A COMPREHENSIVE COMPARATIVE ANALYSIS OF FREERTOS AND ZEPHYR OS FOR EMBEDDED AND IOT SYSTEMS

Nahian Yasmeen[1], Nurul Azma Zakaria[1*], Zuraida Abal Abas[1], Jihadul Islam[1], and Daniel Ismail[2]

[1] Fakulti Teknologi Maklumat dan Komunikasi,
Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya,
76100 Durian Tunggal, Melaka, Malaysia.

[2] PWB Design Service LLC, 7145 N, State Road 1, Suit 1C, Ossian,
Indiana, 46777, United State of America.

*Corresponding Author's Email: azma@utem.edu.my

Article History: Received xxxxx; Revised xxxx; Accepted xxxx

ABSTRACT: An essential part of embedded system is real-time operating system. Choosing the right RTOS is vital to maintain reliability and performance of any system. In this paper we will discuss and compare two reputed and widely known real-time operating systems named Zephyr and FreeRTOS. Zephyr is backed by the Linux foundation whereas FreeRTOS is supported by Amazon web services. By carefully analyzing their characteristics, performance indicator and its dependability, this paper aims to offer information that might help developers to make a wise decision.

KEYWORDS: *Real Time Operating System, Embedded System, Zephyr OS, FreeRTOS*

1.0 INTRODUCTION

Embedded systems are present everywhere, operating a wide range of system and application from house appliances and consumer electronics to industrial automation [1]. An RTOS, also known as the backbone of an embedded system, allocates hardware resources, schedules work, and ensures timely execution of tasks [2]. The structure of RTOS is illustrated in Figure 1. Zephyr and FreeRTOS, both of those RTOS have received a significant amount of attention due to their strong feature sets, adaptability across hardware platforms, and extensive community support.

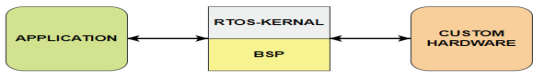

Figure 1: Structure of Real-Time Operating System

Zephyr OS, an open-source real-time operating system developed by the Linux Foundation, has become recognized for its versatility, security features, and comprehensive hardware compatibility. The operating system includes characteristics such as a tiny footprint, effective memory management, and varied board driver support, which makes it excellent for IoT devices [3]. Zephyr OS allows developers to tailor the system to fit specific application requirements which makes it more flexible. On the other hand, FreeRTOS is widely known for its simplicity, stability and its reputation as a long-term system in embedded system [4]. The well documented instruction and ease of use make FreeRTOS an excellent choice for both beginners and experienced developers [5].

This article compares and provides a comparative analysis of Zephyr and FreeRTOS, focusing on major features, performance metrics and common scenario of usages. By investigating

these features, developers and engineers may make more educated judgments when choosing an RTOS for their embedded systems projects.

The following Section 2 explores the approach, covering from fundamentals of operating systems such RTOS and super loops to the study comparing Zephyr OS with FreeRTOS based on literature review, features, and case studies. We review the results in Section 3, and Section 4 brings the article to a conclusion.

2.0 METHODOLOGY

In order to analyze Zephyr and FreeRTOS, this methodology uses a comparative analysis approach of literature review, feature comparison, and use case analysis. With the introduction of RTOS, in the literature review, the gathered information on the existing study on their features, use case and performance would be presented. In feature comparison, a detailed comparison would be performed where the architecture, memory management and security specifications would be presented of those two RTOS. The performance evaluation was gathered based on the task switching task, memory footprint and latency from the existing studies. Moreover, tests were performed on the same and identical MCU module to ensure compatibility. Lastly, the use case analysis would be based on the common factors which would involve real-world applications to understand the practical advantages and limitations of each RTOS.

Super loop and multi-threaded applications are a common term in embedded system. When it comes to bare-metal programming, super loop is an unavoidable term. Super loop is a system stucture where it consists all the necessery tasks in it's executable loop [6]. The structure of super loop is demonstrated in Figure 2. In super loop, the system is unable to execute multiple tasks simultaneously. However, with potential interrupt service routine (ISR), the running task or program can be taken to a halt and the system can be directed to execute arbitrary code based on the internal or external event.

