Sensors and Materials, Vol. 36, No. 11 (2024) 4849–4864
MYU Tokyo

S & M 3835

Node-RED Web-based Monitor and Control of Power System Using Modbus and Message Queuing Telemetry Transport Communication in Raspberry Pi Embedded Platform

Ming-Hung Lin,[1] Sheng-Han Wu,[2] Bo-Wun Huang,[2] Po-Hsun Chen,[1] Chao-Hung Huang,[3] Cheng-Yi Chen,[1*] and Cheng-Fu Yang[4**]

[1]Department of Electrical Engineering, Cheng Shiu University, Kaohsiung City 833301, Taiwan
[2]Institute of Mechatronic Engineering, Cheng Shiu University, Kaohsiung City 833301, Taiwan
[3]Department of Mechanical Engineering, National Kaohsiung University of Science and Technology, Kaohsiung City 807618, Taiwan
[4]Department of Chemical and Materials Engineering, National University of Kaohsiung, Kaohsiung 811, Taiwan

(Received April 30, 2024; accepted September 4, 2024)

Keywords: Raspberry Pi, MQTT communication protocol, Internet of Things, node-RED, programmable logic controller

The rapid development of the industrial Internet of Things (IIoT) has transformed manufacturing processes, particularly in industries such as electronics and aerospace. While larger enterprises have successfully implemented IIoT applications to enhance productivity and facilitate smart manufacturing, small and medium-sized enterprises have faced challenges in adopting these technologies owing to financial constraints and limitations in their existing equipment. Therefore, in this paper, we propose to use Raspberry Pi as the hardware basis and the Cheng Shiu University laboratory as the research target. The approach presented in this paper involves integrating open-source packages to design both the programmable logic controller (PLC) publishing pattern and the PLC subscribing pattern. The proposed approach uses a message queuing telemetry transport (MQTT) communication protocol for transmitting Modbus transmission control protocol (TCP) register data and bidirectional data exchange with a remote monitoring system. An MQTT Broker is established to act as a bridge between the monitoring platform and the control system. A Node-RED-based IoT platform is set up to subscribe to topics from the PLC publisher for data collection. Additionally, the human–machine interface (HMI) in the monitoring system utilizes MQTT communication to publish PLC control commands, and a database is implemented for the historical analysis of monitoring data. In this study, we aim to establish a web-based cross-platform monitoring and control system using low-cost integration through Raspberry Pi 4B embedded systems and various communication protocols. The design facilitates the IoT transformation of PLC control systems. This integrated system has broad applications in construction, production equipment, and power system monitoring to achieve power monitoring and energy conservation.

*Corresponding author: e-mail: k0464@gcloud.csu.edu.tw
**Corresponding author: e-mail: cfyang@nuk.edu.tw
https://doi.org/10.18494/SAM5103

ISSN 0914-4935
https://myukk.org/

1. Introduction

The release of the iPhone in 2007 and the development of the Android system in 2008 made mobile communications increasingly popular. In addition to the touch screen subverting the way mobile phones are operated and becoming a milestone in modern mobile phones, smartphones have also become an integral part of everyone's life, and with the widespread adoption of mobile communication, the interconnectivity among individuals has gradually extended to IoT.[1] IoT refers to a system comprising sensing devices, computing units, and controlled devices. Each device can communicate and transmit data through networks and is equipped with unique identifiers.[2] IoT's rapid application and development have found broad use in data collection, remote monitoring, and automated algorithms, thereby expediting the progress of big data, AI, Industry 4.0, and smart factories.[3,4]

IoT has penetrated lifestyles and various application levels. Smartphones have become a part of everyone's life, and the execution of mobile applications has become the terminal device of the human–machine interface (HMI) of IoT, allowing remote control of equipment. The IoT architecture can be divided into perception, network, and application layers.[5] The perception layer is dedicated to recognizing, sensing, and controlling various states of end objects. The network layer facilitates the transmission of sensed information to the database system in the application layer. At the application layer, diverse data analysis technologies are synthesized to cater to the distinct application needs of various enterprises. The network layer of IoT operates on a network architecture based on transmission control protocol (TCP)/IP transmission. Depending on the distance and application scenario, it is categorized into long-distance transmission, utilizing technology such as 5G communication, and short-distance transmission such as Wi-Fi and Bluetooth.[6] In 1999, IBM developed the message queuing telemetry transport (MQTT) protocol for data transmission in inspecting oil pipelines in deserts.[7] Since then, it has evolved into the most widely utilized communication protocol in the field of IoT. MQTT is specifically designed for machine-to-machine connections, facilitating the interconnection of various actuators and sensors through the internet. This architectural framework involves message transmission between message publishers and subscribers, relaying data through a message broker. Importantly, it ensures data transmission with minimal volume, optimizing efficiency in communication.[8–10]

The implementation of Industry 4.0 relies on digitizing production data and equipment automation, equipment communication links and data exchange, equipment monitoring, data collection, and managing resources to achieve optimized production capacity and rapid development response capabilities.[4,11,12] With the changes in manufacturing, the mass production model is not the only way to maintain business operations, create profits, and reduce costs. To gain a competitive advantage, adapting to small-scale, diversified production, enhancing product-added value, and adopting flexible manufacturing methods with high efficiency are essential. Therefore, applying the industrial IoT (IIoT) involves transmitting a substantial amount of sensor information in smart factories. This pursuit aims to optimize the manufacturing process, enable product traceability, and simultaneously reduce costs while enhancing production output.[13] The composition of the programmable logic controller (PLC)

control systems can be a collection of multistation equipment. Each station's PLC controller can connect to a centralized monitoring system through the Modbus communication protocol. The PLC programming for automated equipment includes conditions for equipment startup, conditional control with branching logic, sequential control procedures, anomaly detection, emergency shutdown, and monitoring of the operational status of each component.[14] Through communication links, equipment systems can be connected in parallel or in series, facilitating workflow adjustments and enabling production methods to be either substation production or flow production. However, compatibility issues may arise owing to different PLC brands and programming languages. While most PLC brands support Modbus communication, it becomes a key condition for integrating diverse systems.[15] In the application of power monitoring control systems, the InduSoft professional development software platform is applied to design a web-based remote monitoring and control system for building power and environmental conditions.[16] The system allows graphical control of laboratory devices, monitors power data, and displays results using curve graphs. As the costs of professional supervisory control and data acquisition development software continue to rise, it may not be cost-effective for general purposes. Reference 17 also highlights the potential benefits of integrating existing control systems with IoT by converting Modbus data into MQTT communication format, making sensor data easily interpretable. Effective data management and application can enhance visibility and usability in smart factories. With Node-RED as an IoT remote monitoring platform, node functions process raw data, monitor instrument outputs, and store data in databases. The established database can aid in optimizing production processes.

