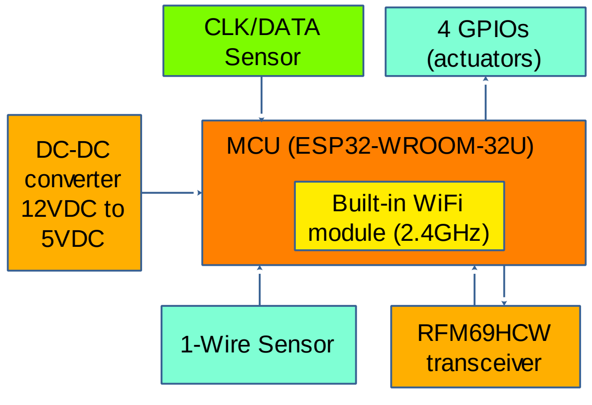


Fig. 9. The block diagram of the proposed sensor node and control node

In Fig. 9, the block diagram of the proposed sensor node that is consists of an ESP32 microcontroller, a DC-DC converter, an RFM69HCW transceiver, and four channels relays. The sensor node can communicate with an RF gateway by using functions of the RFM69HCW module. In case, the prototype node was also configured as a control node that it used 4 channels driver to control a high-power machine such as bulbs, motors, humidifiers. Besides, a proposed firmware of the sensor node was developed and is shown in Fig. 11. The users can reconfigure the sensor node prototype that it can become a sensor node or a control node by using special commands and the serial port of ESP32 microcontroller with baud rate at 115200 bps (bit per second), and Arduino development environment or any serial debug software. We have been implemented the development of firmware of the RF gateway by using the FreeRTOS library and a Dual-core microcontroller ESP32-WROOM-32U. The designed RF gateway is used to communicate with nodes. And sensor nodes collect the environmental data from sensors and transfer the measured data to the wireless gateway using the standard transceiver module RFM69HCW and then the gateway will transfer sensor data to the server-side by using SocketIO protocols.

1. The proposed RF package for sensor node

RFM69HCW has an important role in the proposed system in which it allows sensor nodes to communicate the sensor data with RF gateway, RFM69HCW can operate at an ultra-high frequency of 433 MHz and can send three different packet types: 1) packets with a theoretical unlimited length payload in which there is no length or CRC field, 2) packets with variable length payload and CRC, and 3) packets with a fixed-length payload that have CRC but no length field communicating devices agree on payload length separately, which we introduce in this sub-section. Fig. 10 illustrates more details about the packet structure used by the RFM69HCW to deliver fixed-length payload data

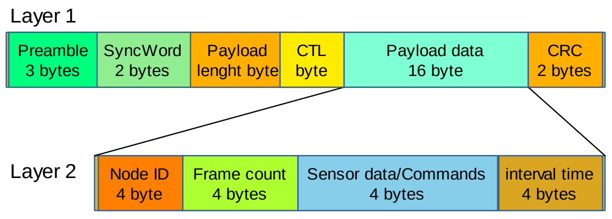


Fig. 10. The customized data structure of the proposed RF gateway for monitoring sensor node and control wireless actuator node. To collect data from sensor nodes, a sensor data field with 4 bytes in length in Layer 2 is used and the opposite, Commands will be used to control the actuator

All packets were sent through the RFM69HCW transceiver consist of the following fields: preamble, sync word (used as a secret key), payload data, and optional CRC bytes.

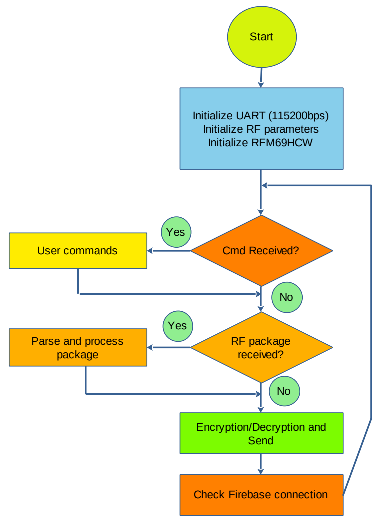

Start
Initialize UART (115200bps)
Initialize RF parameters
Initialize RFM69HCW
Cmd Received?
Yes
User commands
No
RF package received?
Yes
Parse and process package
No
Encryption/Decryption and Send
Check Firebase connection

