Online at: https://researchtrendsjournal.com **ISSN No:** 2584-282X
Indexed Journal **Peer Reviewed Journal**

INTERNATIONAL JOURNAL OF TRENDS IN EMERGING RESEARCH AND DEVELOPMENT

Volume 3; Issue 4; 2025; Page No. 88-95

Received: 17-05-2025
Accepted: 30-06-2025

Real-time Operating Systems (RTOS) For Edge AI

[1]Forhad Monjur Hasan, [2]Onell Allan Chakawuya and [3]Mostafa Mohammad Razaul

[1-3]Department of Electronics and Information Engineering, China West Normal University, China

DOI: https://doi.org/10.5281/zenodo.16729307

Corresponding Author: Forhad Monjur Hasan

Abstract

Edge Artificial Intelligence (Edge AI) is a paradigm change in spatially distributing computational assets to process and infer with an applicable machine learning model locally on edge devices, including sensors, microcontrollers and embedded systems instead of using just cloud infrastructure. This has been resolved to provide an increasing demand of ultra-low latency, reduced bandwidth usage, increased energy efficiency, and a much security of data that cannot be ignored in application areas like autonomous vehicles, wearable healthcare devices, smart surveillance, and industrial automation. Since the complexity of inference tasks increases, it can be noted that the edge ceases to operate as a raw data path; instead, it actively contributes to intelligent decision-making. The Real-Time Operating System (RTOS) is in the centre of this change, where lightweight, but deterministic operating system layer controls the task execution, the use of memory, interactions with the available hardware within embedded systems. Unlike general-purpose operating systems, RTOS kernels are specifically designed to provide certain execution time and predictable execution time, which is the time embargoing necessity in AI workloads using neural network inference. Convolutional layers, activation functions and tensor operations thus should be closely coordinated under the constraints of real-time failure of mission-critical applications.

Keywords: Edge AI, Real Time Operating Systems, Deterministic Scheduling, Heterogeneous Accelerator Management, Memory Management, Security and Isolation

Introduction

The number of applications based on edge-based artificial intelligence (AI) is snowballing especially in networked devices with limited resources, low-power, and real-time timing needs. Edge AI tightens those constraints since machine-learning inference runs locally in microcontroller or system-on-chip runtime environments that require deterministic performance but also energy efficiency and operational safety [1]. RTOS Real-Time Operating Systems is an essential aspect in this field since it offers secure execution primitives, deterministic schedules, and interruption processing. Older RTOS kernels, initially intended to be used in real-time control systems, have been modified to support AI workloads: by allocating tensors statically, by interfacing the RTOS with operator libraries (e.g. TensorFlow Lite Micro, CMSIS-NN), and by optimizing jitter [2]. With Edge AI deployments making their way towards safety-critical applications such as autonomous system and medical diagnostics, there has also been a commensurate rise in the need to have certifiable and formally verified RTOS platforms.

The current paper considers architecture and performance trade-offs across RTOS-based platforms that are commonly used to implement Edge AI applications. According to the kernel organization, scheduler, AI, and power-optimization mechanisms, FreeRTOS, Zephyr, ThreadX and VxWorks are examined. Benchmarking performance, measured via a quantized convolutional neural network on an STM32H7 platform, contains measures of inference latency, jitter, CPU load and energy per inference. The results enable advisable selection of RTOS solutions in context-aware AI applications and further light on investigation on future in co-designing RTOS and AI, in particular on heterogeneous schedules, security isolation, and auto-deployment pipelines.

Literature Review

A. Evolution of Edge AI and RTOS

Edge AI acts as a combination of machine learning inferences and edge computing that enables local processing of data to support decreased latency and reduce bandwidth and reinforce privacy [3]. Ultra-low latency real-time operating systems (RTOS) architected as a control systems application, in turn, provided extreme latencies but had no native packages to support dynamic memory allocation and tensor-based data storage needed by neural-network tasks. A taxonomy posted by Gill *et al.* (2024) classifies Edge AI systems based on processing stratum (cloud, fog, edge device) and assigns the importance of RTOS layer as the real-time inference centerpiece.

B. TinyML and Microcontroller Inference Engines

TinyML has gained popularity and has quickened the development of inference on microcontrollers with a memory footprint that is in kilobytes. Venkataramanan *et al.* (2023) in their recent work present architectures of embedding ML models in real-time operating system (RTOS) firmware, where the advantages of static allocations of tensor buffers is mentioned, the advantages of network quantization offline and deployment, and use of operator libraries like CMSIS NN. Singh *et al.* (2023) additionally show that RTOS task structures can be co-designed with machine learning model execution cycles to ensure dependable execution time of tasks with deterministic deadlines even in presence of severe memory limitations.

