FreeRTOS Application in a Real-Time Air Quality Monitoring Architecture for Smart Campus

N. Silva, E. Alves, W. Júnior, A. Moares, N. Silva, and A. Balieiro

***Abstract*—Monitoring air quality in smart campuses is essential for safeguarding the health of the academic community, as air pollution in university environments can lead to respiratory and cardiac issues. While the Internet of Things (IoT) offers a suitable solution for monitoring and notifying about air quality in smart campuses, it is of paramount importance that these notifications occur in real-time to ensure information reaches users with minimal latency. This paper presents an architecture for collecting, storing, and notifying about atmospheric pollutants in a smart campus and analyzes the use of Real-Time Operating Systems (RTOS) for managing the system. The evaluation focuses on IoT technologies, represented by the end node, gateway, and cloud server. The experimental results show that using RTOS reduces the average transmission interval by 37.5% and achieves higher transmission and reception throughput compared to non-real-time operating systems. For instance, in a monitored location, the transmission and reception throughput reached approximately** 178.23 **and** 164.18 **with RTOS, compared to** 110.97 **and** 99.41 **without RTOS, respectively.**

Link to graphical and video abstracts, and to code: https://latamt.ieeer9.org/index.php/transactions/article/view/9391

***Index Terms*—IoT, Smart Campus, Air Quality, RTOS.**

I. Introduction

AIR pollution is a major issue for society. Every day, a large amount of atmospheric pollutants is released into the air, severely affecting human health. The concentration of atmospheric pollutants has been steadily increasing due to the expansion of human activities [1]. In Brazil, issue is further exacerbated by the high incidence of forest fires [2].

Technologies based on the Internet of Things (IoT) have being employed to design tools for air quality monitoring and notify users about pollution levels, and assist in taking action against pollution sources [3]. IoT consists of sensors and computing systems that make objects smarter and provide services to society [4], such as air quality monitoring.

A Smart campus is an IoT application that integrates these technologies into university campus life, eading to the application of intelligent features across various academic services [5]. Smart Campuses enable universities to combine advanced technologies with infrastructure to provide better service, support decision-making, and promote sustainability. [6].

In Smart Campus scenarios, air quality monitoring is essential since air pollution within a university environment can lead to health issues for the academic community, including respiratory and cardiac problems [7]. These effects can be particularly severe for elderly individuals or those predisposed to respiratory diseases [8]. In addition, many university laboratories conduct experiments with chemicals pose health risks. Continuous exposure to toxic gases, such as Carbon Monoxide, (CO), can cause adverse health effects [9]. Therefore, IoT technologies can be used to monitor and provide alerts about poor air quality within a university campus.

When toxic gases are involved, promptly notifying university campus users is crucial, as they cannot discern whether the air quality is safe, is crucial. In this context, [10] presents an air quality monitoring architecture that informs users about the Air Quality Index (AQI). This architecture consists of low-cost IoT devices, the LoRaWAN (Long Range Wide Area Network) protocol, gateway infrastructure, the FIWARE middleware platform, and an EndNode with ESP32-LoRa. However, the EndNode used to capture and transfer data in real time had limitations regarding air quality monitoring. It adopts the same core of the ESP32-LoRa for managing the data capture and transmission. This sequential processing causes the ESP32 to pause the data transmission while capturing air quality data, resulting in delays in user notifications and failing to meet strict deadlines.

In this context, this paper presents the use of FreeRTOS in the air quality monitoring architecture developed in [10]. FreeRTOS is a free real-time operating system (RTOS) compatible with the ESP32 microcontroller, supporting LoRa communication and integrated with the development environment. RTOSs are operating systems designed to ensure the completion of specific tasks within defined time constraints [11]. Threrefore, an RTOS relies not only on the logical outcomes of computations but also on their timely execution [12]. This capability enables sensor data to be captured and transmitted within predefined intervals, improving response times in IoT applications.

We applied FreeRTOS on the ESP32-LoRa and analyze its impact on EndNode transmission time and data reception by the cloud service, comparing it to the non-RTOS version. We also evaluated data transmission and reception rates. The main contribution of this work is the development of a hard real-time system for air quality monitoring in smart campus scenarios. Unlike air quality monitoring systems that rely

The associate editor coordinating the review of this manuscript and approving it for publication was Guillermo Valencia-Palomo (*Corresponding author: Elton Alves*).

N. Silva, Elton Alves, W. Júnior, A. Moares, and N. Silva are with the Federal University of Southern and Southeastern Pará, Marabá, Pará, Brazil (e-mails: erroso@unifesspa.edu.br, eltonalves@unifesspa.edu.br, wmvj@unifesspa.edu.br, alife.silva@unifesspa.edu.br, and nandsoncunha@unifesspa.edu.br).

A. Balieiro is with the Centro de Informática, Universidade Federal de Pernambuco, Brazil (e-mail: amb4@cin.ufpe.br).

TABLE I
COMPARATIVE TABLE OF RELATED WORKS

Solution	Purpose	IoT Comunication	RTOS	Performance
[7]	Air quality monitoring	Not applicable	✗	✗
[13]	Air quality monitoring	ESP8266 Wifi	✓	✗
[14]	Smart chicken poultry system	Arduino Mega-Wifi	✓	✓
[15]	Real-time fire alarm project	Not applicable	✓	✓
[16]	Health data monitoring system	Gateway Bluetooth	✗	✗
[17]	Noise and air pollution monitoring	Arduino-Wifi	✗	✗
[19]	Indoor air quality monitoring	Gateway Zigbee	✗	✓
[18]	Real-time monitoring of IoT devices	Gateway LoRaWAN	✗	✓
Proposal	**Indoor and outdoor air quality monitoring**	**ESP32-LoRaWAN**	✓	✓

on sequential processing in end nodes, RTOS programming within a LoRaWAN communication architecture enables efficient taks management for capturing and transmitting sensor data, resulting in improved performance.

