Análisis Comparativo de Sistemas Operativos Embebidos Modernos: Zephyr, FreeRTOS, NuttX y RTEMS

Carlos Enrique Elizondo Alfaro
Escuela de Ingeniería Electrónica
Instituto Tecnológico de Costa Rica
Email: alfaquillo@estudiantec.cr

***Resumen*—Los sistemas operativos embebidos han evolucionado desde implementaciones minimalistas hasta plataformas robustas capaces de gestionar redes, seguridad y tiempo real. Este ensayo analiza las características técnicas de cuatro sistemas operativos representativos Zephyr, FreeRTOS, NuttX y RTEMS evaluando su arquitectura, planificación de tareas, soporte de hardware y aplicabilidad en entornos industriales e IoT. A partir de documentación oficial y literatura científica, se identifican las tendencias de adopción y los retos de integración en sistemas críticos, priorizando la eficiencia, seguridad y mantenibilidad del software embebido.**

I. Introducción

El panorama tecnológico actual está siendo construido y moldeado por una gran cantidad de dispositivos que los usuarios tienen disponibles del Internet de las Cosas (IoT), los cuales se integran en aplicaciones que van desde la domótica, los personales hasta la industria 4.0 y los vehículos autónomos. Estos dispositivos operan típicamente con restricciones de potencia, memoria y capacidad de procesamiento, a la vez que deben ejecutar lógica de aplicación compleja y, a menudo, manejar múltiples tareas de forma concurrente y confiable [1]. En este contexto, los sistemas operativos de propósito general son inadecuados debido a su falta de determinismo y su alta huella de memoria. Los Sistemas Operativos de Tiempo Real (RTOS) proveen la respuesta necesaria, actuando con software ligero que garantiza que las tareas críticas se ejecuten dentro de tiempos estrictos [2].

II. Características

Un RTOS no se define necesariamente por su velocidad, sino por la previsibilidad y confiabilidad de su respuesta a eventos externos. Es decir, sistema en tiempo real debe cumplir con sus especificaciones funcionales y temporales, donde un fallo en el tiempo de respuesta puede considerarse tan grave como un fallo funcional [3]. Este ensayo tiene como objetivo explorar cómo evoluciona y cuál es el estado actual de los RTOS de código abierto más relevantes para el IoT. Mediante un análisis comparativo de FreeRTOS, Zephyr, NuttX y RTEMS, y apoyándose en estudios de desempeño y documentación oficial, se busca proporcionar un marco para la selección del RTOS adecuado según los requisitos específicos de una aplicación de IoT.

Desde el punto de vista de arquitectura, un sistema embebido con RTOS se construye sobre un núcleo o kernel que gestiona la CPU, la memoria y los periféricos. La característica principal es el planificador (scheduler), que decide qué tarea se ejecuta en cada momento. Los RTOS utilizan comúnmente algoritmos de planificación basados en prioridades, como el de Prioridad Fija con Expulsión (Priority Preemptive Scheduling), que asegura que la tarea de mayor prioridad lista para ejecutarse siempre obtenga la CPU [4].

Para los dispositivos de IoT, las demandas específicas van más allá del mero determinismo. La eficiencia energética es muy importante, ya que muchos dispositivos funcionan con batería. Además, deben soportar una amplia gama de hardware, desde microcontroladores de 8 bits hasta SoCs de 32 y 64 bits, y ofrecer capacidades robustas de conectividad (Wi-Fi, Bluetooth, BLE, LoRa, etc.) de forma nativa o mediante librerías [1]. La arquitectura de ”Fog of Everything”[5] explica este concepto, requiriendo que los nodos de edge computing no solo capturen datos, sino que también los procesen de manera inteligente, lo que aumenta la carga sobre el RTOS subyacente.

Finalmente, en dominios críticos como el automotriz o aeroespacial, el RTOS debe facilitar el cumplimiento de estándares de seguridad funcional como ISO 26262 [6], lo que impone requisitos estrictos sobre el diseño, la verificación y la documentación del software.

III. RTOS Actuales

Ahora vamos a comparar los RTOS open source mas importantes que podemos encontrar en el mercado.