Fig. 11. Flow chart of the proposed sensor node

When a node wishes to contact a device, it broadcasts its data packet and can optionally include an address byte so that all nodes except the desired receiving node ignore the packet. According to a document from HopeRF, the theoretical maximum payload for a variable length packet with CRC is 255 bytes, but to use the on-chip FIFO of the RFM69HCW allowing for simple communication between our micro-controller and the transceiver over SPI communication bus, we are limited to a 66-byte payload as shown on Layer 1 in Fig. 10. The optimized payload data in which with important parameters include received signal strength indication of a sender (RSSI), frame count, sensor data or commands, and interval time as illustrate in Layer 2 (see Fig. 10.). In addition, by reading the RSSI from the far side that the proposed RF gateway can manage the quality of connections instability and broadcast data packages success.

2.5 Websocket

The proposed RF gateway can connect to a server-side by using SocketIO (SocketIO client). SocketIO is a web technology in the HTML5 specification providing bi-directional communications channels over a single TCP/IP connection [7]. In this way, the Websocket protocol makes possible more interaction between a browser and a server. However, the legacy browsers lack the support of the emerging Websocket technology. Some investigation explored that Websocket compatible with integration server-side for applications in IoT. On the other hand, SocketIO, by using a plugin of Node.js, which aims to easily make real-time applications possible in every web browser and mobile device, blurring the differences between the different transport mechanisms, which is focused on care-free real-time and, supported in many programming languages. HTTP/HTTPS that it uses POST/GET methods to transfer data on the internet network and it is a common method for Web monitoring and long-polling. However, the key issue of HTTP is which a server cannot send requests to a client-side (eg: gateway, nodes) for the establishment of a connection between server and clients. Other issues, the security policies because of allowing emulation of a push data mechanism to the server with a limited interval time and the latency of data packages [15], [22]. Another popular communication on the network in IoT application is MQTT/MQTT-SN protocol that they have been used topics to publish and subscribe data over the network. MQTT allows operation over TCP [15] and was designed for devices that had more memory and processing power than many of the lightweight, power-constrained IoT devices have available to them. TCP requires more handshaking to set up communication links before any messages can be exchanged. This increases wake-up and communication times, which affects the long-term battery consumption. TCP connected devices tend to keep sockets open for each other with a persistent session. This adds to power and memory requirements. MQTT protocol is a type of centralized network architecture that a centralized broker that can limit scale in which the broker can affect scalability as there is additional overhead for each device connected to it. The network can only grow as large as the local broker hub can support it. According to IBM document is carried out that MQTT is data-agnostic so it's possible to send images, texts in any encoding, encrypted data, and virtually every type of data in binary format [15], [23]. To solve these issues, we use SocketIO that enables real-time, bi-directional, and event-based communication between RF gateway and server-side. In addition, we also utilize the resource of SocketIO to develop a Websocket server and APIs to access parameters on the RF gateway.

3 Portable Website Interface and Implementation

3.1 Portable Website Interface

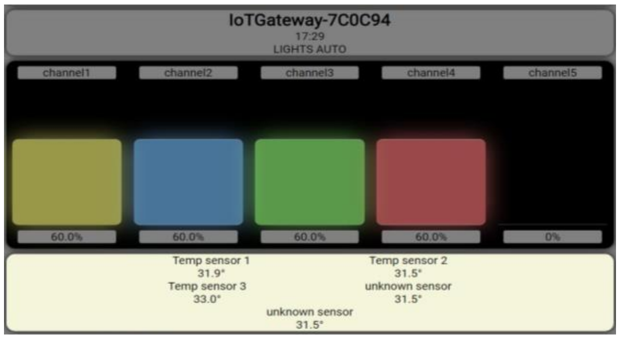
A friendly portable web interface was developed for management parameters of RF gateway as well as logging the measurement result from sensor nodes and supporting control of the actuator node on the wireless network. And we would like to thank to authors [24], who provided resources to development the web interface. The web interface is consists of a general tab, a setup tab, a channels tab, and a histogram of each single sensor node. To schedule actuators, we can use the channels tab that allows program a scheduler by using a click event handler of mousers, touch screen on mobile devices as shown in Fig .7Furthermore, devices may not provide any user interface or may not be accessible when it is developed process and it will become more difficult to maintenance and changeable configuration in future. On the others hand, the user needs to configure devices that might require the user deeper knowledge of the IoT devices. Because of the limitations of the non-volatile memory of the ESP32 microcontroller and this is a challenge for design a portable web application. Therefore, we used gzip compression method to deploying a web interface in which the server generally has to be configured over file types and those files should be gzipped. By using gzip method to compress files, the ESP32 is not only reduced the size of server files but also can reduce network response time to clients