This work is organized as follows. Related works are discussed in Section II. Section III presents the adopted methodology. Section IV analyzes the obtained results. Finally, Section V concludes this paper and points out future directions.

II. Related Work

Several studies have involved IoT for air quality monitoring and RTOS. For instance, [13] presents an IoT-based air quality system that measures CO2 levels, air humidity, and the heat. The authors employ FreeRTOS on an Arduino Nano to manage related to sensor data transmission and visualization. Data are trasmitted to a web server using the secure HTTP POST protocol via the ESP8266 Wi-Fi module.

An IoT and RTOS-based monitoring system for poultry farms is presented in [14]. The proposed system uses three sensors: one to measure ammonia levels, another for carbon dioxide, and a third for humidity and temperature. FreeRTOS is implemented on the Arduino Mega, and sensor data are transmitted to the internet via a Wi-Fi interface. In contrast, [15] proposes a real-time alarm system for detecting forest fires using FreeRTOS on an Arduino Nano. The authors evaluate the proposed RTOS in terms of latency, jitter, and worst-case response time. However, their study does not incorporate IoT concepts, such as communication between devices and the cloud. On the other hand.

Some studies perform real-time monitoring using IoT without employing any RTOS for task management. For instance, [16] proposes a real-time and multi-scenario health data monitoring system. The authors use the Message Queuing Telemetry Transport (MQTT) protocol for communication with a gateway via Bluetooth. In [7], a solution for real-time air quality monitoring in urban environments is presented. It does not adopt any communication protocol between the network's hardware, relying solely on a wireless connection and the Intel Edison Board platform.

In [17], in turn, proposes a system that utilizes sensors to detect noise levels and concentrations of harmful gases, such as sulfur dioxide and carbon dioxide. The collected data is transmitted to an Arduino microcontroller and subsequently sent to a cloud environment, where it can be accessed remotely. Alerts are issued to authorities when pollution levels exceed predefined limits. The system is designed to be low-cost, efficient, and user-friendly, enabling continuous monitoring and prompt actions to mitigate pollution.

Since IoT applications generally exhibit different communication patterns, the authors in [18] evaluate LoRa communication as a real-time monitoring solution for IoT devices at the Federal University of Campinas in Brazil. Their results demonstrate that, within a smart campus environment, LoRa communication provides reliable real-time communication with IoT devices.

As observed, several studies that adopt RTOS and IoT, but with different communication technologies [13], [14], [15]. Others perform real-time monitoring but without employing any task management mechanisms [16], [7], [17], [18]. Additionally, some focus on smart campuses with LoRaWAN communication, but without evaluating the programming of IoT devices for real-time task execution [19]. In this respect, this work fills these gaps by presenting an air quality monitoring architecture for smart campuses designed to operate in real time with FreeRTOS, supporting both indoors and outdoors environments. To the best of our knowledge, no previous works in the literature combines and evaluates IoT devices with long-range communication (ESP32 and LoRaWAN) and FreeRTOS applied in air quality monitoring architecture tailored for smart campus scenarios. Table I summarizes the main differences of this proposal from related works.

III. Methodology

This work employed an exploratory research, as it evaluated an RTOS in an already developed architecture. Additionally, a quantitative research was conducted to collect sensor data and perform a comparative performance analysis of the RTOS implementation. Descriptive statistics were used to guide the process of data collection, organization, analysis, and interpretation.

Thus, this section describes the architecture of the proposed air quality monitoring system, highlighting the comparison between IoT devices with and without RTOS. Additionally, the hardware and software components, the use of RTOS, and the evaluation scenario of the proposal are presented.