In this paper, we refer to the experimental system presented in Ref. 16, where the air conditioning system was replaced, and a modified control method was incorporated. A gateway utilizing the MQTT protocol was established for message exchange. Python programming was employed for real-time conversion to the Modbus communication format, enabling bidirectional data acquisition and control functions with the PLC. The acquired data can be stored in a database system for future data analysis and applications. The frontend web design and website development adopted IBM's Node-RED visual programming tool to create a web-based graphical remote monitoring system. In this study, we elevate existing control systems and devices to a web-based IoT platform for remote monitoring and control by integrating Modbus and MQTT communication protocols. Administrators can monitor the system's status and functionality anytime through this integrated Web-based monitoring system.

2. Design of a Cross-platform Monitoring and Control System

In this study, the monitoring system uses the Node-RED visual IoT development tool as the leading software for system planning. The overall system software architecture is shown in Fig. 1. A Python program is developed as an execution sequence, utilizing the paho.mqtt.client and pyModbusTCP.client functions to implement the Modbus TCP communication protocol for reading PLC register data and converting it into the MQTT communication format for publishing. Similarly, it converts the MQTT communication format data back into the Modbus TCP communication protocol for writing into PLC registers, completing the subscribing

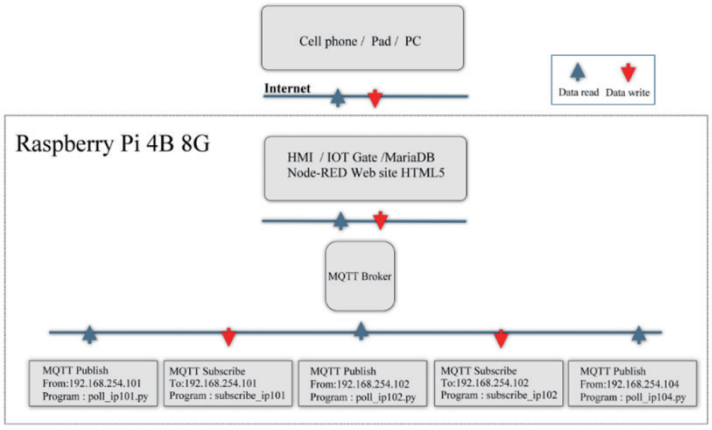

Fig. 1. (Color online) Software architecture of web-based monitoring and control system.

endpoint program. Furthermore, subscription and publishing nodes relayed through the MQTT broker to Node-RED flows undergo data processing, flow control, and HMI construction, enabling bidirectional data retrieval and remote control within the IoT architecture. Through internet connectivity, computers, smartphones, and tablets can log in to the HMI graphical interface, clearly representing the system status and immediate remote device control. On the basis of the laboratory hardware architecture in Ref. 16, we can exhibit overall power monitoring, subdistribution power monitoring, power data inquiry, and air conditioning control. The design screen of each tab is adjusted in accordance with the required functions. The clear visual interface increases its convenience. It is easy to control the system's various functions and understand the device's status. The general design content and function description are discussed next.

The design and execution process of the power monitoring system, as illustrated in Fig. 2, includes the following steps: (1) reading Modbus registers and publishing power monitoring data in MQTT format; (2) designing Node-RED flow processes, including subscribing to topics for data reception, generating power demand data, creating graphical monitoring interfaces, and outputting system alerts; (3) designing Node-RED flow processes for the real-time monitoring screen of today's demand units, calculating the 15 min average power demand, obtaining the daily highest demand value and occurrence time from the 96 demand units, using the daily highest demand as the basis for the statistical recording of yearly, monthly, and daily power demand values and power usage records in a MariaDB database; (4) designing a query form interface to query the MariaDB database and generate historical charts of power demand and power usage.

2.1 Modbus power data acquisition and MQTT message publishing

The laboratory power data is measured from the multiloop digital power meter SMB350. The I-8837 PLC#3 programmable controller obtains the measured power information from the SMB350 through Modbus RS485 RTU communication and stores the data in the Modbus register. The main control unit can use the Modbus TCP communication protocol to read power

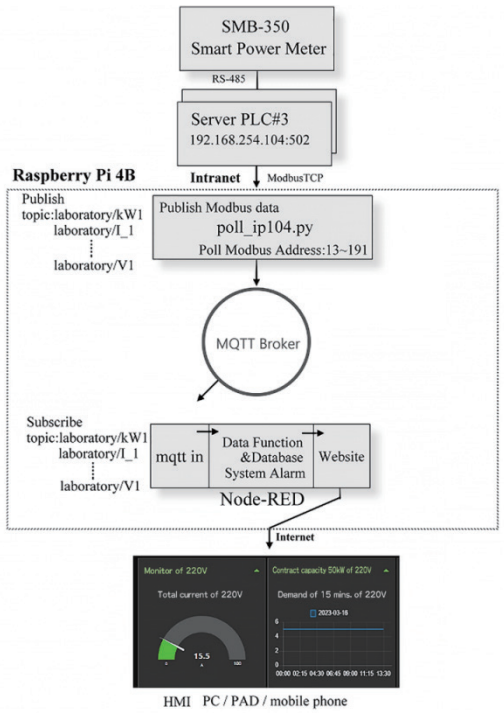

Fig. 2. (Color online) Design and execution workflow of power monitoring system.

data from the programmable controller I-8837 PLC#3 using IP: 192.168.254.104 with port 502. Since the meter measurement value can be changed with time, the system is configured to update data every second. Considering the allocation of hardware resources, the PLC publisher program in the main control unit is configured to poll the registers on the PLC server every 3 s to obtain the latest power data. The acquired power data is then published in the MQTT communication format. The MQTT Broker forwards this data to the subscribing node in Node-RED, which can graphically display the data in accordance with the planned layout. The power measurement data is distributed across Modbus registers at addresses 7 to 191.