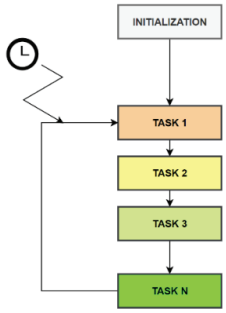

Figure 2: Super Loop Architecture

On the other hand, RTOS, unlike super loop architecture illustrated in Figure 3, can execute multiple tasks simultaneously and independently [7]. RTOS is crucial for time-sensitive tasks and operation where multiple tasks and their processes must be performed simultaneously. RTOS tasks can be executed at predetermined specified times and each of the functions is activated and triggered when the timer's duration expired as shown in Figure 4 [8].

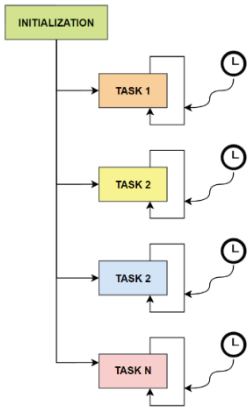

Figure 3: Generic RTOS Architecture

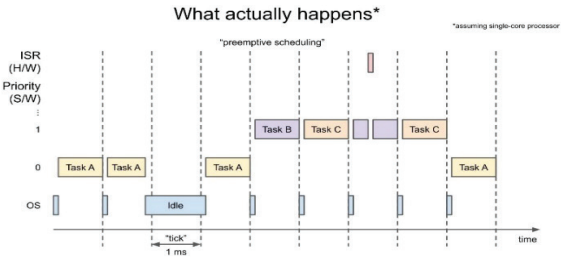

Figure 4: RTOS Task Scheduling Algorithm in RTOS [8]

2.1 Literature Review

FreeRTOS is widely known and recognized for its configurability and popularity in embedded systems [9]. It has been evaluated for its ability to function under single event upset faults, underlining the selectivity in fault tolerance technique [10]. When dealing with ARM devices, FreeRTOS has been a well adopted RTOS, and various analysis has been conducted to identify the vulnerabilities in the application.

According to [11], the analysis on FreeRTOS under multiple scheduling rules by applying SPIN model checker has detected and reproduces multiple faults. Those failures emphasize the challenge of reducing multitasking concerns like deadlocks and starvation. Furthermore, the incorporation of FreeRTOS kernal, API and libraries in a unique RiotModel framework highlights its utility in tackling multithreading difficulties in IoT devices, notably in real-time digital healthcare applications, displaying flexibility and versatility in many cases [12].

Zephyr has gained its attention due to its lightweight and scalable nature. According to the study [13] conducted on the Zephyr RTOS, the operating system demonstrated in managing various data input while maintaining real-time monitoring capabilities, and enabling adaptive and responsive system which are critical for real-time application in a variety of contexts.

As discussed in [14] the security framework in Zephyr OS highlights a significant strength with its up-to-date features which includes, secure boot, cryptography, firmware upgrades, which provides high level integrity and protection against malicious assaults. Moreover, its robust infrastructure is further reinforced by a modular and flexible design, which allows the application to main its own and specialized security solutions. Although Zephyr being known for its modularity among hardware and configurable nature, the steep learning curve makes it difficult for beginners [15].

2.2 Feature Comparison

In this section, a comparative analysis of their features would be provided based on their key features which adhere to embedded system and real time operating systems. Table 1

provides a basic comparison of Zephyr OS and FreeRTOS on the basis of architecture, supported hardware, features, ecosystem, performance and security.