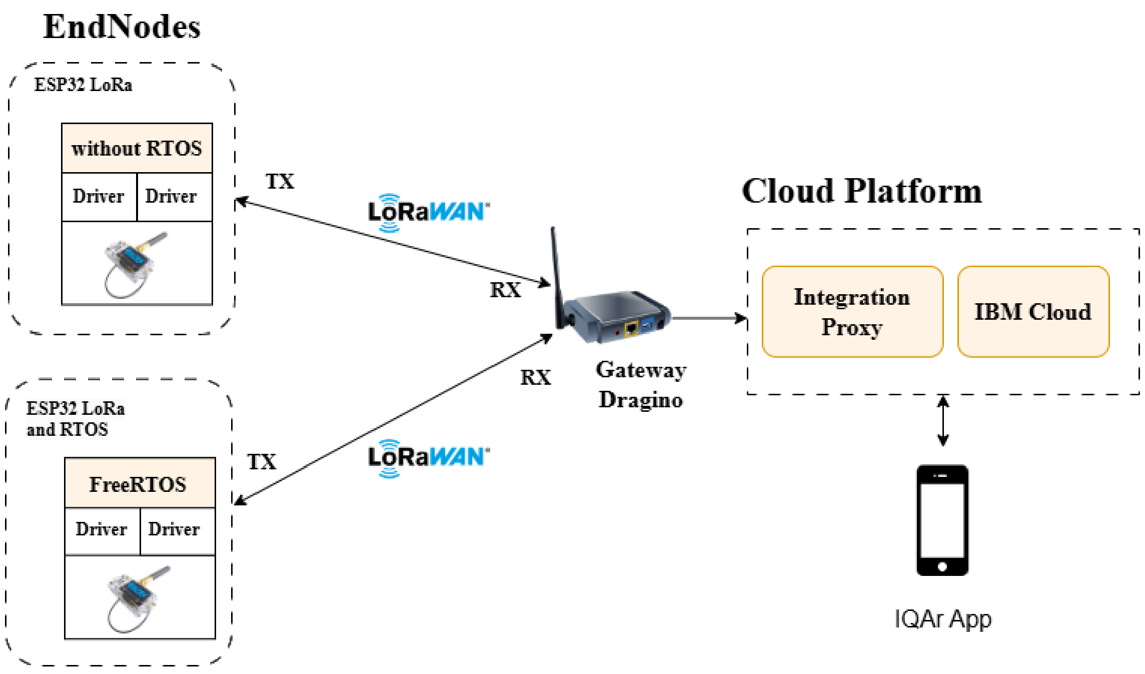


Fig. 1. Proposed air quality monitoring architecture with EndNodes, Gateway and Cloud Platform.

A. *Proposed Architecture*

Fig. 1 presents the air quality monitoring architecture, comprising three parts: (1) EndNodes (ESP32 Lora with and without RTOS), (2) Dragino gateway, and (3) Cloud platform.

The first part encompasses the ESP32 microcontroller with LoRaWAN communication support. LoRa is a technology designed for low power consumption, low data transfer rates, and long-range communication, making it suitable for IoT applications [18]. Our proposal evaluates the performance of the ESP32 LoRa programmed with RTOS compared to the ESP32 LoRa without RTOS, using the FreeRTOS library for LoRaWAN communication on the ESP32. MQ131 and MQ135 sensors were attached to the EndNodes, with the collected data transmitted (TX) to the Dragino gateway, which then forwards it to the final destination (RX), the cloud platform (IBM Cloud), for user notification. In this context, it is crucial to evaluate the transmission and reception throughput of this communication, considering a real-time air quality monitoring system.

The second part comprises a Dragino DLOS8N gateway, which receives the data transmitted by the EndNode. This connection with the EndNode is facilitated by the ChirpStack on the Dragino gateway, enabling the support for building a LoRaWAN network server.

The third part consists of the cloud platform infrastructure, comprising the proxy integration module and the IBM Cloud database. The integration proxy enables that data coming from the gateway to to be processed and stored in the IBM Cloud service. The IBM Cloud hosts information about the endNode locations, as well as the data sent by the sensors related to pollutants and performance metrics. Finally, the user, through a mobile application, can view the air quality index on the university campus.

B. *Circuit Simulation*

Fig. 2 shows the electronic connection scheme of the EndNode with the MQ131 and MQ135 sensors, micro SD memory card module, and the LoRaWAN connection to the Dragino DLOS8N gateway. The EndNode corresponds to the ESP32 LoRa 868/915 MHz with an OLED display and a 6 dBi antenna, featuring LoRa communication, an integrated OLED display, a Dual-Core microcontroller with a frequency of up to 240 MHz, and 32 MB of flash memory

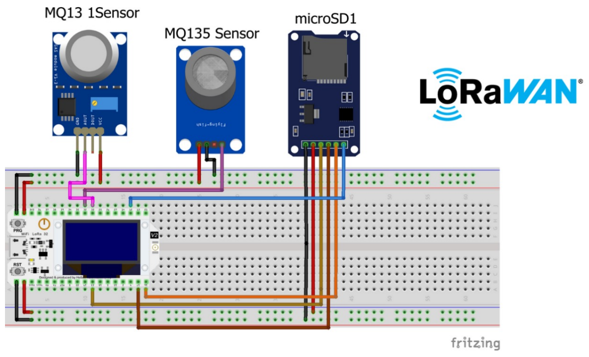


Fig. 2. EndNote electronic circuit with MQ131 and MQ135 sensors, SD card and connection to the gateway.

The MQ131 sensor measures ozone (O3) gas concentration, while the MQ135 sensor can detect concentrations of ammonia (NH3), carbon dioxide (CO2), benzene (C6H6), nitric oxide (NO), smoke, and alcohol. These sensors were chosen for being low-cost and compatible with the ESP32-LoRa. The micro SD memory card is used to store sensor data and the timestamps of TX transmissions to the cloud service, and the date and time of the gas concentration measurements. A 5-second interval was set for the EndNode to collect data from the MQ131 and MQ135 sensors. In this context, the literature does not define strict time deadlines for such tasks. Therefore, the chosen interval is considered a low delay for a real-time IoT scenario involving the EndNode, Gateway, and cloud service connection. Additionally, it represents a time interval that the ESP32-LoRa antenna power can reliably sustain. In the IBM Cloud service, the RX data, including date and time, are stored.

The synchronization and time update of the EndNode were carried out using the Network Time Protocol (NTP). NTP connects the EndNode to an external server, synchronizing its clock with the local time configuration. This setup allows both the ESP32 LoRa with RTOS and the ESP32 LoRa without RTOS to maintain synchronized connections with the Dragino DLOS8N, enabling performance evaluations at comparable time instances.