The main control monitoring system integrates the functions of pyModbusTCP.client and paho.mqtt.client to establish a connection between the PLC publisher program, PLC, and the MQTT Broker, as illustrated in Fig. 3. The read_input_registers function is utilized to retrieve a maximum of 125 register data each time. Therefore, it is necessary to obtain array-type data in two separate instances. The variable reg_list acquires the array values starting from address 0, capturing 125 registers. The variable reg_list2 acquires the array values starting from address 125, capturing 66 registers. Figure 4 illustrates an example of monitoring 220 V loop current. The three-phase total current I_1 value is obtained from the array reds_1 (corresponding to Modbus address 73). The client.publish() function is employed to publish MQTT messages with the topic “laboratory/I_1” and the payload of the I_1 value in JSON format. Similarly, the values of the R, S, and T phase currents are also published as MQTT messages using the same code and approach.

```
#Establish connection with 192.168.254.104 PLC Server
c_ip104 = ModbusClient(SERVER_HOST104, SERVER_PORT104)
if not c_ip104.open():
      print("unable to connect to ",SERVER_HOST104,":",SERVER_PORT104)

#Establish a connection with MQTT Broker
client = mqtt.Client()
client.connect("127.0.0.1", 1883, 60)
client.username_pw_set("xxxx","xxxx")
```

Fig. 3. (Color online) Part of the PLC publisher program used to establish a connection between pyModbusTCP.client and MQTT Broker.

```
#Three-phase 220V main loop current monitoring  modbus address 73
#I_1 = regs9[72]
client.publish("laboratory/I_1", json.dumps(regs9[72]))

#220V R phase current monitoring modbus address 13
#I_11 = regs9[12]
client.publish("laboratory/I_11", json.dumps(regs9[12]))

#220V R phase current monitoring modbus address 14
#I_12 = regs9[13]
client.publish("laboratory/I_12", json.dumps(regs9[13]))

#220V R phase current monitoring modbus address 15
#I_13 = regs9[14]
client.publish("laboratory/I_13", json.dumps(regs9[14]))
```

Fig. 4. (Color online) Part of the PLC subscriber program used to obtain current value and publish it in MQTT format.

2.2 Node-RED HMI and MQTT messages

Figure 5 illustrates the Node-RED MQTT subscriber node settings, where one can subscribe to topics published by the PLC endpoint. Within the flow, by double-clicking the mqtt in node, one can access the attribute editor and input the MQTT Broker address: 127.0.0.1 and port: 1883, with the topic set to "laboratory/I_1", quality of service configured as 2, and output set to "a parsed JSON object". The settings for the remaining three-phase current configurations are completed using the same method. This node configuration can be directly applied to the entire Node-RED project, facilitating the swift addition of subscription topic flows. The node and flow configurations are illustrated in Fig. 6. The MQTT in the node receives the topic "laboratory/I_1", with msg.payload values being integers. Within the node of "set flow I_1", the reading values are multiplied by a factor of 0.001 to obtain actual measurement values. Subsequently, the topic is specified as "220 V Main Loop electric circuit" through the change node. By connecting the "mqtt in", "change", and "gauge" nodes with dragging lines, upon deployment, the msg.payload values can be directly transmitted to the gauge node, displaying the current status graphically under the title "220 V_Main Loop electric circuit". The function of the "220 V alarm output function" node is shown in Fig. 7. When the current value exceeds 50 A, the msg.payload value represents the content of the overload alarm voice output. Since the power data is updated every 3 s, the voice alarm needs to undergo a delay node, with a 10 s interval for output. This prevents messages from accumulating and causing alerts to fail to stop when the condition is resolved. The "audio out" node serves as the voice output module. Upon deployment of the aforementioned connections, the electric current monitoring HMI interface can be easily completed.

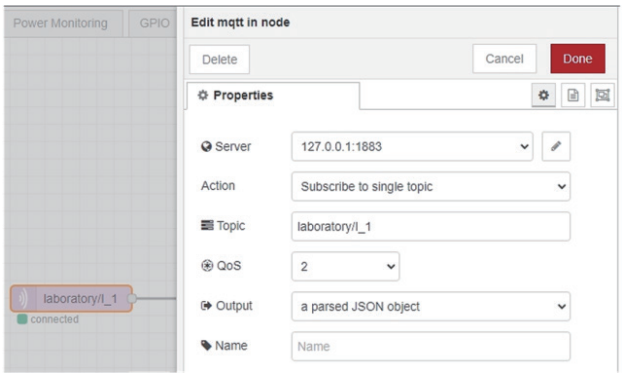

Fig. 5. (Color online) Subscriber settings of "mqtt in" node in Node-RED.

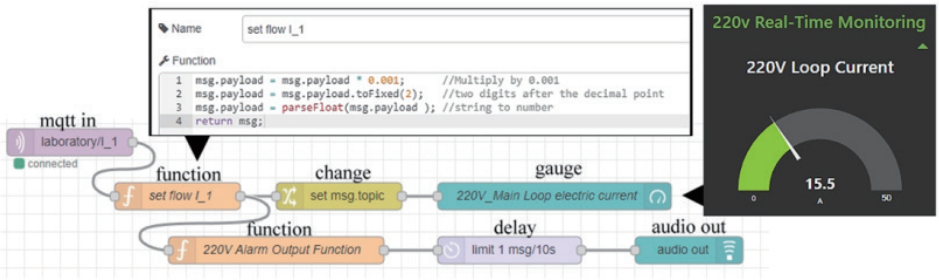

Fig. 6. (Color online) Process and nodes of 220 V loop current monitoring.