Table 1: Comparative Analysis of Features: FreeRTOS and Zephyr RTOS

Feature	FreeRTOS	Zephyr OS
Architecture and Design	Simple, small footprint, priority-based preemptive scheduling	Feature-rich, scalable, multi-threaded, preemptive, and cooperative scheduling
Supported Hardware	Wide range including ARM, AVR, MIPS; popular platforms like STM32, ESP32	Broad range including ARM, x86, RISC-V; extensive support for various boards
Features	Basic real-time features: task management, timers, queues, semaphores	Rich set of features: Bluetooth, Wi-Fi, USB, file systems, networking protocols
Community and Ecosystem	Large, active community; extensive documentation; supported by AWS [16]	Growing community; robust documentation; backed by the Linux Foundation [16]
Performance and Efficiency	Low latency, fast context switching, suitable for real-time applications	Good task scheduling and context switching, higher memory usage and CPU overhead
Security	Supported by AWS, regular updates, integrated with AWS IoT services	Regular updates, rigorously tested codebase, strong commitment to security

Table 2 emphasizes the detailed nature of the comparison between FreeRTOS and Zephyr and specifies the focus on embedded and IoT systems. We evaluated six key criteria: real-time performance, connectivity, cloud integration, security, resource efficiency, and community support. Each criterion involved analyzing specific attributes such as scheduling methods, communication protocols, security certifications, hardware compatibility, feature sets, and community activity.

Table 2: In-Depth Feature Comparison of FreeRTOS and Zephyr

Feature	FreeRTOS	Zephyr
Real-Time Performance	Low latency, fast context switching, priority-based preemptive scheduling	Multi-threaded, preemptive, and cooperative scheduling
Connectivity	Basic connectivity, AWS IoT integration	Comprehensive connectivity, including Bluetooth, Wi-Fi, and more
Cloud Integration	Seamless AWS integration	Flexible integration with various cloud services
Security	AWS-supported security features, regular updates	Rigorous security measures, regular updates, OpenSSF Gold Badge
Resource Efficiency	Minimal resource usage, suitable for constrained devices	Efficient performance, higher memory and CPU usage
Community and Support	Large community, extensive documentation, AWS support	Growing community, robust documentation, Linux Foundation and industry support

2.3 Use Case Analysis

This section outlines a comparative analysis of Zephyr and FreeRTOS based on their suitability for embedded systems and IoT project. For the analysis we consider the factors such as real-time performance, connectivity, security and community support. Although these two real-time operating systems serve relevant applications and systems but differs in their functions and capabilities.

In Table 3, comparison of Zephyr OS and FreeRTOS based on schedular type, target, network stacks and connectivity has been illustrated [16]. From the table we can see that preemptive and round-robin scheduling is supported by FreeRTOS, which includes network stack support for TCP/IP, CAN and CoAP, as well as BLE, Wi-Fi and Ethernet connectivity [17]. On comparison, Zephyr supports preemptive round-robin, and cooperative scheduling as well as a larger range of network protocols such as Modbus and LoRa [18].

Benchmarks are an important factor for optimization, competitive insights and continuous learning. In Figure 5, the average cycle count run of each RTOS Bench test case was calculated by repeating each test thousand times [18]. Zephyr demonstrated faster context switch times compared to FreeRTOS and lower interrupt latency. However, FreeRTOS performs better in mutex block and mutex unblock. On the other hand, Zephyr OS is quicker in blocking while FreeRTOS is more efficient in unblocking.

Table 3: Comparison on Schedular Type, Target, Network Stacks and Connectivity: FreeRTOS and Zephyr RTOS [16]

			Scheduler type				Target			HAL		Network Stacks						Connectivity					
Name	Developer/Sponsor	License	Preemptive	Round-Robin	Cooperative	Other	IoT	General Purpose	Safety (SIL-3)	Language	OS specific	Manufacturer	File System	TCP/IP	CoAP	CAN	Modbus	BLE	WiFi	LoRa	Ethernet	Group
Zephyr Project	Linux Foundation	Apache 2.0	●	●	●	●	●	○	◉	C	●	○	●	●	●	●	○	●	●	●	●	A
FreeRTOS	Amazon	MIT	●	●	●	○	○	●	●[1]	C	○	●	●[1]	●[1]	○	○	○	○	○	○	○	B

Comparison of commonly used RTOS, ●: true, ○: false, ◉: announced but not yet implemented, ⓡ: no information given, [1]: sold separately, [2]: SIL-4