The communication between the EndNode and the Dragino DLOS8N gateway via LoRaWAN consists of two main stages: (1) the first stage involves pairing the EndNode with the gateway through pairing preparation, request preparation, the pairing request, and the response to the request; (2) the second stage occurs after successful pairing and consists of detecting the sensors and enabling full data transmission from the EndNode to the gateway.

C. RTOS Implementation

The development of the RTOS on the ESP32 LoRa was carried out using the Arduino LMIC Library ("MCCI LoRaWAN LMIC Library"). This library is designed to operate in the Arduino environment, allowing the use of SX1272, SX1276 transceivers, and various RFM9x compatible transceiver modules for data transmission. Fig. 3 synthesizes the tasks created for the scheduler to manage the core, with the tasks representing those executed in this work. The scheduler is responsible for defining when and for how long the programs will have access to the core.

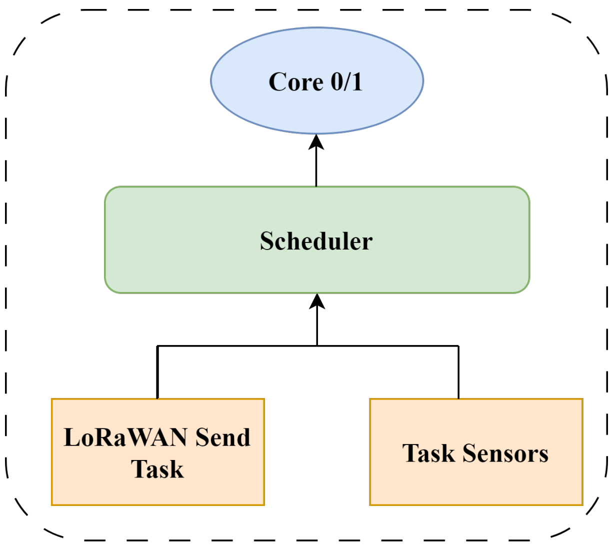


Fig. 3. LoRaWAN Send Task and Task Sensors task scheduling process with cores 0 and 1 of ESP32 LoRa with RTOS.

The `Task Sensors` is responsible for detecting data from the MQ131 and MQ135 sensors. Once detected, the data is stored in a queue for recording in the EndNode's memory. Concurrently, the `LoRaWAN Send Task` performs several functions: initializing the SPI (Serial Peripheral Interface) communication pins, writing the gateway keys to the device memory, and attempting to initiate the communication session. It then verifies the chip frequency, activates the sub-bands, configures the spreading factor, and adjusts the antenna gain. Once the communication configuration is complete, it sends the transmission and schedules the next one. Additionally, the `Send Task` retrieves the data from the queue, formats it for storage, and transmits it.

FreeRTOS was employed using a preemptive algorithm based on task priorities to determine the execution order. Table II presents the task priorities adopted in this study, `task_sensors` and `task_send_LoRaWAN`. The `task_sensors` task is assigned a higher priority ("2") because it detects sensor data and stores it in a queue for transmission via LoRaWAN communication. in contrast, the `task_send_LoRaWAN` task has a lower priority ("1"), as it retrieves data from the queue only after the data has been successfully stored.

TABLE II
TASK PRIORITIES

Priority	Task	Remarks
2	task_sensors	Data collection from sensors
1	task_send_LoRaWAN	LoRaWAN Communication

The tasks were created using the `xTaskCreate` function, which also initializes the task scheduler. Task priority scheduling is handled automatically by the scheduler, determining which core will execute each task based on their assigned priorities. The two tasks are configured as follows: the first task, with a priority of 1 (the lowest priority), is allocated a stack size of 4,096 words, representing the maximum stack capacity. The second task, with a priority of 2 (the highest priority), is also assigned a stack size of 4,096 words.

```
xTaskCreate(task_send_LoRaWan,
            "LoraWan_task",
            STACK_SIZE_LORAWAN,
            NULL,
            PRIO_LORAWAN,
            NULL);

xTaskCreate(task_sensors,
          "Sensores_task",
          STACK_SIZE_SENSORES,
          NULL,
          PRIO_SENSORES,
          NULL);
```

D. Evaluation Scenario

The scenario adopted in this study is shown in Fig. 4, which corresponds to Campus II of the Federal University of Southern and Southeastern of Pará which has an area of

52,031.90 m². Given that LoRaWAN technology provides wide communication coverage, the EndNodes were positioned at three different locations relative to the gateway. In block 1, the EndNodes were placed 145 meters from the gateway. In block 6 and the warehouse, they were positioned 135 meters and 30 meters away from the gateway, respectively. These locations correspond to the experiments for data collection.

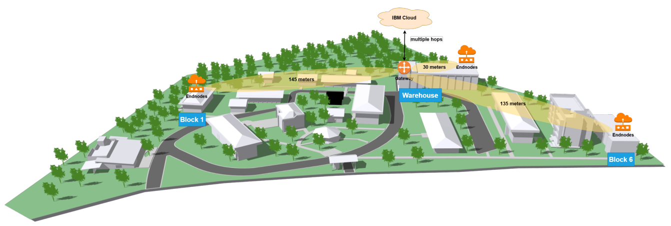


Fig. 4. Evaluation scenario of the proposal with the locations of the EndNodes and IBM Cloud.