3.2 Implementation RF Gateway with Online Mode

An IoT dashboard is still under development process that allows remote monitoring of measurement results from sensor nodes. This also allows remote controls to various high-power electrical devices such as lights, motors. For experiment deployment, the remote server provides internet services that it has been allowed the proposed RF gateway to connect to backend APIs, storage information of sensor nodes, actuator nodes and management of these nodes. For illustrating of operation modes, after turn on the power supply to the RF gateway, if the internet connection is available then the RF gateways can establish a connection to the remote server by using SocketIO protocols with a set of special parameters to handshake such as domain name, connection port, specify events of interest definition. On the opposite, if the internet is not valuable then the offline mode will be archived.

3.3 Portable Website Interface

The main purpose of this proposed web interface is to manage all operations of the designed RF gateway that it is not used to log the sensor data when the internet is not available, but also allows wireless control actuators through a router WiFi connection, the user can access the portable web interface on mobile devices or even laptops. The main page is shown in Fig. 12 with 5 channels and five temperature-measured sensors on the bottom of the home page. In Fig. 8, the more detailed RF gateway with support to the technician, developer by visualizing parameters in the "Status" container included system start time, firmware version, WiFi status and RSSI of the transceiver module. We also develop others function such as management of sensors nodes, control buttons and using user and password for safety control actuators as Sensors block, Lights block, and System block, respectively.



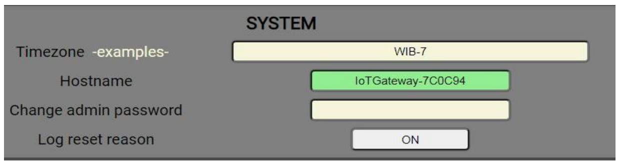
IoTGateway-7C0C94
17:29
LIGHTS AUTO

channel1	channel2	channel3	channel4	channel5
60.0%	60.0%	60.0%	60.0%	0%

Temp sensor 1
31.9°
Temp sensor 2
31.5°
Temp sensor 3
33.0°
unknown sensor
31.5°
unknown sensor
31.5°

Fig. 12. A screenshot of the proposed portable web interface: 1) Upper part shows the management of the actuator with a percentage of pulses; 2) Lower part illustrates the result of measurement temperature sensors.

To avoid multi-clicks generation and protect high-power actuator change status (turn on and turn off) to frequent times in a period, a simple user and password security based is added and shown in Fig. 13. This interface, also allows users to change reference time (time zone) that will be recorded in log files.



SYSTEM

Timezone -examples-	WIB-7
Hostname	IoTGateway-7C0C94
Change admin password	
Log reset reason	ON

Fig. 13. A screenshot of the configurations of the proposed RF gateway: Time zone, hostname, security password, and the button that allow log reset events.

Fig. 14 shows an example historical data of 2021-08-14 of ambient temperature that was monitored in our laboratory. The users can roll up and roll down historical data by using arrows. Additionally, this deployment also allows for a better understanding of the potential of measured data. Because of the non-volatile memory constraint of ESP32-WROOM with 4 Megabytes (MB) therefore we added an automatic delete function to the file manager tab and even the users can direct delete those historical data files.