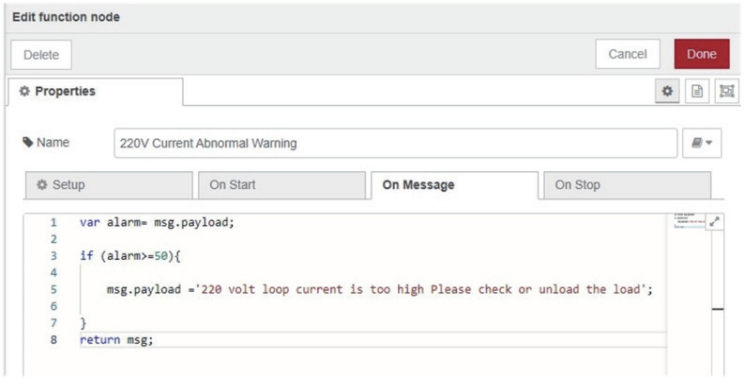

Fig. 7. (Color online) Function node settings and alarm voice output.

2.3 Monitoring power average demand and consumption

After signing a contract with power consumers, Taiwan Power Company (TPC) provides the contract capacity (kW) to consumers. The calculation of basic power tariffs is also based on the contract capacity. The charge is twice the basic power rate for excess power consumption of less than 10% of the contracted capacity; excess power consumption exceeding 10% of the contracted capacity is charged at thrice the basic power rate. As the cost of exceeding the basic contract

capacity is prohibitively high, the essential purpose of demand monitoring lies in the reasonable allocation of power usage and scheduling power consumption time to reduce power cost. TPC's demand unit operates on a 15 min cycle, accumulating effective power within the cycle and calculating its average value. There are 96 demand units per day, totaling approximately 2880 demand units per month, with the highest demand representing the maximum value. To promptly understand the usage status of demand, the design of monitoring screens should include a real-time display of demand values, 15 min average demand values, and alarm outputs so that the electricity of some equipment can be unloaded before exceeding the system′s set demand value. The process and nodes of real-time demand display are shown in the line chart and Node-RED flowchart, as shown in Fig. 8.

The laboratory database for recording power consumption data is illustrated in Fig. 9 and comprises 12 tables. These tables are distinguished on the basis of power loops and categorized into 220 and 110 V. Tables "220v_kwdata" and "110v_kwdata" are utilized to record 96 demand unit values per day, along with timestamps. Tables "220v_daykwdata" and "110v_daykwdata" log the maximum demand unit values per day and the corresponding dates. Similarly, tables "220v_monkwdata" and "110v_monkwdata" are designated for storing the monthly maximum

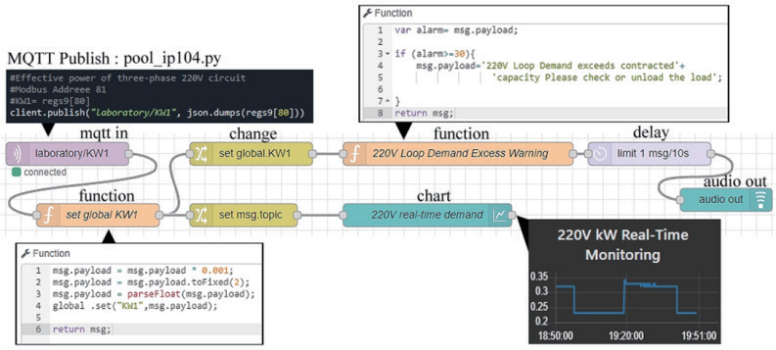

Fig. 8. (Color online) Process and nodes for displaying real-time demand using a line chart.

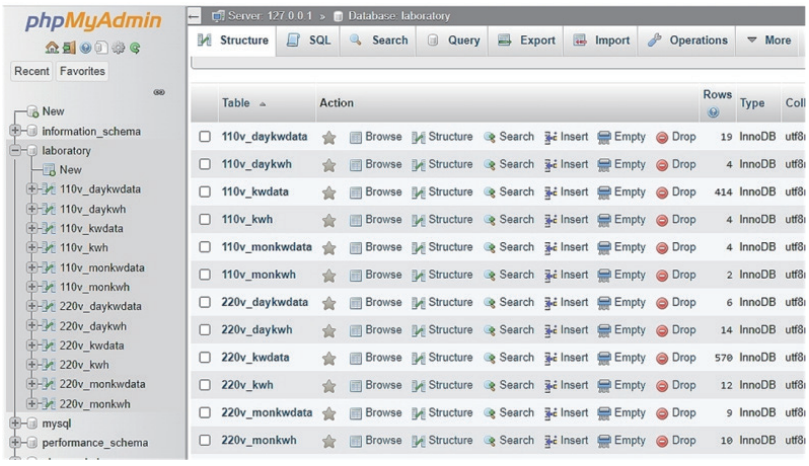

Fig. 9. (Color online) Database and table settings for proposed system.

demand unit values. Tables "220v_kwh" and "110v_kwh" record the total daily meter values. Tables "220v_daykwh" and "110v_daykwh" log the daily power consumption values and dates. Lastly, tables "220v_monkwh" and "110v_monkwh" are intended to store monthly power consumption values. Node-RED establishes a connection between MySQL nodes and the MariaDB database server. The connection settings specify the MariaDB server location, connection port:3306, username, password, and the database name to be connected. Following the configuration update, the connection setup is completed successfully.

The process and nodes for calculating demand units are illustrated in Fig. 10. Taking the 220 V power loop as an example, the global variable KW1 represents the real-time demand value updated every second. The global variable "sun_kw" accumulates the KW1 variable 60 times per min and 900 times every 15 min. However, one second of the program is designed to reset the variable at the 1st second. Therefore, the sum of values accumulated over 15 min is divided by 899 to obtain the average demand unit value. The calculation cycle occurs at the 15th, 30th, 45th, and 0th minute of each hour. The content of the function node comprises MariaDB insertion record commands. Data for each demand unit data is stored in the table laboratory/220v_kwdata according to date and time. The "series" field in the table records the title, the "data" field inserts the demand value, the "system_time" field inserts the record time, and the "labels" field inserts the period when the demand unit occurred, such as 15:00. After inserting the record, the variable "sun_kw" is reset to continue accumulating and calculating the next demand unit, recording the demand variations at different times.