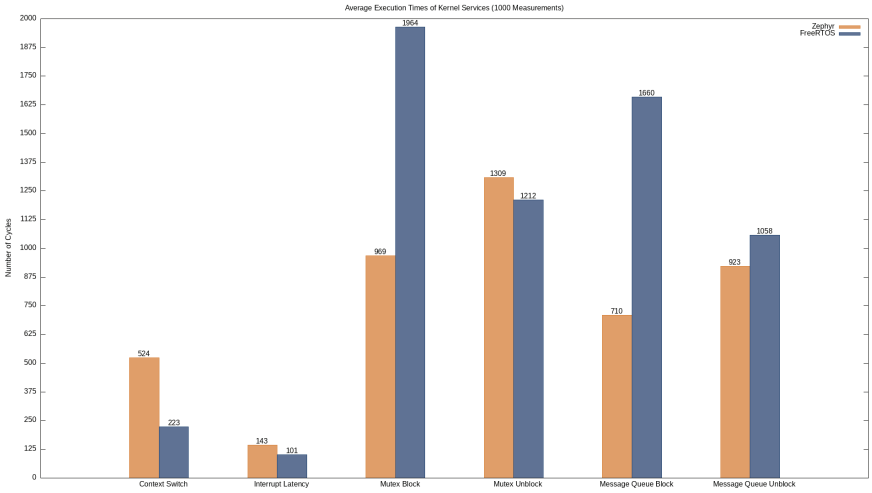

Figure 5: Comparing Benchmark Performance of Zephyr OS & FreeRTOS [18]

3.0 RESULT AND DISCUSSION

Overall, the comparative analysis on various factors reveals various metrics and demonstrates strength and weakness in their performance and feature sets. For real-time performance Zephyr OS proves to be prominent in the context of interrupt latency making it a better choice for applications where these factors are critical. Its multi-threaded, pre-emptive, and cooperative scheduling capabilities provide for greater flexibility in complicated applications. While FreeRTOS is slower in switching and interrupt latency, but faster in mutex operations which makes it a better option in managing mutual exclusion efficiently. This is essential for applications and systems that rely heavily on mutexes for task synchronization.

For connectivity and cloud when using Amazon's cloud service, FreeRTOS provides advantage over other operating systems. Conversely, the rich and flexible features of Zephyr make it adaptable and a versatile option for different deployment scenarios. In regard to

security, Zephyr OS due to its regular updates and rigorous measures in security, it has earned the achievement of OpenSSF gold badge [19]. Nevertheless, FreeRTOS is supported by AWS, benefits from Amazon's security features and regular updates which makes it a reliable option for IoT applications.

Regarding resource efficiency, FreeRTOS has an advantage for small, resource-constrained applications as it is known for its minimal resource usages. Zephyr tends to have higher memory and CPU usage due to its rich features set, which makes it compatible for highly resource-constrained devices. Community and support are an important factor which determines the evolution of technology. Zephyr benefits from the support of the Linux Foundation and from major industries such as Intel and Nordic, impacting the growth of the community and its users. On the other hand, FreeRTOS has a large and active amount of community members thanks to its simplicity and extensive documentation. The FreeRTOS community is full of resources with multiple active forums which aids ample resources and assistance during the development phase.

3.0 CONCLUSION

In conclusion, both Zephyr and FreeRTOS have their unique strengths and weaknesses. The strengths of those RTOSes, target its segment of applications and requirements. The extensive connectivity, robust security features and adaptability of Zephyr makes it an appropriate real time operating system for security-sensitive, IoT and high resource constrained edge devices. In contrast, FreeRTOS seems to be a better fit for edge devices with minimal resource usages and makes it an appropriate real time operating system for applications integrated with Amazon's cloud services. The simplicity in instruction and documentation of FreeRTOS makes it a good choice for beginners whereas someone needs to have a minimal level of understanding to conduct development on Zephyr OS. While FreeRTOS is a viable option for beginners because of its simplicity in instructions and documentation, Zephyr OS development requires a certain amount of knowledge to begin the endeavor.

ACKNOWLEDGMENTS

The authors would like to express their sincere gratitude to Universiti Teknikal Malaysia Melaka and the Fakulti Teknologi Maklumat dan Komunikasi for their unwavering support and resources provided for this research. This study was funded by the Kesidang Scholarship Program. Additionally, the authors extend their heartfelt thanks to all team members for their invaluable support and contributions throughout the study.