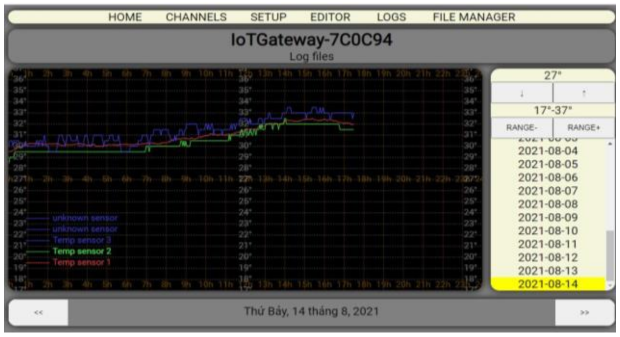
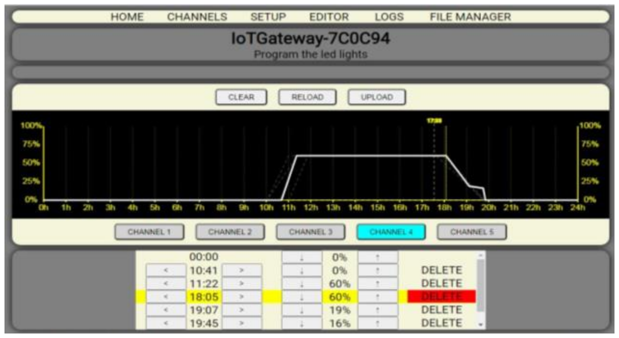


Fig. 14. A screenshot of the line chart of the results of measurement laboratory temperature sensors with the date of separation.



00:00	0%	
10:41	0%	DELETE
11:22	60%	DELETE
18:05	60%	DELETE
19:07	19%	DELETE
19:45	16%	DELETE

Fig. 15. An example user interface to program scheduler of an actuator by using mouse click event handler; the vertical axis shows the percentage of lights (using PWM); the horizontal shows the time with accuracy in a minute.

Lastly, Fig. 15 illustrates a friendly web interface by using a mouse click event handler. When a mouse clicked a point, a specific command with string format would be generated and stored in the no-volatile memory of ESP32. The proposed RF gateway can manage up to 50 commands in one channel. Limitations are still turning on, turning off, and generating pulse to GPIOs. However, three types of control signals are common in various IoT applications

4 Experiment Result and Discussion

In this section, we will briefly overview of the proposed system as well as discuss about functions in short.

4.1 Experiment with Wireless Sensor Nodes

We have been made setup experiments of our system to illustrate the advantages of the proposed RF gateway FreeRTOS-based with five DS18B20 temperature sensors and an actuator node and separate into two sensor nodes include RF1 and RF2. The temperature sensor DS18B20 with characteristics is illustrated in Table 2 that it can be used in monitoring applications with high accuracy of ± 0.5 Celsius degrees. On the RF1, we have soldered 4 temperature DS18B20 sensors and RF2 has only one temperature sensor. For the RF3, we have been elaborated as an actuator node that can receive the wireless command from the implemented RF gateway and we used a 2.4GHz WiFi router to provide WLAN connections to the designed gateway. The experiment system is shown in Fig. 1.

TABLE 2
The Characteristics of the DS18B20 Temperature Sensor

Description	DS18B20 sensor
Measuring range	-55^{0}C to 125^{0}C
Accuracy	-550C to 1250C
Operating range	± 0.5 0C over range (-100C to 850C)
Supply voltage	-550C -1250C
Response time	3.0-5.5VDC
Sensor type	≤ 750ms
Dimension/pins	1-Wire interface
Number of sensors	3 pins

4.2 Application of the Proposed RF Gateway

The proposed RF gateway can operate in two modes included an online and offline mode that we have been introduced in section 3. An IoT dashboard is also developing for remote management of RF gateways. The implemented RF gateway and wireless nodes are designed and developed under elaborating many studies and the main idea of this gateway is focused on using the FreeRTOS library. We have been success deployment and experiment with five sensor nodes as shown in Fig. 12 and Fig. 14. A simple web interface of the designed RF gateway has been deployed by using a mouse click event handler that is easy for the users and without complex software settings as illustrated in Fig. 15. It also supports touch event handlers on any sensitive screen of mobile devices.

5 Conclusion

The proposed RF gateway can be applied in various applications in IoT areas that can be used to monitor wirelessly environment sensors and control actuators with the limitation of communication range of RFM69HCW module approximately 400 meters long. These desired results have been covered the designed requirements. However, there are a few drawbacks of these results as such it is impossible to apply to transmit video because of the limitations of wireless module RFM69HCW. Another disadvantage of the designed RF gateway that is only connects to a 2.4GHz WiFi router, but it cannot connect to a 5GHz WiFi router. Therefore, it should be considered before selecting a router WiFi in practice. In addition, the design requires a lot of tasks, each of which requires its stack, and many of which require a queue on which events can be received. This solution, therefore, uses a lot of RAM of microcontroller, therefore, it will become difficult to deploy on a common microcontroller in practice. Besides, the context switching between tasks of the same priority will waste processor cycles.