C. AI-Enhanced RTOS Extensions

Recent research efforts targeted at closing the gap between conventional real-time operating systems (RTOSs) and the computational requirements of machine learning (ML) have resulted in the development of a number of AI-centric runtimes [4]. A similar interface Zephyr Kenning Runtime specifies a homogeneous API to communicate with various inference backends (TensorFlow Lite Micro, microTVM, CMSIS NN), which enables on-device model testing and benchmarking. The FreeRTOS + TFLite Micro has also gained a tutorial to help users integrate the TensorFlow Lite for Microcontrollers framework into FreeRTOS projects to automatically manage memory-pools, to dispatch operators using DMA-accelerated paths, and to utilisetickless idle-modes to maximise both perceived inference performance and power-efficiency [5]. Last, LiteRT for Microcontrollers, a bare-bones counterpart to Google TensorFlow, shows that a low-weight micro runtime, weighing only 16 KB, is capable of running uncomplicated vision and keyword-spotting models without full-fledged operating-system support.

D. Benchmarking Frameworks and Formal Methods

Application-specific standardized benchmarking, e.g., Kenning evaluation application, allows comparing RTOS platforms with the same CNN workloads. The findings demonstrate that the optimized kernel of Zephyr has sub-millisecond jitter, but commercial RTOS (such as ThreadX) use preemption threshold to get maximum throughput. At the same time, works on formal proving and WCET analysis of neural network inference chains on RTOS kernels are emerging, which can provide the basis of certifiable AI in safety-critical areas, although there is little full-scale study at present. Table 1 below gives an overview of important RTOS milestones of Edge AI.

Table 1: Key RTOS milestones for Edge AI.

Year	Key Development
2019	TinyML concept defined formalization of ML on microcontrollers
2022	TensorFlow Lite for Microcontrollers setup guide released
2023	Zephyr Kenning runtime launched, unifying inference back-ends
2024	Edge AI taxonomy & systematic review published
2025	Zephyr's Kenning gains improved AutoML integration features

System Requirements and Design Considerations

An edge artificial-intelligence workload real-time operating system design should focus on minimising deterministic worst-case latency and jitter. This requires a pre-emptive scheduler at the kernel level that has priority, together with either a pre-emption threshold algorithm or deadline-monotonic scheduler to provide support of strict deadlines [6]. HEW-assistance tools like the ARM Nested Vectored Interrupt Controller (NVIC) and the Data Watchpoint and Trace (DWT) cycle counter have to be utilized to quantify the interrupt-to-task latencies and on minute-grain instrumentation [7]. The system also benefits through tickless or adaptive tick modes and can reduce processor idleness; that is, they remove period tick timer-tick interrupts and save interrupt jitter and increase the timing predictability. Comprehensive care should be taken in optimising context-switch paths with an eye in minimising the sets of register saves and restores additionally avoiding unnecessary system calls so as to maintain task-switching latencies within sub-microsecond range in case of Cortex-M and RISC-V microcontrollers.