En el caso de FreeRTOS, este sistema operativos fue adquirido por Amazon Web Services (AWS) en 2017, FreeRTOS es el RTOS más ampliamente adoptado en el mundo de los microcontroladores. Su principal fortaleza es su simplicidad y su pequeña huella de memoria, lo que lo hace ideal para dispositivos con recursos extremadamente limitados [7]. El kernel de FreeRTOS es del tipo monolítico, pero su diseño modular permite incluir solo los componentes necesarios para la aplicación. Su integración profunda con la nube de AWS a través de librerías permite a los desarrolladores conectar dispositivos de forma segura y sencilla a servicios en la nube para gestión, actualizaciones over-the-air (OTA) y análisis [8].

Como parte negativa en comparación con alternativas más modernas, su ecosistema de drivers y su soporte para hardware más complejo pueden ser menos extensos.

Desde el punto de vista de Zephyr, este sistema operativo esta bajo el soporte de The Linux Foundation, Zephyr es un RTOS moderno y altamente modular diseñado específicamente para dispositivos IoT conectados y con recursos limitados. Una de sus características más destacadas es su modelo de configuración basado en Kconfig y Devicetree, heredado de Linux, permite un nivel de personalización y portabilidad del hardware [9]. Zephyr soporta una amplia gama de arquitecturas (ARM, x86, RISC-V, etc.) y cuenta con un conjunto extenso y nativo de drivers y protocolos de comunicación. Recientemente, el proyecto ha puesto un fuerte énfasis en la seguridad, documentando un marco de seguridad proactivo que aborda amenazas comunes en IoT [10]. Su arquitectura puede considerarse un híbrido, acercándose a un micronúcleo bien estructurado, lo que facilita la contención de fallos.

Ahora si nos vamos por NuttX, este se distingue por su objetivo de mantener una alta compatibilidad con los estándares POSIX y ANSI. Esta característica lo hace particularmente atractivo para desarrolladores que provienen de entornos Linux o Unix, ya que reduce la curva de aprendizaje y facilita el desarrollo de aplicaciones existentes a este sistema operativo [11]. Si lo comparamos con FreeRTOS, NuttX presenta una arquitectura más compleja y con muchas características, siendo similar a un sistema operativo tipo Unix en miniatura. Esto lo hace adecuado para dispositivos más potentes, como SoCs de aplicación, donde se requiere un sistema de archivos complejo, manejo avanzado de memoria o la ejecución de aplicaciones más grandes. Su uso se extiende a sistemas embebidos que demandan una interfaz de usuario o capacidades de networking avanzadas.

Y finalmente, si analizamos RTEMS, este sistema operativo es desarrollado para sistemas embebidos en el ámbito aeroespacial y de defensa, RTEMS (Real-Time Executive for Multiprocessor Systems) es conocido por su robustez, alto desempeño y soporte para arquitecturas multiprocesador [12]. El diseño de este sistema operativo está orientado a aplicaciones de misión crítica donde la confiabilidad es absoluta. RTEMS ha sido utilizado en numerosas misiones espaciales, y su núcleo ha sido sometido a rigurosos análisis de rendimiento y verificación para estos entornos [13]. Si bien su huella de memoria es generalmente mayor que la de FreeRTOS o Zephyr, ofrece un conjunto de APIs maduras y estables (clásicas y POSIX) y es una opción preferente para aplicaciones de alta gama en los sectores industrial, aeroespacial y de networking.

IV. Desempeño

Ahora bien, vamos a analizar el desempeño de cada uno de los sistemas operativos y sus casos de uso especializados, tenemos un estudio que proporciona una evaluación comparativa del rendimiento de varios RTOS en dispositivos IoT [1], midiendo métricas como la latencia de interrupciones, el overhead de cambio de contexto y el consumo de memoria. Los resultados indican que no existe un ”mejorRTOS universal, sino que el rendimiento óptimo depende de la carga de trabajo y la plataforma de hardware específicas. Por ejemplo, FreeRTOS puede exhibir la menor latencia en contextos simples, mientras que Zephyr o RTEMS pueden ofrecer un mejor rendimiento sostenido en sistemas con múltiples tareas y periféricos complejos. También, existen anaálisis de otros orígnenes y validan que la elección debe basarse en pruebas empíricas con el hardware objetivo [14].

La adaptación de RTOS a arquitecturas emergentes es otro campo de innovación. El trabajo sobre la portación de un RTOS a la plataforma ESP32 basada en RISC-V demuestra la importancia de la flexibilidad del kernel para aprovechar las ventajas de estas nuevas arquitecturas abiertas [15].