Efforts were made to ensure that the devices were equidistant from the gateway and synchronized using the NTP protocol. This approach aimed to facilitate a fair comparative evaluation under identical conditions regarding distance and data traffic. In the test scenario, there was no vegetation obstructing the line of sight between the gateway and block 6, or between the gateway and the warehouse. Conversely, the line of sight between the gateway and block 1 was partially obstructed by an area of vegetation, which could potentially interfere with the LoRa communication signal.

IV. Results and Discussions

This section presents the evaluation results, focusing on the data transmission (TX) by the EndNodes and data reception (RX) on the cloud service. The analysis includes TX and RX throughput, TX and RX time intervals, and the TX and RX time gains. All tests were conducted with a 5-second interval for data capture by the EndNodes. To perform a comparative evaluation between the ESP32 LoRa and ESP32 LoRa with RTOS approaches, three experiments were conducted based on the scenario depicted in Fig. 4. The experiments are described as follow:

1) Block 1:Starts on 12/24/2024 at 4:11:02 PM and ends on 12/26/2024 at 10:46:33 AM.
2) Warehouse: Starts on 12/19/2024 at 10:14:04 AM and ends on 12/21/2024 at 05:23:48 AM
3) Block 6: Starts on 12/23/2024 at 2:47:02 PM and ends on 12/24/2024 at 3:24:14 PM.

Table III displays the number of samples obtained (with and without RTOS) for TX and RX rates in the evaluation scenario. Notably, the use of RTOS resulted in a higher TX rate and, consequently, a higher RX rate. This reinforces that, with RTOS, the EndNode is better equipped to meet data transmission deadlines, thereby reducing delays in the TX rate. Table IV presents the success and loss percentages for TX and RX rates. The adoption of RTOS led to a higher success rate and a lower loss rate in the TX and RX processes, demonstrating that the scheduling of the `task_sensors` and `task_send_LoRaWan` tasks enhances the efficiency of communication process between the EndNode and the gateway.

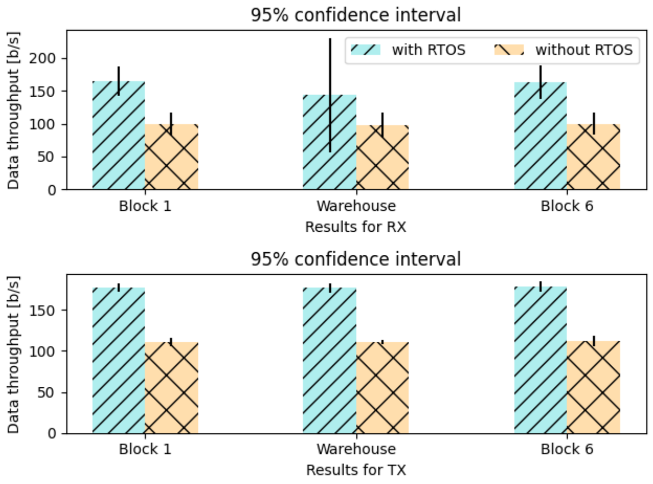


Fig. 5. RX (upper part) and TX (lower part) throughput obtained in the evaluation scenarios.

Fig. 5 illustrates the RX throughput (upper part) and TX throughput (lower part) of the EndNodes, with a 95% confidence interval represented by vertical error bars. These throughputs represent the number of bits transmitted or received over the total duration of the experiments. The results demonstrate that employing FreeRTOS enables faster TX data transmission across all three environments. Similarly, RX throughput is also higher when using FreeRTOS. These findings findings confirm that the integration of RTOS facilitates a greater volume of data transmission and reception within a given time interval.

Fig. 6 presents the results for the time intervals between RX receptions (upper part) and TX transmissions (lower part) across the three environments, with a 95% confidence level. These intervals represent the time elapsed between two successive RX receptions or TX transmissions. The results demonstrate that, in all three locations, the use of RTOS resulted in shorter TX intervals. In Block 1, Warehouse, and Block 6, the TX intervals were approximately 5 seconds, aligning with the time required for the EndNode to capture sensor data. Conversely, without RTOS, longer TX intervals were observed in all environments, with values close to 8 s. Similarly, for RX, RTOS led to shorter intervals between receptions. In contrast, the absence of RTOS led to significantly longer RX intervals across all experiments.

Fig. 7 illustrates the time gain achieved for RX (top) and TX intervals (bottom) when RTOS is applied, based on measurements taken in the warehouse. The results indicate that the use of RTOS significantly reduces the time intervals for both TX and RX. Specifically, an average time gain of 3.78 s was observed for RX, while the average gain for TX was 2.98 s.

Tables V and VI provide a comparative summary of the arithmetic mean and error margins for RX and TX throughputs, as well as for RX and TX time intervals across all evaluated locations, both with and without the use of RTOS.