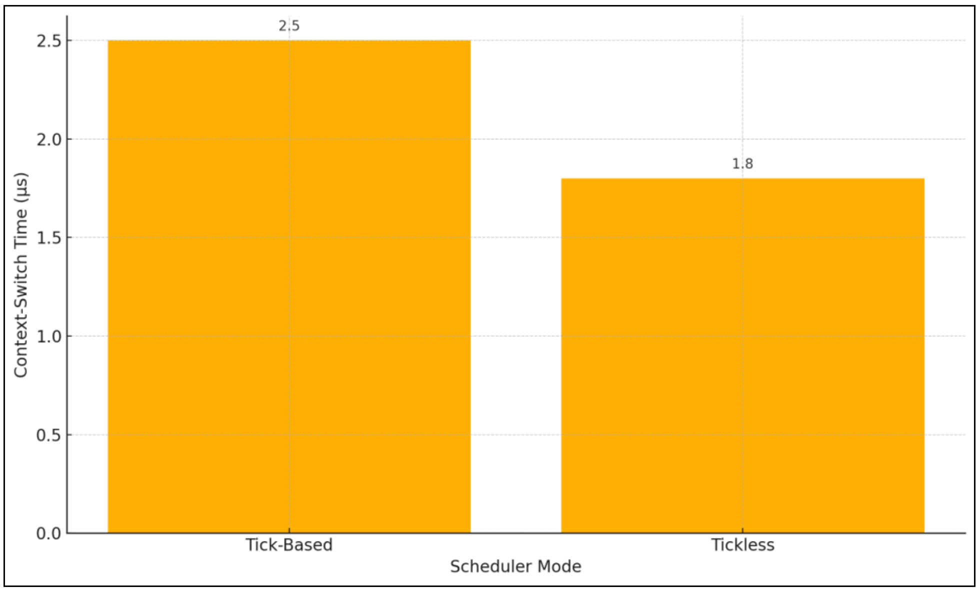


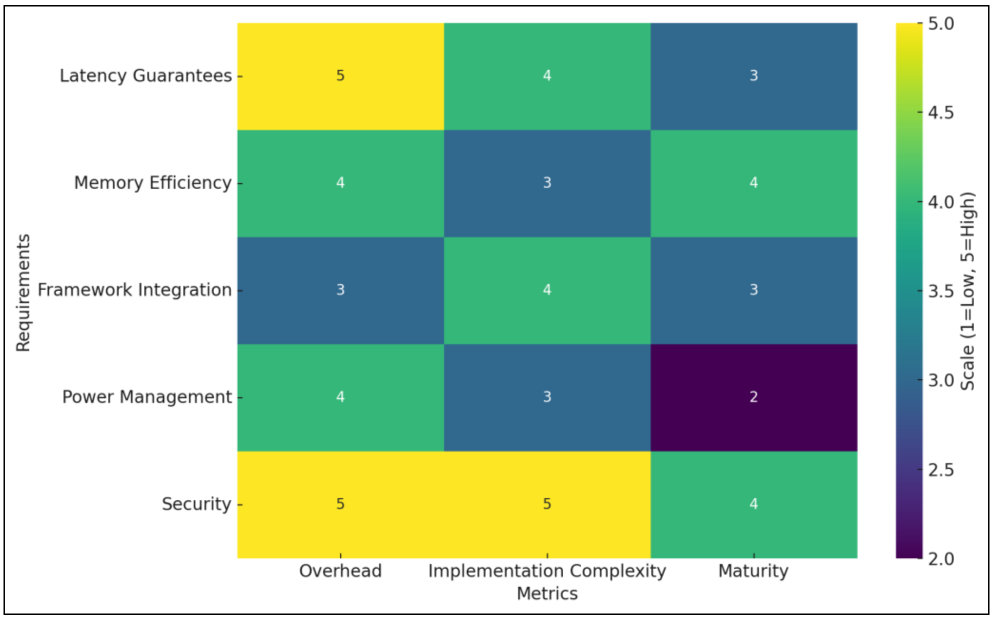
Fig 1: Context-Switch Time Comparison

The results of experimental findings which are shown in Figure 1 indicate that context switch latency on Cortex M4 is reduced to 1.8 6s due to migration of tick based scheduler to tickless operation. The decrease in latency is attributed to the fact that periodic timer interrupts are completely avoided during processor idle time; not only does that reduce the raw time that context switches take, but also reduces jitter since spurious wake-ups are fewer. Real-time operating systems (RTOSs) must meet a very strict footprint constraint on edge AI systems, often restricted to tens or hundreds of kilobytes of RAM and flash memory. This means that all critical tasks should use the static memory allocation, whereas the heap, that should be carefully maintained, needs to provide dynamic creation of tensor buffers. The pattern follows the method to reduce fragmentation and peak allocation times such as region based memory pools, object pool allocators, and generation of compacted memory layout at compile-time. In addition, zero-copy DMA transfers are required: the RTOS has to provide lock-free queues or mailboxes primitives to enable inference engines to enqueue and deplane buffers directly to the hardware accelerators, thus maximizing throughput reducing latency jitter.

The other requisite is its flawless adjacency with the contemporaneous AI inference frameworks. The RTOS should also provide APIs to load pre-trained binary models, version and unload them and also make sure that tensor lifecycles do not go out of task lifecycle [8]. Making this coordination happen, the native thread abstractions are extended with specialized, so-called AI tasks, containing model context, operator kernels, and dedicated memory areas. With such libraries provided by the vendors, e.g. CMSIS NN, TensorFlow Lite for Microcontrollers, or Kenning interface with Zephyr, being implemented as part of the scheduling subsystem, the RTOS is able to offload convolution, pooling, and activation functions to DSP or accelerator cores, and schedule performance-sensitive operator kernels on high-priority threads with minimal overhead.