From the laboratory database, querying the 220v_kwdata table for real-time demand involves setting up a line chart that updates every 30 s. The average 15 min demand values for the day are outputted to the chart node by sorting the data based on the current date. Figure 11 illustrates the screen displaying the line chart of today's demand records for power real-time monitoring. Similarly, other power data, such as power factor, should be arranged and set up in the layout using the same method and process.

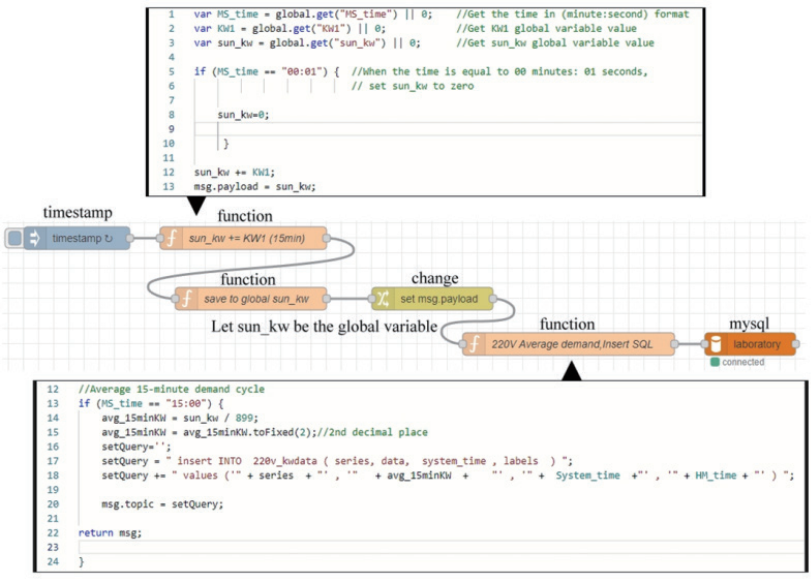

Fig. 10. (Color online) Process of unit demand calculation and database insertion record table code snippet.

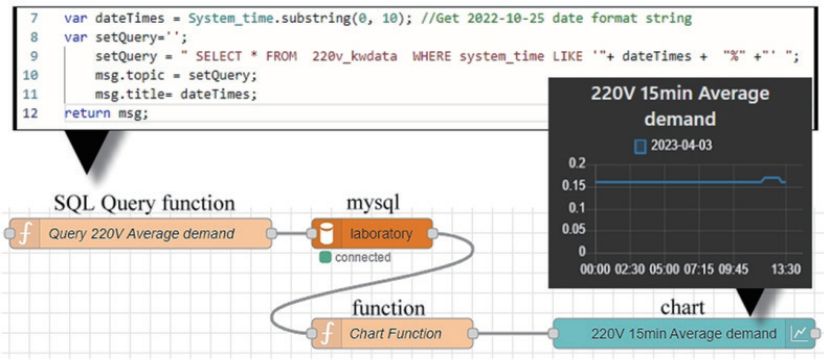

Fig. 11. (Color online) The query data table is displayed on the monitoring screen as a line chart.

Owing to the PLC register's 16-bit integer range of −32768 to 32767 and the fact that multiloop meters accumulate values in 32 bits, two 16-bit registers are used for calculation. The calculation formula is High_bit × 32767 + Low_bit. Figure 12 illustrates an example of a 220 V power loop, depicting the nodes and processes involved in displaying and utilizing the power consumption on the HMI screen.

The msg.payload values of KWH1_2Lbit and KWH1_2Hbit are obtained separately by the "mqtt in" nodes and set as flow variables. These values in the 220V_KWH function node are then multiplied by a factor of 0.01 to convert them into actual power consumption values. The power consumption value is then set as a global variable KWH1_2. Apart from the total power consumption display, Fig. 13 depicts storing the KWH1_2 variable value into the 220v_kwh table at 23:56:00 daily. It retrieves the previous day's power consumption value from the table, subtracts it to obtain the current day's power consumption, and then stores it in the 220v_daykwh table. On the 1st day of every month at 01:00, the system reads the table 220v_daykwh, calculates the total power consumption for the month, and stores it in the 220v_monkwh table to complete the power consumption trend database.

2.4 Power consumption historical trend query

Power consumption trend data is generated from daily power consumption data processing. The accumulated data can generate annual, monthly, and daily power consumption trend charts, which can be used to analyze applications in production capacity allocation and appropriate power usage, for example, how to schedule power usage during peak, off-peak, and shoulder periods, as well as plan the layout for peak and off-peak power prices during summer and non-summer months. Taking the demand unit records of the 220 V power loop as an example, the method implemented in this study for generating data involves reading the table 220v_kwdata at 00:01 daily to obtain the previous day's maximum demand value, which is then stored in the table 220v_daykwdata to record the maximum demand value for each day. Similarly, at 00:01 on the 1st of each month, the table 220v_daykwdata is read to obtain the record of the highest demand for the current month, which is then stored in the 220v_monkwdata table to record the occurrence of the highest demand for the month.

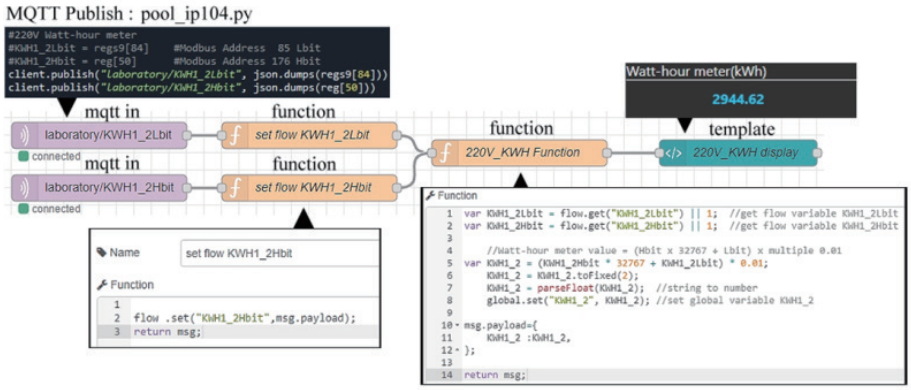

Fig. 12. (Color online) Kilowatt-hour numerical conversion process and partial program of the function.