Acknowledgment

This research was supported by Branch of Vietnam Research Institute of Electronics, Informatics, and Automation (Vielina-

HCMC) , HUTECH University and Nguyen Tat Thanh University, Ho Chi Minh City, Vietnam (address: Nguyen Tat Thanh University, Ho Chi Minh City, 70000, Vietnam).

REFERENCES

[1] Giang T. L., Nhat M. T., Thang V. T., IoT System for Monitoring a Large-Area Environment Sensors and Control Actuators Using Real-Time Firebase Database, 12th Intelligent Human Computer Interaction International Conference (IHCI2020), Daegu, South Korea, pp.3-20, Nov, 2020.

[2] “Document Google Firebase”, https://firebase.google.com

[3] “Document Amazone FreeRTOS”, https://aws.amazon.com/freertos/

[4] S. Verma, Y. Kawamoto, Z. M. Fadlullah, H. Nishiyama, and N. Kato, Survey on network methodologies for real-time analytics of massive IoT data and open research issues, IEEE Communications Surveys & Tutorials , vol. 19, no. 3, pp. 1457–1477, 2017.

[5] Çoban, S., Gökalp, M.O., Gökalp, Eren E., P.E., Koçyi ˘git, A., Predictive Maintenance in Healthcare Services with Big Data Technologies., In Proceedings of the 2018 IEEE 11th Conference on Service-Oriented Computing and Applications (SOCA), Paris, France., pp. 93–98, 2018

[6] Chen MJ, Duh JM, Shie RH, Weng JH, Hsu HT., Dynamic real-time monitoring of chloroform in an indoor swimming pool air using open-path Fourier transform infrared spectroscopy. Indoor Air., Vol.26(3): 457–67, 2016

[7] Internet Engineering Task Force, IETF RFC 6455: “The Web-Socket Protocol”, IETF, Orlando, Fla, USA, 2011

[8] Wan-Young C., Giang T. L., Thang V. T. and Nam H. N., Novel proximal fish freshness monitoring using batteryless smart sensor tag, Sensors and Actuators B: Chemical, Vol. 248, Sep, 2017

[9] W Tao, L Zhao, G Wang, R Liang, Review of the internet of things communication technologies in smart agriculture and challenges, Computers and Electronics in Agricultural, Elsevier, 2021

[10] Shahzad, M.K.; Cho, T.H. A Network Density-adaptive Improved CCEF Scheme for Enhanced Network Life time, Energy Efficiency, and Filtering in WSNs. Adhoc Sens. Wirel. Netw., Vol.35 , pp.129-149, 2017

[11] “Samsung Smart Things Hub”, https://www.smartthings.com/products/smartthings-hub

[12] “Open Source Brain-Computer Interface Database”, https://openbci.com/

[13] M. Di Natale, H. Zeng, P. Giusto, A. Ghosal, Understanding and Using the Controller Area Network Communication Protocol: Theory and Practice, Springer Science & Business Media, 2012

[14] “Awesome Benefits of Internet of Things (IoT) for Web and Mobile Appl Development”, https://theiotmagazine.com/

[15] Application-aware end-to-end delay and message loss estimationin Internet of Things (IoT) - MQTT-SN protocols

[16] Wolf L, Büsching F, Institute of Operating Systems and Computer Networks (eds) 17. GI/ITG KuVS Fachgespräch Sensornetze 13. & 14., Braunschweig: Technical Report. Braunschweig, 2018

[17] Rajib Mall, “Real-Time Systems: Theory and Practice”, pp. 73-74, Edition 2nd CSE.