Asimismo, la creciente integración de inteligencia artificial en el edge (Embedded AI) presenta nuevos desafíos. Los RTOS modernos deben gestionar eficientemente cargas de trabajo heterogéneas que combinan tareas de control en tiempo real duro con inferencias de modelos de machine learning, las cuales pueden ser computacionalmente intensivas y tener patrones de acceso a memoria impredecibles [16]. La capacidad de un RTOS para aislar estas tareas y garantizar los plazos de las críticas será un factor diferenciador en el futuro.

V. Concusiones

La selección de un RTOS para un proyecto de IoT es una decisión arquitectónica crucial con implicaciones a largo plazo en el rendimiento, la mantenibilidad y la seguridad del producto. FreeRTOS sigue siendo una opción excelente para proyectos que priorizan la simplicidad, una mínima huella de memoria y una integración directa con AWS. Zephyr Project se observa como una alternativa poderosa y moderna para dispositivos más complejos y diversos, con un fuerte enfoque en la configurabilidad, la seguridad y un ecosistema de drivers en rápido crecimiento. NuttX es la opción ideal para desarrolladores que buscan un entorno similar a POSIX en hardware con recursos moderados. Por último, RTEMS permanece como la solución de referencia para aplicaciones de misión crítica que demandan la máxima confiabilidad y soporte para hardware de alto rendimiento, incluso multiprocesador.

El futuro de los RTOS en el IoT pasará por una mayor estandarización de las APIs, mejoras en la seguridad proactiva y una gestión más eficiente de los recursos para soportar las demandas de la inteligencia artificial embebida. La comunidad de código abierto que impulsa estos proyectos será, sin duda, el motor principal de esta innovación continua.

Referencias

[1] R. Belleza and E. Pignaton, “Performance study of real-time operating systems for internet of things devices,” *IEEE Access*, 2021. [Online]. Available: https://digital-library.theiet.org/doi/10.1049/iet-sen.2017.0048

[2] T. Noergaard, *Embedded Systems Architecture*, 2nd ed. Newnes, 2013. [Online]. Available: https://gacbe.ac.in/images/E%20books/Embedded%20Systems%20Architecture%20-%20A%20Comprehensive%20Guide-%20T.%20Noergaard%20(Newnes,%202005).pdf

[3] P. Laplante, *Real-Time Systems Design and Analysis: Tools for the Practitioner*, 4th ed. Wiley, 2012. [Online]. Available: https://staff.emu.edu.tr/alexanderchefranov/Documents/CMSE443/CMSE443%20Spring2020/Laplante2012%20Real-Time%20Systems%20Design%20and%20Analysis.pdf

[4] S. C. et al., *Advances in Real-Time Systems*. Springer, 2020. [Online]. Available: https://download.e-bookshelf.de/download/0000/6170/83/L-G-0000617083-0013219313.pdf

[5] E. Baccarelli, P. Vinueza-Naranjo, M. Shojafar, M. Scarpiniti, and J. Abawajy, "Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study," *IEEE Access*, vol. PP, pp. 2169–3536, 05 2017.

[6] *ISO 26262: Road vehicles – Functional safety*, International Organization for Standardization Std., 2018. [Online]. Available: https://fscdn.rohm.com/en/products/databook/white_paper/iso26262_wp-e.pdf

[7] R. Barry, *FreeRTOS Reference Manual*, Amazon Web Services, 2023. [Online]. Available: https://www.freertos.org/media/2018/FreeRTOS_Reference_Manual_V10.0.0.pdf

[8] A. Shmagin, *How to Connect to AWS IoT Core using Amazon FreeRTOS for Embedded Devices – Hands-on Workshop Using STM32L4 Discovery Kit IoT Nod*, 2019. [Online]. Available: https://www.st.com/content/dam/AME/2019/technology-tour-2019/toronto/presentations/T5S1_Toronto_STM32L4_Amazon_FreeRTOS_M.Cantone.pdf

[9] The Zephyr Project, *Zephyr RTOS Documentation*, The Linux Foundation, 2024. [Online]. Available: https://docs.zephyrproject.org/3.7.0/zephyr.pdf

[10] ——, *Zephyr Security Overview*, The Linux Foundation, 2024. [Online]. Available: https://docs.zephyrproject.org/latest/security/security-overview.html

[11] Apache Software Foundation, *NuttX RTOS Documentation*, 2024. [Online]. Available: https://nuttx.apache.org/docs/latest/

[12] RTEMS Project, *RTEMS Classic API Guide*, 2024. [Online]. Available: https://docs.rtems.org/docs/main/c-user/

[13] F. Nicodemos, O. Saotome, and G. Lima, "Rtems core analysis for space applications," in *2013 III Brazilian Symposium on Computing Systems Engineering*, Niteroi, Brazil, 2013, pp. 125–130.