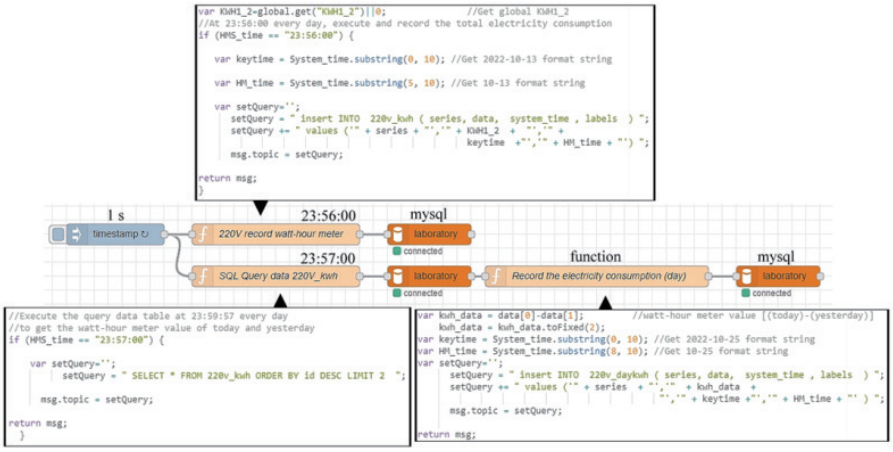

Fig. 13. (Color online) Process and function of total power consumption record.

The power monitoring data query page can utilize a form node to generate input forms, with the date format set as the sorting value for the database tables. Figure 14 illustrates the query of the table system_time field of 220v_kwdata for the sorting value of 2023-03-25, resulting in the retrieval of 96 demand unit data for that day. The output can be exported as an Excel file using the Excel node, while the chart node can generate line charts or bar charts to represent the power consumption trend. For monthly demand trend queries, the system_time field of the table 220v_daykwdata is queried with the sorting value of 2023-03. Similarly, for annual demand trend queries, the system_time field of the 220v_monkwdata table is queried with the sorting value of the year 2022. The "Chart Function" node obtains the required data by database query command and transposes it into the data array format. Using the multiple entries of the same procedure enables the chart node to display multiple data sets as bar charts. For example, the relationship between various fields queried from the 220v_kwdata table based on the day parameter and displayed by the chart node is shown in Fig. 15.

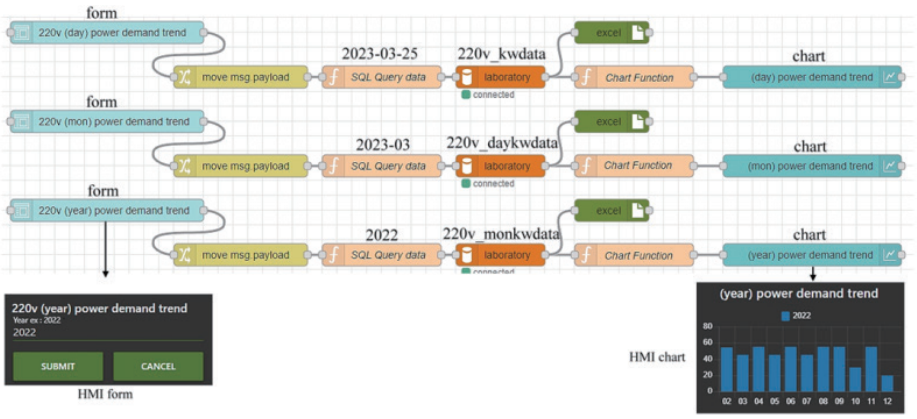

Fig. 14. (Color online) Node process for querying year/month/day demand trends.

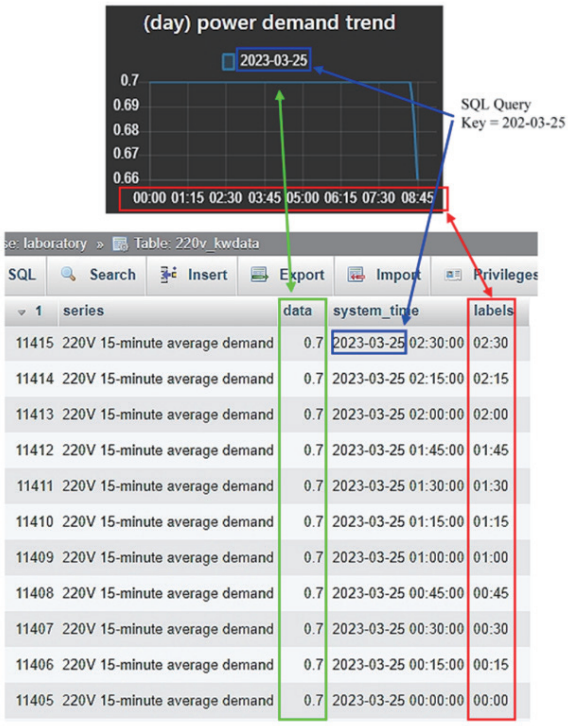

Fig. 15. (Color online) Correlation between date trend query data table and chart node display.

2.5 Air conditioning control system

The laboratory has two variable-frequency constant-temperature air conditioners with a total cooling capacity of 14.0 kW. The air conditioners can be set to memory mode with predefined temperature and fan speed settings, and they start operating in accordance with these settings once the power is turned on. The main unit can control the air conditioners' operation by driving external relays to open or close the electromagnetic contactors using the GPIO outputs of a

Raspberry Pi 4B. Figure 16 illustrates the nodes and processes for integrating the air conditioners into the HMI interface control. The topics "laboratory/air_conditioner ON" and "OFF" are used to control the air conditioner's operation, with msg.payload set to 1 for ON and 0 for OFF. The button node is similar to the HMI Button node, while the LED node displays the air conditioner's status (ON or OFF). The msg.payload input values for the rpi-gpio out node control the specified GPIO channel to output either low voltage (OFF) or high voltage (ON), corresponding to 0 and 1, respectively.