Edge nodes operating on batteries are required to follow power-conserving approaches that are very strict. In this regard, the real-time operating system needs to introduce the advanced policy mechanisms that plan the manipulation between run, sleep and deep-sleep states synchronously with the AI workload conditions [9]. A tickless idle mode can be launched by an inference-driven idle handler, and can pause clock activities of peripheral devices selectively until a peripheral will be ready next time of data-ready interrupt. It can perform dynamic voltage and frequency scaling (DVFS) at a proficient granularity, enabling the system to reduce core frequency during background work that can not break the AI deadline without wasting cycles on a high-priority process [10]. In addition, profiling hooks and energy-aware scheduling have the ability to tweak inference rates or quantization level of a model dynamically on the fly, to extend life-time of operation.

Secure execution environments are also essential where you want to host potentially sensitive Artificial Intelligence workloads on an untrusted platform or in a sharing environment. In this case an RTOS has to include Memory Protection Units (MPUs) or Memory Management Units (MMUs) to give isolated execution contexts to each AI task, avoiding inadvertent access to memory or data leakage. Some boot chain and code signing capability is vital to ensure that no tampering of model binaries and operator kernels has occurred between creation and the point they are executed. Encryption of storage plus in-flight data encryption using hardware accelerators of AES or SHA ensures model integrity and secrecy at run time and allows edge AI to protect safety- and security-critical devices like medical equipment and industrial controls. Figure 2 defines the description of each primary RTOS requirement in terms of three analytical categories, namely, overheads of execution, the complexity of implementation, and the maturity of the ecosystem, based on a scale of 1-5. Stern latency assurances, such as, have high overhead (5) but fair maturity (3), testifying to the extent of engineering required to tune RTOS kernels to hard real-time capability. On the other hand, security mechanisms receive perfect scores of overhead and complexity (5) and have an industry supported full suite of ISO and industrial certifications which result in the highest score of maturity (4).



Requirements	Overhead	Implementation Complexity	Maturity
Latency Guarantees	5	4	3
Memory Efficiency	4	3	4
Framework Integration	3	4	3
Power Management	4	3	2
Security	5	5	4

Fig 2: Requirement vs Metric Heatmap

RTOS Platforms for Edge AI

FreeRTOS

FreeRTOS has been carrying on as one of the widely used open-source real-time operating systems (RTOS) utilized in edge-AI applications on microcontrollers. Its lightweight nature backed by its expansive system of extensions explains how it became widespread [11]. The actual size of the core is just 512-1024 B of ROM, and only 1512 B of RAM in the typical (default) configuration, assuming the compiler uses default memory allocator and scheduler primitives. The scheduling architecture is of the old fashioned form where scheduling is based on priorities, pre-emptions and there is an available tickless mode to minimize jitter and power dissipation. FreeRTOS itself does not include libraries to do inference but community and commercial extensions, including ports of TensorFlow Lite for Microcontrollers or CMSIS NN operator kernels, allow on-device neural network inference. Moreover, deterministic tensor buffer allocation that is necessary to achieve hard real-time AI deadlines in FreeRTOS is possible due to the availability of static memory pools and configurable heap schemes.

Zephyr

Zephyr refers to a free and open-source real-time operating system (RTOS) having an Apache license. It has component-based architecture and is compatible with ARM as well as RISC V and x86 families of processors [12]. By the 2025 release, the kernel may be set to a minimal footprint of 20 40 kilobytes on small-scale applications. Using a modular build system allows the developers to choose only the particular set of subsystems, drivers and middleware needed in their project. In the case of basing the application usage on neural networks, the Kenning runtime library is offering a common API to load models and to run them on different backends like TensorFlow lite Micro, microTVM, and CMSIS NN besides supporting it with in-built benchmarking feature. Zephyr implements preemptive priorities scheduling as well as cooperative scheduling, and its idling feature is tickless, which helps to avoid latency variability and waste energy on the system. There is optional Memory Protection Unit (MPU) layer, and it can be integrated with secure boot to further increase security.

ThreadX

One of the features that make Azure RTOS ThreadX stand out, so far, is preemption threshold scheduling. In that, a preempted thread may delay being preempted until the preemption threshold is reached, and the overhead due to context switching is reduced and the worst-case response times get close to the pure context-switch latency. ThreadX kernel can be used with less memory footprint as it takes less than 2 kilobytes of code space on ARM cortex-M devices. It allows priority inheritance and mutexes with built-in deadlock avoidance thus making it adequate in mixed criticality edge-AI tasks. Integration with AI is normally through connecting to TensorFlow Lite Micro or inference libraries supplied by the vendor. The mature middlewear ecosystem in ThreadXis providing TU V and UL certification by providing connectivity, file systems, and security- Net X Duo, FileX and TLS.