TABLE III
SUMMARY OF THE NUMBER OF SAMPLES IN EXPERIMENTS

	Number of Samples			
	RX		TX	
Location	**With RTOS**	**Without RTOS**	**With RTOS**	**Without RTOS**
Block 1	27,037	15,882	30,477	19,069
Warehouse	27,480	16,230	30,839	19,299
Block 6	14,583	8,700	16,935	10,612
Total	69,100	40,812	78,251	48,980

TABLE IV
MEASURING SUCCESS AND FAILURE IN EXPERIMENTS

	Loss Rate (%)		Success Rate (%)	
Location	**With RTOS**	**Without RTOS**	**With RTOS**	**Without RTOS**
Block 1	11.287	16.712	88.712	83.287
Warehouse	10.892	15.902	89.107	84.097
Block 6	13.888	18.017	86.111	81.982

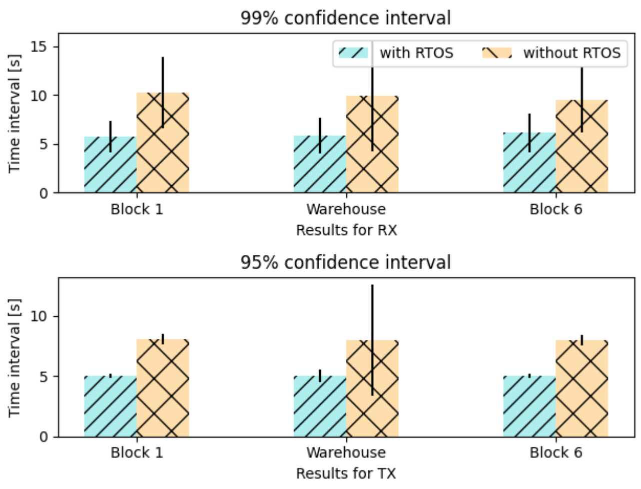


Fig. 6. Time interval for RX (upper part) and TX (lower part) obtained in the evaluation scenarios.

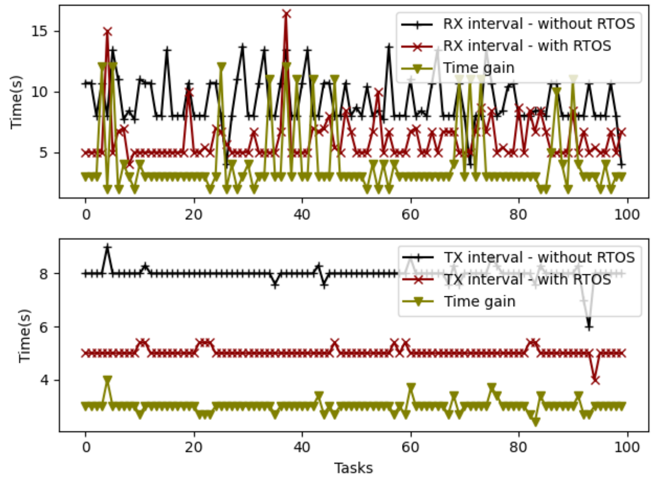


Fig. 7. Results for RX (upper part) and TX (lower part) time gain obtained between without RTOS and with RTOS.

These results reinforce the superior performance of the proposed approach when utilizing RTOS.

For RX and TX throughput, higher averages were achieved with the use of RTOS. The highest RX average was observed at the Warehouse, likely due to its proximity to the gateway. Meanwhile, the RX throughput averages at Block 1 and Block 6 were 164.18 ± 2.55 and 143.33 ± 3.01, respectively, reflecting their comparable distances from the gateway. The TX throughput averages using RTOS remained consistent across all three locations, demonstrating uniformity in data transmission performance.

Regarding the time interval between TX and RX, the use of RTOS confirmed that the average TX time values were close to 5 seconds in all three locations. Without RTOS, the worst time was recorded in the Warehouse, where the average transmission time was 8.05 ± 0.005 seconds. The RX time was also showed improvement with RTOS, where the averages were consistent across locations. In contrast, without RTOS, the worst RX time was recorded in Block 6, with an average of 9.73 ± 0.068.

The results demonstrated that employing RTOS on the EndNode-LoRa significantly improved performance in the air quality monitoring evaluation scenario on the university campus compared to the EndNode-LoRa without RTOS. Without FreeRTOS, the ESP32 LoRa processes data capture and transmission tasks solely on core 0, leading to task interruptions as one task is paused to accommodate another. In contrast, with FreeRTOS, the capture and transmission tasks operate without interruptions, as they are allocated to separate cores of the ESP32 LoRa based on their assigned priorities.

V. CONCLUSION

This work presented the use and performance evaluation of RTOS in an air quality monitoring architecture across three different locations on a university campus. The ESP32 LoRa was sequentially programmed to capture and transmit sensor data, while the ESP32 LoRa with RTOS was programmed with the same functions as the ESP32 LoRa, but using FreeRTOS

TABLE V
SUMMARY OF RESULTS CORRESPONDING TO DATA THROUGHPUT

	Metric corresponding to data throughput			
	Mean for RX		**Mean for TX**	
Location	**With RTOS**	**Without RTOS**	**With RTOS**	**Without RTOS**
Block 1	164.18 ± 2.55	99.41 ± 1.41	178.23 ± 2.53	110.97 ± 1.57
Warehouse	164.91 ± 0.63	99.61 ± 0.40	176.85 ± 0.11	110.48 ± 0.06
Block 6	143.33 ± 3.01	98.13 ± 1.86	176.76 ± 3.36	110.00 ± 2.11