3. Experimental Results

The integration of experimental software and hardware to convert Modbus register data into MQTT communication for publishing is achieved through the "poll_ip101.py", "poll_ip102.py", and "poll_ip104.py" programs. Conversely, the program for subscribing to MQTT and writing to Modbus registers is designed by the "subscribe_101.py" and "subscribe_102.py" programs. Node-RED is utilized for all flow designs, HMI screen setups, data visualization, debugging, and integration with installed environments such as Mosquitto MQTT Broker, Node-RED, and MariaDB on the hardware Raspberry Pi 4B for stability testing. The practical test results of accessing HMI screens and displaying data by logging into the Node-RED dashboard are discussed next.

3.1 Power monitoring system

Upon logging into the Node-RED dashboard at IP: 120.118.143.181:1880/ui and entering the username and password, users are immediately directed to the real-time power monitoring HMI screen shown in Fig. 17. Figure 17 shows the power monitoring trend and data value of the circuit loops of 220 and 110 V. This includes real-time and 15 min average power demand values, as well as the R, S, and T phase voltages and currents, apparent power, and reactive power. Other

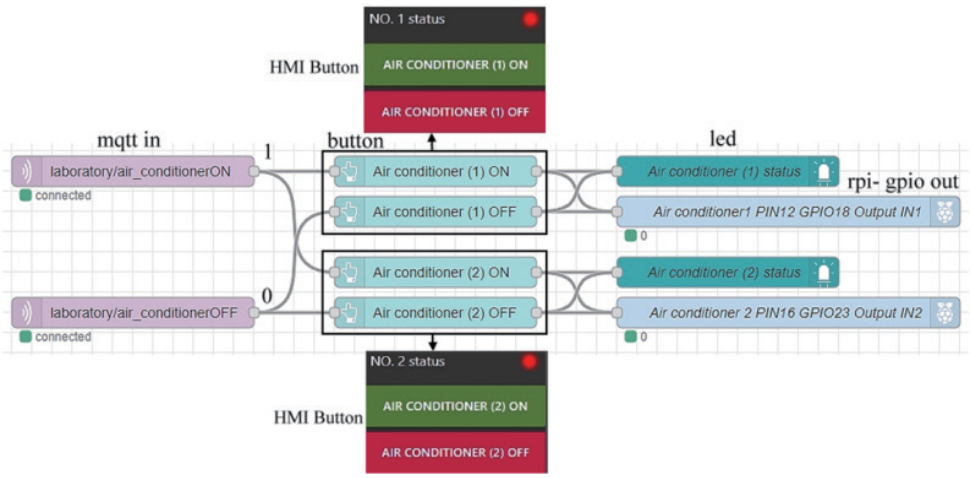

Fig. 16. (Color online) Process and nodes of air-conditioning HMI control.

Fig. 17. (Color online) Real-time power monitoring screen for 220 and 110 V main circuit.

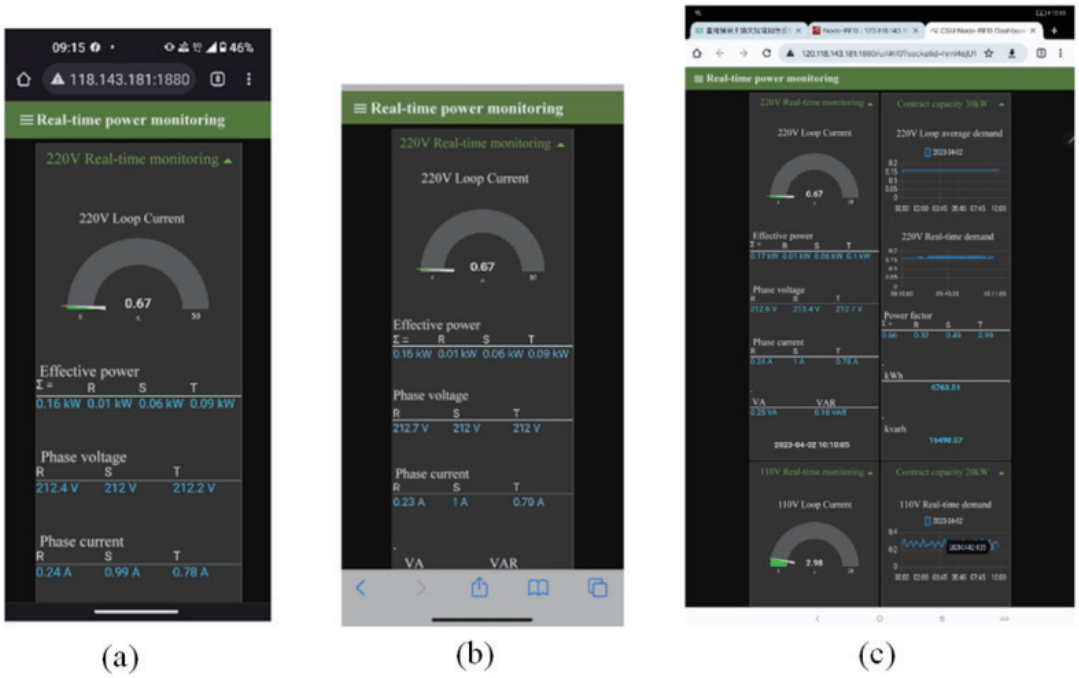

Fig. 18. (Color online) Cross-platform testing: (a) 5.1-inch Android smartphone running Chrome, (b) 5.5-inch iPhone running Safari, and (c) 10.6-inch Android tablet running Chrome.

data on electrical consumption include power factor and energy consumption. In instances where the real-time demand exceeds the contracted capacity or when the main circuit's current usage is too high, a voice alarm is triggered, alerting users to take action to reduce power load and inspect equipment accordingly.