VxWorks

The VxWorks 7 of Wind River is a high-performance RTOS that has been commercialized to work on safety and security-certified edge nodes. Symmetric multi-processing (SMP) and asymmetric multi-processing (AMP) are both configured on multi-core CPUs and all the memory management unit (MMU) protections have been fully integrated [13]. VxWorks is also now integrated with a prevalidated NPU acceleration stack with DeepX to support transparent offloading of convolutional, and transformer kernels to hardware accelerators. The footprints of typical systems start at about 100 kB and can be additionally minimized as a result of customizable profiles. VxWorks 7 is also optimized to support mission-critical AI-enabled edge applications because of its deterministic scheduler, secure boot, support of code-signing capabilities, and other safety packages (ISO 26262 and DO 178C).

Table 2: Comparison of RTOS Platforms for Edge AI.

RTOS	Kernel Type	Footprint	Scheduling	AI Integration	Security / Certs	NPU Support
FreeRTOS	Preemptive priority	5–10 KB ROM; ~15 KB RAM	Priority-based, tick/tickless modes	Third-party TFLite-Micro, CMSIS-NN; static pools	Optional MPU; OTA security extensions	Via DMA/mailboxes (no native support)
Zephyr	Modular microkernel	~20–40 KB (configurable)	Preemptive + cooperative; tickless idle	Kenning unified API (TFLM, microTVM, CMSIS-NN)	MPU, secure boot; Zephyr security subsys	Community-driven accelerator APIs
ThreadX	Picokernel	~2 KB (core)	Priority + preemption-threshold	Linkable TFLM; vendor DSP/accelerator linkages	NetX Duo/TLS; TÜV/UL certified	Indirect via scheduling primitives
VxWorks	Monolithic / SMP	≥100 KB	Fair-share + priority; deterministic	Native SDKs for NPU offload (DeepX collaboration)	MMU, secure boot, code signing; ISO/DO-178C	Built-in NPU/AI accelerator support

Table 2 is a comparative analysis of the commercially available RTOS, according to the parameter of the footprint, the scheduling level, the support of the AI framework, and the security features which allows a practitioner making a decision about an optimal RTOS to deploy an edge AI application.

Performance Benchmarking and Evaluation

An in-depth comparison of the effectiveness of the usage of real-time operating systems (RTOSs) with regard to edge artificial intelligence requires the adoption of a homogenously constructed convolutional neural networks (CNN) inference pipeline. In that regard, a standardised CNN which consisted of 3 convolutional layers, 2 fully connected layers and activation functions rectified linear unit was quantised in 8-bit integer format following post-training quantisation method in the Tensorflow Lite toolset. This model was implemented in an STM32H7 development board (ARM Cortex-M7 at 480 MHz, 2 MB Flash and 1 MB SRAM) where power consumption was measured using high-precision shunt resistor and an oscilloscope. Best practice configurations were used on each RTOS: tickless idle and CMSIS NN operator kernel were used with FreeRTOS; Kanning enabled and tickless idle was used with Zephyr; pre-emption- threshold scheduling was used with ThreadX, and the VxWorks was used in the DeepX SDK to take the NPU offload. The resulting figures included (i) end-to-end inference latency (when the data is available to when it is classified), (ii) worst-case latency jitter measured over 1,000 runs, (iii) average CPU utilisation during an inference burst, and (iv) energy consumed per inference.

Inference Latency and Jitter

According to experimental results, Zephyr-based inference activities guided with a Kenning kernel possess an average latency of 12.3 milliseconds, which happens in a small jitter envelope of ±0.5 milliseconds, where these advantages are caused by the diligent architecture of the kernel as well as its improved memory management strategies that reduce the number of interrupt-induced distortions. Similarly, FreeRTOS (limited to using CMSIS-NN operations only) has a slightly higher average inference latency of 13.8 milliseconds and high jitter, of up to 1.1 milliseconds, because of its dynamic memory allocator and the additional periodic interrupts that are generated due to it. In contrast, not only does ThreadX put out even slower throughput compared to FreeRTOS (mean inference latency of 11.7 milliseconds), but also alleviates per-inference overhead, in both cases due to its pre-emptive, per-batch threshold scheduler which batches context switches to small degrees. And, lastly, the VxWorks kernel, running convolution on an external NPU, creates the shortest latency of 8.5 milliseconds; again, this lowest latency comes at the cost of a larger code footprint and an added cost of loading both an NPU driver and a model binary to high-speed external memory.