REFERENCES

[1] J. Wang, "Research Status and Prospect of Embedded System," in *6th International Conference on Mechatronics, Control and Electronic Engineering (MCEE 2023)*, 2023.

[2] C. Wulf, N. Charaf and D. Goehringer, "Scheduling of Hardware Tasks in Reconfigurable Mixed-Criticality Systems," in *2022 IEEE 30th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, New York, 2022.

[3] Y.-k. Lee, Y. kim and J.-n. kim, "Implementation of TLS and DTLS on Zephyr OS for IoT Devices," in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju, 2018.

[4] Y. Xing, D. Wang and Y. Zhao, "Analysis and Implementation of an Embedded System Platform Based on FreeRTOS and Cortex-M3," in *ICCIS 2017: Proceedings of the 2017 2nd International Conference on Communication and Information Systems*, New York, 2017.

[5] C. S. Stangaciu, M. V. Micea and V. Cretu, "An Analysis of a Hard Real-Time Execution Environment Extension for FreeRTOS," *Advances in Electrical and Computer Engineering*, vol. 15, pp. 79-86, 2015.

[6] M. Nahas, "Implementation of Highly-Predictable timeTriggered Cooperative Scheduler using Simple," *International Journal of Electrical & Computer Sciences IJECS-IJENS,* vol. 11, pp. 36-41, 2011.
[7] L. M. R., P. O. and C. J.-P., "A generic RTOS model for real-time systems simulation with systemC," *Proceedings Design, Automation and Test in Europe Conference and Exhibition,* vol. 3, pp. 82-87, 200.
[8] S. Hymel, "Introduction to RTOS - Solution to Part 3 (Task Scheduling)," MAKER.IO, [Online]. Available: https://www.digikey.my/en/maker/projects/introduction-to-rtos-solution-to-part-3-task-scheduling/8fbb9e0b0eed4279a2dd698f02ce125f.
[9] T. Schlotterbeck Suárez, Exploration of FreeRTOS on a RISC-V Architecture, 2022.
[10] A. Bosio, M. Rebaudengo and A. Savino, "Reliability assessment of FreeRTOS in Embedded Systems," in *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S)*, Baltimore, 2022.
[11] C.-K. Lin and B.-Y. Wang, "Analyzing FreeRTOS Scheduling Behaviors with the Spin Model Checker," *arXiv preprint arXiv:2205.0748,* 2022.
[12] P. Pratim Ray, "Parametric Analysis of Resource-Constrained Devices for Internet of Things Using FreeRTOS," *Inventive Computation and Information Technologies,* vol. 563, 2023.
[13] W. Qi, H. Su and A. Aliverti, "A Smartphone-Based Adaptive Recognition and Real-Time Monitoring System for Human Activities," *IEEE Transactions on Human-Machine Systems,* vol. 50, pp. 414-423, 2020.
[14] U. Tariq, "Combatting ransomware in ZephyrOS-activated industrial," *Heliyon,* vol. 10, pp. 1-19, 2024.
[15] Z. Cekerevac, Z. Dvorak and T. Pecnik, "Top seven iot operating systems in mid-2020," *MEST JOURNAL,* vol. 8, pp. 47-68, 2020.
[16] "FreeRTOS vs. Zephyr - What's the Difference?," [Online]. Available: https://thisvsthat.io/freertos-vs-zephyr.
[17] F. Javed, M. Afzal Khalil, M. Sharif and B.-S. Kim, "Internet of Things (IoT) Operating Systems Support, Networking Technologies, Applications, and Challenges: A Comparative Review," *IEEE Communications Surveys & Tutorials,* vol. 20, pp. 2062-2100, 2018.
[18] R. Seetha and R. Nandakumar, "Benchmarking on RISC-V Core and Performance Analysis of Two Open-Source Real-Time Operating Systems," in *Proceedings of the NIELIT's International Conference on Communication, Electronics and Digital Technology*, New Delhi, 2023.
[19] S. Banik and V. Zimmer, Correction to: Firmware Development, Apress, Berkeley, CA, 2022, pp. 1-127.