[18] “FreeRTOS FAQ - Memory Usage, Boot Times & Context Switch Times”, https://www.freertos.org/FAQMem.html

[19] Naresh R. Shanbhag, Algorithms Transformation Techniques for Low-Power Wireless VLSI Systems Design, International Journal of Wireless Information Networks, Vol. 5, No. 2, 1998

[20] Reinder. J. Bril, Wim F. J. Verhaegh, John J. Lukkien, Extract worst-case response times of real-time task under fixed-priority scheduling with defered preemption, in proceedings 19th Euromicro Conference on Real-Time Systems (ECRTS'07), pp.4-6, Pisa, Italy, 2007

[21] C. L. Liu and James W. Layland. 1973. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment, JACM 20, Vol.1, pp.46–61, 1973.

[22] Alessandro Ludovici 1 and Anna Calveras, A Proxy Design to Leverage the Interconnection of CoAP Wireless Sensor Networks with Web Applications, Sensors, Vol.15, pp.1217-1244, 2015

[23] “IBM Documentation: MQTT restrictions and limitations”, https://www.ibm.com/docs/en/maximo-monitor/8.4.0?topic=messaging-restrictions-limitations

[24] Renzo Mischianti, “Resoure web application design”, https://www.mischianti.org/2020/10/26/web-server-with-esp8266-and-esp32-byte-array-gzipped-pages-and-spiffs-2/

[25] Riu Mao, “HLA Compliant Real-Time Operation System Simulation, Ottawa-Carleton Institute for Electric Engineering”, Factculy of Engineering, 2002

[26] Wang, L.; Yang, Y.; Zhao, W.; Xu, L.; Lan, S. Multi-rate Network Coding for Energy-Efficient Multicast in Heterogeneous Wireless Multi-hop Networks. Adhoc Sens. Wirel. Netw., Vol.32, pp.197-219, 2016

[27] “ESP32 A feature-rich MCU with integrated Wi-Fi and Bluetooth connectivity for a wide-range of applications”, https://www.espressif.com/en/products/socs/esp32/overview

[28] Maier, A., Sharp, A. and Vagapov. Y. (2017). Comparative analysis and practical implementation of the ESP32 microcontroller module for the Internet of Things. In: Proc. 7th IEEE Int. Conference on Internet Technologies and Applications ITA-17, Wrexham, UK, 2017

Biographies

Giang Truong Le

He received the B.E. degree in Electronics and Communication Engineering from Ho Chi Minh City University of Technology, Ho Chi Minh, Vietnam, in 2011 and the M.S. degree in Electronic Engineering from Department of Electronic Engineering, Pukyong National University, Busan, South Korea, in 2016. He is currently working as an executive researcher at Nguyen Tat Thanh University, Ho Chi Minh City, Vietnam. His current research interests include Internet of Things, RFID, wireless sensor network and vegetable freshness monitoring.

Thang Viet Tran

Thang Viet Tran received the B.E. degree in Electronic Engineering from Can Tho University, Can Tho, Viet Nam, in 1997, the M.S. degree in Automatic Engineering from Ho Chi Minh City University of Technology, Ho Chi Minh, Viet Nam, in 2003 and Ph.D. degree in Electronic Engineering from Pukyong National University, Busan, South Korea, in 2016. He is a senior reseacher in Vietnam Research Institute of Electronics, Informatics and Automation. His current research interests include ultra-low power sensor node design with energy harvesting, wireless sensor network, IoT based application and automation system.

Phuc Thinh Doan

He received the B.S. degrees from Mechatronics, Mechanical Engineering Department of Ho Chi Minh City University of Technology (HCMUT), Viet Nam in 2008. He then received the M.S. and Ph.D degrees from Mechatronics, Mechanical Engineering Department of Pukyong National University, Korea, in 2011 and 2013 respectively. He is currently a researcher in Nguyen Tat Thanh Hi-Tech Institute, Nguyen Tat Thanh University, Ho Chi Minh City, Vietnam. His research interests includes Robotics, Motion control, Mobile Robot, Automatic Guided Vehicle, IoT, Automation and Smart Factory.

Trong Hai Nguyen

He was born in Vietnam on February 1, 1975. He received the B.S. and M.S. degree in Dept. of Electronics and Telecommunication, Ho Chi Minh City University of Technology, Vietnam in 1999 and 2003. He received Ph.D. in Mechatronic Engineering from Pukyong National University since 2017. His research fields of interest are nonlinear control, robust control, path planning algorithm, artificial intelligence.