TABLE VI
SUMMARY OF RESULTS CORRESPONDING TO TIME INTERVAL

	Metric corresponding to time interval			
	Mean for RX		**Mean for TX**	
Location	**With RTOS**	**Without RTOS**	**With RTOS**	**Without RTOS**
Block 1	6.13 ± 0.53	9.63 ± 0.04	5.03 ± 0.002	8.00 ± 0.003
Warehouse	5.73 ± 0.16	9.65 ± 0.11	5.03 ± 0.003	8.05 ± 0.005
Block 6	5.84 ± 0.006	9.73 ± 0.068	5.03 ± 0.003	8.00 ± 0.004

The results showed that employing RTOS in an IoT application with LoRaWAN communication presented advantages compared to using a traditional EndNode. Creating two tasks in the RTOS, one for capturing and another for sending, allows the ESP32 LoRa to send a task while performing a new capture stored in a queue. On the other hand, without RTOS, when initiating a data sending instruction, the ESP32 LoRa interrupts all other functions while sending. Thus, after completion, a new capture and send instruction is initiated, resulting in longer TX and RX times. In this regard, with the use of FreeRTOS, a higher data throughput of TX and RX was obtained, the highest RX and TX throughput values occurred in the Warehouse, with 164.91 ± 0.63 and 176.85 ± 0.11, respectively. On the other hand, without RTOS, the RX and TX values were 99.61 ± 0.40 and 110.48 ± 0.006, respectively, resulting in a data loss of 15.902%.

In terms of TX and RX time improvements, better compliance with the transmission time criterion was observed. The average TX rate in all experiments was approximately 5.03 s using RTOS. Without RTOS, the average TX rates were close to 8s, highlighting an improvement of 2.98s with RTOS. enhanced compliance with time deadlines is crucial for air quality monitoring within a smart campus environment, particularly when timely notifications to users are essential. Additionally, it was observed that the presence of vegetation did not affect the performance of the proposed system.

In general, the use of RTOS in the proposed air quality monitoring architecture successfully met the 5-second timeframe adopted in this study. This is important because enables timely alerts during critical moments when air pollution levels may pose a risk. Furthermore, with the use of RTOS, the highest success rates were achieved in the TX and RX processes, with the highest rate recorded in the Warehouse, reaching 89.107%.

This work presented the following scientific implications regarding the use of RTOS in an air quality monitoring architecture utilizing LoRaWAN communication: improved TX and RX data throughput, better assurance of meeting transmission times between TX and, consequently, RX reception, and more favorable success and data loss rates due to task management. These findings highlight the importance of critical real-time applications for air quality monitoring in smart campuses, especially when aiming for a shorter detection and reaction time from the sensors, as well as greater accuracy in the data received.

Future directions include analyzing other communication protocols between the EndNode and the gateway, such as the ZigBee protocol, evaluation of scenarios with varying distances and vegetation, which may impact the performance of the RTOS in the proposed air quality monitoring. Moreover, investigating the time until the air quality index notification is another interesting point. A comparison of FreeRTOS with other available RTOS options an evaluation of hardware resource usage with an RTOS are also important areas for future exploration.

REFERENCES

[1] H. Mokrani et al., “Air quality monitoring using iot: A survey” in IEEE International Conference on Smart Internet of Things (SmartIoT). IEEE, 2019, pp. 127-134. doi: 10.1109/SmartIoT.2019.00028.

[2] R. dos Anjos Gomes, R. B. de Lima, T. do Vale Brsil, ” Impact of the fires in the amazon with the incidence of cases of respiratory problems in the population” in Revista Foco, vol. 17, n. 12, pp. 1-19, 2024. doi: https://doi.org/10.54751/revistafoco.v17n12-004.

[3] S. Kumar and A. Jasuja, ”Air quality monitoring system based on IoT using Raspberry Pi” in International conference on computing, communication and automation (ICCCA). IEEE, 2017. p. 1341-1346. doi: 10.1109/CCAA.2017.8230005.

[4] M. H. A. Abdelsamea, M. Zorkany and N. Abdelkader, ”Real Time Operating Systems for the Internet of Things, Vision, Architecture and Research Directions” in *World Symposium on Computer Applications & Research* (WSCAR). IEEE, 2016, pp. 72-77. doi: 10.1109/WSCAR.2016.21.

[5] N. Min-Allah and S. Alrashed, ”Smart campus—A sketch”, *Sustain. Cities Soc.*, vol. 59,p. 102231, 2020. doi: https://doi.org/10.1016/j.scs.2020.102231.

[6] M. Liu and L. Li, ”The construction of smart campus in universities and the practical innovation of student work” in *Proceedings of the 1st International Conference on Information Management and Management Science*, 2018, pp. 154-157. doi: https://doi.org/10.1145/3277139.3278307.

[7] T. Kumar and A. Doss, ”AIRO: Development of an Intelligent IoT-based Air Quality Monitoring Solution for Urban Areas”,*Procedia Computer Science*, vol. 218, pp. 262-273, 2023. doi: https://doi.org/10.1016/j.procs.2023.01.008.

[8] M. Simoni, S. Baldacci, S. Maio, S. Cerrai, G. Sarno and G. Viegi, ”Adverse effects of outdoor pollution in the elderly”, Journal of thoracic disease, 2015, vol. 7, no. 1, pp.34. doi: 10.3978/j.issn.2072-1439.2014.12.10.

[9] F. Fatemi, A. Dehdashti and M. Jannati, ”Implementation of Chemical Health, Safety, and Environmental Risk Assessment in Laboratories: A Case-Series Study”, Journal of Occupational Hygiene Engineering, 2018, vol. 5, no. 2 p. 20-27, 2018. doi:10.3389/fpubh.2022.898826.