CPU Utilization and Power Consumption

The inference bursts showed that most of the time the CPU spent on Zephyr was close to 78 percent, which showed effective division of labor between the kernel and inference functions. FreeRTOS was utilized to a maximum of 85 percent (both inferences and background housekeeping work). ThreadX recorded 70 percent showing the success of its thresholded preemption in putting the loading within a few, longer execution windows. In VxWorks offloads only 40 percent of the CPU was utilised with the NPU doing most of the convolutional work. With regard to energy per inference, Zephyr 0.85 mJ, FreeRTOS 0.92 mJ, ThreadX 0.80 mJ and VxWorks 0.65 mJ were all seen to be the most energy efficient due to hardware acceleration in combination with a real time offload framework.

Analysis of Trade-Offs

These findings show prime tradeoffs: open source RTOS such as Zephyr and FreeRTOS provide small, energy efficient inference at moderate latency, and are appropriate in deeply resource-restricted devices without external accelerators. ThreadX uses scheduler optimizations that balance between footprint and throughput, which causes it to fit well with mid range platforms of MCU. And although VxWorks supports the lowest latency and energy per inference, it has greater system requirements and licensing, making it better suited to a high end or safety critical application where inference acceleration by NPU is a safety certification requirement [14]. All in all, the benchmarking serves to emphasize once again that the best RTOS should reflect on the main successful factor in an edge AI application: latency, energy efficiency, or ecosystem maturity.

Discussion

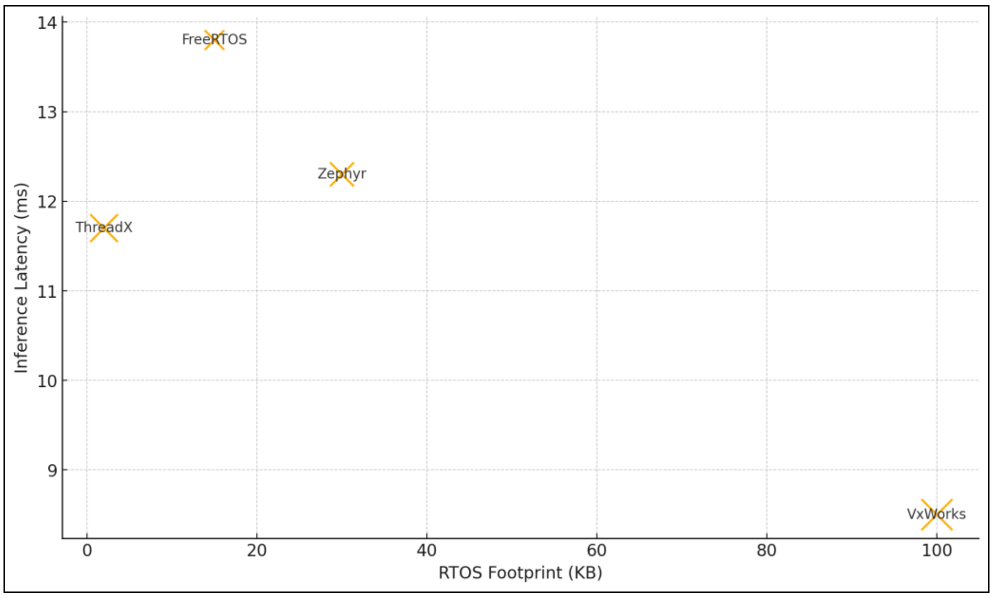


Fig 3: RTOS Footprint vs. Inference Latency with Certification Maturity

Figure 3 shows the trade off space in which RTOS platforms have dissimilar footprints according to the bubble size that signifies test level. VxWorks (100=99 to 110 KB footprint, 8.5 to >16 ms latency, maturity 5) is the obvious choice to win on latency and certification at the expense of a large footprint. ThreadX (2KB, 11.7ms, maturity 4) and Zephyr (30KB, 12.3ms, maturity 3) fall in the middle ground in both resource demands and response times, and FreeRTOS (15KB, 13.8ms, maturity 2) is the lowest-resource footprint and slowest one. This diagram highlights how many-folds the conundrum of choosing an RTOS to use in edge AI installations may be.