3.2 Cross-platform testing of monitoring system

The remote monitoring interface designed by Node-RED has cross-platform functions and can automatically adjust the display style to fit different screen sizes. Figure 18 shows the results of a cross-platform test of real-time power monitoring using different operating systems and screen resolutions. Figures 18(a) and 18(b) show the monitoring screens executed using Android and iPhone mobile phones, respectively. In contrast, Fig. 18(c) shows the monitoring screen results executed on a 10.6-inch tablet using the Chrome browser.

4. Conclusions

In this study, we utilized Node-RED as a web-based IoT integration platform, employing Python programming to integrate MQTT and Modbus communication protocols to achieve cross-platform system monitoring and control functionalities for various laboratory devices, accessible via mobile devices. All functionalities were executed through a user-friendly graphical interface. The following results were achieved in this research implementation. Embedded integration was achieved at a low cost using open-source software packages and Raspberry Pi 4B hardware, including Node-RED, Mosquitto MQTT Broker, MariaDB, and MQTT subscriber and publisher programs for reading and writing Modbus registers. PLC control systems can be upgraded to a networked control system, utilizing flow control and an HMI designed with Node-RED. Commands were issued to PLC control systems in a scenario-control manner, enabling functional control actions to be executed simultaneously. It was demonstrated that devices supporting Modbus in PLCs can easily realize PLC IoT control systems, providing existing devices with the possibility of sustainable use without the need for complete replacement. PLC system monitoring data was transformed into meaningful and easily recognizable data types in MQTT communication format. From the perspective of industrial production management, these processed data will become part of the big data sources. The power monitoring system displays real-time graphical interface information for each circuit, includes power demand and current monitoring alarm systems, stores data such as power demand and power consumption in the database, and generates yearly, monthly, and daily data in line charts and bar charts for trend analysis and report generation through database query forms. We established a complete web-based monitoring architecture and implementation method, which can be expanded for future smart building security management and monitoring applications.

Acknowledgments

This research was partially supported by projects under grant No. CS-111-03.

References

1 R. Kamal: Internet of Things: Architecture and Design Principles, McGraw Hill Education (Chennai, India, 2022) 2nd ed., Chap. 1.
2 A. K. Gupta and R. Johari: Proc. 2019 IEEE 4th Int. Conf. Internet of Things: Smart Innov. Usages (IEEE, 2019) 1–5. https://doi.org/10.1109/IoT-SIU.2019.8777342
3 P. Kiartsilapin and W. Sawangsri: Proc. 2019 3rd Int. Conf. Robot. Automat. Sci. (IEEE, 2019) 233–237. https://doi.org/10.1109/ICRAS.2019.8808939
4 S. K. Jagatheesaperumal, M. Rahouti, K. Ahmad, A. Al-Fuqaha, and M. Guizani: IEEE Internet Things J. **9** (2022) 12861. https://doi.org/10.1109/JIOT.2021.3139827
5 P. Sethi and S. R. Sarangi: J. Electr. Comput. Eng. **2017** (2017) 1. https://doi.org/10.1155/2017/9324035
6 N. Ahmed, D. De, F. A. Barbhuiya, and M. I. Hussain: IEEE Internet Things J. **9** (2022) 916. https://doi.org/10.1109/JIOT.2021.3104388
7 Introduction to MQ Telemetry: https://www.ibm.com/docs/en/ibm-mq/9.0?topic=telemetry-introduction-mq (accessed July 2023).
8 K. Namee, R. Kaewsaeng-On, J. Polpinij, G. M. Albadrani, K. Rueagraklikhit, and A. Meny: Proc. 2022 17th Int. Joint Symp. Artif. Intell. Natural Lang. Process. (IEEE, 2022) 1–6. https://doi.org/10.1109/iSAI-NLP56921.2022.9960287
9 R. A. Light: J. Open Source Softw. **2** (2017) 265. https://doi.org/10.21105/joss.00265
10 C. R. M. Silva and F. A. C. M. Silva: Proc. 2019 SBMO/IEEE MTT-S Int. Microw. Optoelectron. Conf. (IEEE, 2019) 1–3. https://doi.org/10.1109/IMOC43827.2019.9317637
11 D. Cemernek, H. Gursch, and R. Kern: Proc. 2017 IEEE 15th Int. Conf. Ind. Inform. (IEEE, 2017) 239–244. https://doi.org/10.1109/INDIN.2017.8104778
12 M.-Q. Tran, M. Elsisi, K. Mahmoud, M.-K. Liu, M. Lehtonen, and M. M. F. Darwish: IEEE Access **9** (2021) 115429. https://doi.org/10.1109/ACCESS.2021.3105297
13 P. Kiartsilapin and W. Sawangsri: Proc. 2019 3rd Int. Conf. Robot. Automat. Sci. (IEEE, 2019) 233–237. https://doi.org/10.1109/ICRAS.2019.8808939
14 G. Bhandari, M. Prakash Hiremath, A. Joglekar, A. Kulkarni, D. Kulkami, C. Mahadeva, S. B. Mohanty, D. Raghunath, M. B. Raju, and R. Shorey: Proc. 2020 Int. Conf. Commun. Syst. Netw. (IEEE, 2020) 688–690. https://doi.org/10.1109/COMSNETS48256.2020.9027475
15 S. Tamboli, M. Rawale, R. Thoraiet, and S. Agashe: Proc. 2015 Int. Conf. Smart Technol. Manage. Comput., Commun., Controls, Energy Mater. (IEEE, 2015) 258–263. https://doi.org/10.1109/ICSTM.2015.7225424
16 C. Y. Chen, C. Y. Liu, C. C. Kuo, and C. F. Yang: Micromachines **8** (2017) 241. https://doi.org/10.3390/mi8080241
17 K. Ferencz and J. Domokos: Proc. 2020 IEEE Int. Conf. Automat., Quality and Testing, Robot. (IEEE, 2020) 1–6. https://doi.org/10.1109/AQTR49680.2020.9129934