[10] N. Cunha et al.” Arquitetura de Monitoramento de Qualidade de Ar baseada no LoraWAN e FIWARE em um Campus Universitário”, in Anais do XV Workshop de Computação Aplicada à Gestão do Meio Ambiente e Recursos Naturais, Brasília/DF, 2024, pp. 169-178, doi: https://doi.org/10.5753/wcama.2024.3038.

[11] B. S. N. Reddy et al. ”A Comprehensive Review on Functional Analysis of Real-Time Operating Systems”, in 3rd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA). IEEE, 2023. p. 1098-1102. doi: 10.1109/ICIMIA60377.2023.10426594.

[12] P. Hambarde, R. Varma and S. Jha, ”The survey of real time operating system: RTOS” in International Conference on Electronic Systems, Signal Processing and Computing Technologies. IEEE, 2014. pp. 34-39. doi: 10.1109/ICESC.2014.15.

[13] B. Septian, M. Misbahuddin and F. Arkan, ”Freertos based air quality monitoring system using secure internet of things”, Jurnal Teknik Informatika (JUTIF), vol. 3, n. 1, p. 147-153, 2022. doi: https://doi.org/10.20884/1.jutif.2022.3.1.172.

[14] T. S. Gunawan, M. F. Sabar, H. Nasir, M. Kartiwi and S. M. A. Motakabber, ”Development of smart chicken poultry farm using RTOS on Arduino” in *IEEE International Conference on Smart Instrumentation, Measurement and Application* (ICSIMA). IEEE, 2019, pp. 1-5. doi: 10.1109/ICSIMA47653.2019.9057310.

[15] L. D. O. Turci, “Real-Time Operating System FreeRTOS Application for Fire Alarm Project in Reduced Scale,” International Journal of Computing and Digital Systems, 2017, vol. 6, no. 4, pp. 198-204. doi: http://dx.doi.org/10.12785/IJCDS/060405.

[16] H. Wang, J. Huo, J. Hu and T. Wang, ”Research on Real-time Monitoring System of Multiscenario Health Data Based on Internet of Things” in *3rd International Conference on Digital Society and Intelligent Systems* (DSInS). IEEE, 2023, pp. 332-336. doi: 10.1109/DSInS60115.2023.10455560.

[17] Janeera, D. A., et al. ”Smart embedded framework using arduino and IoT for real-time noise and air pollution monitoring and alert system.” 2021 International conference on artificial intelligence and smart systems (ICAIS). IEEE, 2021. doi: 10.1109/ICAIS50930.2021.9396041.

[18] L. F. Ugarte, M. C. Garcia, E. O. Rocheti and E. L. Jr, ”LoRa communication as a solution for real-time monitoring of IoT devices at UNICAMP” in *International Conference on Smart Energy Systems and Technologies* (SEST). IEEE, 2019, pp. 1-6. doi: 10.1109/SEST.2019.8849100.

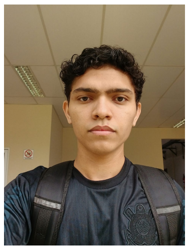
[19] Benammar, Mohieddine, et al. ”A modular IoT platform for real-time indoor air quality monitoring.” Sensors 18.2 (2018): 581. doi: https://doi.org/10.3390/s18020581.

Noedson Silva was born in Brazil. He is an undergraduate student at the Federal University of Southern and Southeastern Pará. His interests include: internet of things and embedded systems.

Elton Alves was born in Brazil. He holds a degree in Computer Engineering from the Federal University of Pará and a PhD in Electrical Engineering (Energy Systems) from the Federal University of Pará. He is currently an adjunct professor at the Federal University of Southern and Southeastern Pará. His areas of interest include: Artificial Intelligence, Machine Learning, Embedded Systems and Atmospheric Discharges.

Warley Júnior was born in Brazil. He received his Ph.D. in Computer Science from the Federal University of Pernambuco, Recife-Brazil, in 2018. He is currently an adjunct professor at the Federal University of Southern and Southeastern Pará. His current research interests are: Software Defined Networks, Mobile Edge Computing, Fog Computing, Mobile Cloud Computing, and Machine Learning.

Alife Moraes was born in Brazil. He is an undergraduate student at the Federal University of Southern and Southeastern Pará. His interests include programming and software development, with a focus on creating practical applications using technologies such as Python, JavaScript, and various frameworks for web and graphical user interface (GUI) development.



Nandson Silva was born in Brazil. He is an undergraduate student at the Federal University of Southern and Southeastern Pará. His interests include: machine learning and backend development.

Andson Balieiro was born in Brazil. He received his Ph.D. in Computer Science in 2015 from the Federal University of Pernambuco (UFPE), Brazil. He is currently a professor at the Center for Informatics (CIn) of UFPE. He has worked on R&D projects funded by companies (e.g., Morotola Mobility and Ericsson) and government institutions (e.g., National Council for Scientific and Technological Development-CNPQ). He won the FET Best Paper award at the 31st Wireless and Optical Communications Conference (WOCC). His research interests include ultra-reliable and low-latency communications, 5G/6G networks and their key enablers such as cognitive radio, network slicing, network function virtualization, multiple-access edge computing and software-defined networking, as well as machine learning applications.