[14] Y. Eren, "Comparative study of freertos vs nuttx performance," 2024, gitHub Repository. [Online]. Available: https://github.com/nelanbu/RTOS-perf

[15] R. Gautam, D. Pujara, M. Shah, and D. Shah, "Adapting a real-time operating system to the risc-v based esp32," in *Smart Trends in Computing and Communications*, T. Senjyu, C. So-In, and A. Joshi, Eds. Singapore: Springer Nature Singapore, 2024, pp. 459–468.

[16] X. Huang, "Embedded artificial intelligence: A comprehensive survey," *Electronics*, vol. 14, no. 17, p. 3468, 2025. [Online]. Available: https://www.mdpi.com/2079-9292/14/17/3468

Cuadro I
RESUMEN DE REFERENCIAS

Referencia	**Ideas Principales**
Belleza et al. (2021)	Estudio de desempeño de RTOS para dispositivos IoT. Evaluación comparativa de sistemas operativos de tiempo real enfocados en aplicaciones de Internet de las Cosas, analizando métricas de rendimiento.
Zephyr Project (2024)	Documentación oficial del RTOS Zephyr. Describe arquitectura, características, APIs y herramientas de desarrollo para este sistema operativo de código abierto mantenido por The Linux Foundation.
Barry (2023)	Manual de referencia de FreeRTOS. Documentación completa del sistema operativo en tiempo real más popular para microcontroladores, incluyendo APIs, scheduling y gestión de memoria.
Apache Foundation (2024)	Documentación de NuttX RTOS. Sistema operativo en tiempo real tipo UNIX para microcontroladores, con énfasis en compatibilidad con estándares POSIX y amplio soporte hardware.
RTEMS Project (2024)	Guía del API clásico de RTEMS. RTOS de alto desempeño para sistemas embebidos críticos, especialmente en aplicaciones aeroespaciales, médicas y militares.
Laplante (2012)	Diseño y análisis de sistemas en tiempo real. Cubre teorías, herramientas y prácticas para el desarrollo de sistemas de tiempo real, incluyendo scheduling y análisis de desempeño.
Noergaard (2013)	Arquitectura de sistemas embebidos. Guía comprehensiva sobre diseño de sistemas embebidos, incluyendo hardware, software y consideraciones de integración.
Gautam et al. (2024)	Adaptación de RTOS para RISC-V en ESP32. Investigación sobre portabilidad de sistemas operativos en tiempo real a arquitecturas RISC-V, usando ESP32 como plataforma de prueba.
ISO 26262 (2018)	Seguridad funcional para vehículos. Estándar internacional para desarrollo de sistemas electrónicos en automóviles, aplicable a RTOS usados en automotive.
Elphinstone et al. (2015)	Evolución de microkernels L4. Análisis de 20 años de desarrollo de microkernels L4, desde L3 hasta seL4, con enfoque en seguridad y verificación formal.
Baccarelli et al. (2017)	Arquitecturas Fog computing energéticamente eficientes. Propone arquitecturas de computación en niebla para IoT que optimizan el consumo energético en sistemas distribuidos.
Chakraborty et al. (2020)	Avances en sistemas de tiempo real. Compilación de investigaciones recientes en sistemas de tiempo real, incluyendo nuevos algoritmos y aplicaciones emergentes.
Shmagin (2019)	Conexión a AWS IoT Core con FreeRTOS. Tutorial práctico para conectar dispositivos embebidos usando STM32L4 Discovery Kit a servicios cloud de AWS.
Zephyr Project (2024)	Seguridad en Zephyr RTOS. Descripción de características de seguridad, mecanismos de protección y mejores prácticas para desarrollo seguro con Zephyr.
Nicodemos et al. (2013)	Análisis de RTEMS para aplicaciones espaciales. Evaluación de desempeño y confiabilidad de RTEMS en contextos aeroespaciales con requisitos críticos de seguridad.
Eren (2024)	Estudio comparativo FreeRTOS vs NuttX. Análisis de rendimiento comparativo entre ambos RTOS, disponible como repositorio GitHub con métricas y benchmarks.
Huang (2025)	Inteligencia Artificial Embebida - Estudio comprehensivo. Revisión del estado del arte en IA para dispositivos embebidos, incluyendo optimizaciones y casos de uso con RTOS.