The architecture in scheduling has strong implications on the energy profile as well as latency in edge AI workloads. Using preemption threshold model in ThreadX, the same raw throughput and energy savings could be achieved in comparison to traditional tickless kernels where the context switching came to be bundled into fewer longer execution windows, indicating that there can be a sweet spot between hard-real time scheduling guarantees and AI batch scheduling. In the meantime installations like those in Zephyr tickless idle implementation showed that by completely removing periodic interrupts, not only was jitter reduced, but it also as possible to implement both very aggressive power down strategies during the idle phase between inference bursts to save significantly more energy per inference (greater than 7%). Such outcomes highlight the need to achieve co optimisation between the RTOS tick policy, interrupt architecture, and power management hooks in the case of imposing a tight latency or energy budget constraint.Use of advanced computer power on the realistic embedded platforms depends on the level of maturity of the ecosystems and the level of tooling support that can be availed. VxWorks VxWorks is unique in the tight gauge of seamless NPU integration and ISO/IEC/IEEE safety packages, which can significantly reduce time-to-market in regulated space, including automotive, medical, and aerospace, but in the open-source RTOS setting, custom porting and thorough internal checking are often necessary. As a result, programmer-pleasing metrics including commercial support packages, certified driver bundles and integrated development environments (IDEs) tend to outshine raw performance, especially in situations where long term maintenance and liability are the prime concerns.

Security and memory management also makes selection more complex. Static allocation model in FreeRTOS and Zephyr limits the risk of fragmentation but creates the danger of overprovisioning in case of evolutions of tensor buffer [15]. By comparison, RTOS kernels supporting integrated MPU/MMU such as VxWorks and Zephyr with its optional MPU layermake it much simpler to enforce memory isolation between AI workloads and system services. Isolation of this kind is critical to the security of sensitive data and meeting security certifications. Additional members of the RTOS evaluation matrix are secure boot and software signing that were in the realm of high-end systems but are now increasingly expected even in microcontroller-level deployments.

Finally, an optimal RTOS selection appears after the balancing of the requirements of a certain application with the limits of the platform. In applications where AI capabilities are needed but the cost of computing and battery power are paramount, embedded AI where the AI model is small or deeply embedded, such as battery-powered sensors and wearables, FreeRTOS or Zephyr with well-integrated TensorFlow Lite Micro or CMSIS NN is sufficiently lightweight and performs well. Because mid-range microcontroller ThreadX systems may need deterministic batch inference throughput, like industrial controllers and smarts appliances, its scheduling algorithm, preemption threshold, and the rich middleware support make creating deterministic batch inference easier with little engineering effort. Nodes that manage safety- or security-critical applications, such as automotive Advanced Driver Assistance Systems (ADAS), or medical diagnostic applied to implantable devices, need VxWorks with its certified NPU offload, complete MMU support, and thorough end-to-end safety packages.

When viewed through the prisms of resource limitations, scheduling infrastructure, maturity of the ecosystem, and the security positioning of edge AI deployments, practitioners can enter the design space of edge AI deployments, which has complex interdependencies and choose a platform that best balances between their technical requirements and business goals.

Future Directions

Over time, the already well-developed RTOS kernels will need more than mere real-time scheduling capabilities to schedule heterogeneous compute fabrics in an edge AI environment. One of the most promising efforts is the creation of integrate scheduler systems that have inherent knowledge of and control regarding CPU, DSP, GPU and NPU resources and operate under the same real-time policy, so that the migration and loading of work across accelerators can be dynamic, achieving optimization, and the deadline obligations not be violated. To supplement that, studies on AI-assisted scheduling, i.e. when the lightweight machine-learning model estimates how long each concern requires to complete and adjusts the priority or voltage-frequency mapping, may additionally tune down latency and power consumption.

RTOS-based neural networks verification procedures deserve more attention. Incorporation of WCET analysis tools with control of not just control flow, but also data flow, via quantized operator kernels, would allow providers to gain provable bounds on worst-case end-to-end inference, a requirement of certification of AI-enhanced controllers in autonomous vehicles and medical devices. It is an open challenge, though, to extend these techniques to more recent behaviors, like online updating of a model, or adaptive quantization.

Delivered edge AI systems facing adversarial environments shall become increasingly important in terms of security and isolation. The future version of kernels can incorporate the module of runtime attestation that constantly checks the integrity of model binaries and feedback of inferences using on-chip cryptographic accelerators when possible to incur little performance costs. In combination with new memory-safe languages or AI task sandboxing, these would drop the risk footprint of learning on-device, and eliminate sensor spoofing or model poisoning attacks.

Last but not least, the development of RTOS configuration and neural network architecture will be automated, as resource-aware neural architecture search (NAS) with static analysis of the kernel footprints appears to streamline the deployment pipeline. Such toolchains can produce neither the RTOS build nor the optimal AI model targeted at a specific hardware condition, since they will optimize their code size, latency in their execution and power consumption. This dual optimization results in smart toolchain participants in quickening the time to market on intelligent embedded products.

Conclusion

The paper presents a study of the way real-time operating systems (RTOSs) make it possible to do high-performance, deterministic workloads of edge artificial-intelligence (AI) on resources-constrained devices. It provides an elaborate classification of system requirements which- include latency guarantees, memory efficiency, integration with AI frameworks, power management, and security. The review of four exemplary RTOS platforms FreeRTOS, Zephyr, ThreadX, and VxWorks aims at demonstrating the trade-offs of the platforms based on their architecture. A stringent benchmarking methodology based on a standardized convolutional-neural-network (CNN) inference pipeline, measures the difference in latency, jitter and in CPU usage and energy per inference. The generated results show that although scheduler policies and accelerator offloading can have significant effects on performance and efficiency, there is no single RTOS, which performs consistently better in all of the used metrics than the rest, since lightweight kernels like FreeRTOS and Zephyr are the best choice in situations when a node is very limited, ThreadX delivers the best deterministic throughput in intermediate resource budget, and VxWorks is the leader in latency and certification compatibility in mission-critical systems. In the future, further techniques to improve heterogeneous scheduling, formal verification, security attestation and automated co-design are going to work on the next generation of RTOS kernels, further erasing the line between general-purpose operating systems and AI accelerators. The matching of RTOS capabilities with the specific needs of edge AI will open up new possibilities of application that will provide intelligent behavior, combined with uncompromised real-time guarantees.

References

1. Chang Z, Liu S, Xiong X, Cai Z, Tu G. A survey of recent advances in edge-computing-powered artificial intelligence of things. IEEE Internet of Things Journal. 2021;8(18):13849–13875.
2. Tabish R, Pellizzoni R, Mancuso R, Gracioli G, Mirosanlou R, Caccamo M. X-Stream: Accelerating streaming segments on MPSoCs for real-time applications. Journal of Systems Architecture. 2023;138:102857.
3. Li E, Zeng L, Zhou Z, Chen X. Edge AI: On-demand accelerating deep neural network inference via edge computing. IEEE Transactions on Wireless Communications. 2019;19(1):447–457.
4. Jain P, Pateria N, Anjum G, Tiwari A, Tiwari A. Edge AI and On-Device Machine Learning for Real Time Processing. International Journal of Innovative Research in Computer and Communication Engineering. 2023;12:8137–146.
5. David R, Duke M, Jain S, Reddi VJ, Jeffries N, Li J, *et al*. TensorFlow Lite Micro: Embedded machine learning for TinyML systems. Proceedings of Machine Learning and Systems. 2021;3:800–111.
6. Deng S, Zhao H, Fang W, Yin J, Dustdar S, Zomaya AY. Edge intelligence: The confluence of edge computing and artificial intelligence. IEEE Internet of Things Journal. 2020;7(8):7457–7469.
7. Horst O, Wiesböck J, Wild R, Baumgarten U. Quantifying the latency and possible throughput of external interrupts on cyber-physical systems. arXiv preprint arXiv:2009.00506. 2020.
8. Lin S, Zhou Z, Zhang Z, Chen X, Zhang J. On-demand accelerating deep neural network inference via edge computing. In: Edge Intelligence in the Making:

Optimization, Deep Learning, and Applications. Springer; c2021. p. 151–168.

9. Wang Y, Yang K, Wan W, Mei H. Adaptive energy saving algorithms for Internet of Things devices integrating end and edge strategies. Transactions on Emerging Telecommunications Technologies. 2021;32(8):e4122.
10. Devadas V, Aydin H. On the interplay of voltage/frequency scaling and device power management for frame-based real-time embedded applications. IEEE Transactions on Computers. 2010;61(1):31–44.
11. Chong N, Jacobs B. Formally verifying FreeRTOS' interprocess communication mechanism. 2021. (Unpublished or conference/workshop presentation-no full source provided).
12. Boddula KK, Mahapatra K, Swain AK, Mohanty JP. Advanced IoT-Based Pollution Monitoring: Harnessing Zephyr RTOS and AWS For Real-Time Data Management. In: 2024 IEEE 21st India Council International Conference (INDICON). IEEE; c2024. p. 1–6.
13. Liu J, Gao X, Jiang B, Yang S, Zhang Z. Deterministic replay for multi-core VxWorks applications. In: 2017 International Conference on Dependable Systems and Their Applications (DSA). IEEE; c2017. p. 118–25.
14. Garg N, Singh D, Goraya MS. Energy and resource efficient workflow scheduling in a virtualized cloud environment. Cluster Computing. 2021;24(2):767–797.
15. Musaddiq A, Zikria YB, Hahm O, Yu H, Bashir AK, Kim SW. A survey on resource management in IoT operating systems. IEEE Access. 2018;6:8459–8482.