



## A Comparative Analysis of VxWorks OS and Windows CE in Embedded Real-Time Applications

Saleh Abuazoum<sup>1,\*</sup>  

<sup>1</sup>Electrical and Electronic Engineering Department, Faculty of Engineering, Sebha University, Sebha, Libya

### ARTICLE HISTORY

Received 04 January 2025  
Revised 05 April 2025  
Accepted 07 May 2025  
Online 11 May 2025

### KEYWORDS

Real Time Systems;  
VxWorks;  
Windows CE;  
Embedded System.

### ABSTRACT

Real-time operating systems (RTOS) play a pivotal role in critical applications such as aerospace, industrial automation, and healthcare. Among the widely used platforms, VxWorks and Windows CE stand out for their versatility and real-time capabilities. This paper provides a comparative analysis of VxWorks and Windows CE, focusing on their architectures, scheduling mechanisms, and suitability for real-time applications. The study examines each operating system's core features, real-time performance metrics, and application-specific implementations using data from documented benchmarks and case studies. The analysis reveals that VxWorks excels in hard real-time environments due to its deterministic scheduling, while Windows CE offers greater flexibility and integration capabilities for soft real-time tasks. This study provides valuable insights for developers and engineers in selecting the appropriate RTOS based on application-specific requirements.

## تحليل مقارنة بين نظام تشغيل VxWorks و Windows CE في التطبيقات المضمنة في الزمن الحقيقي

صالح أبو عزوم<sup>1,\*</sup>

### الكلمات المفتاحية

أنظمة الزمن الحقيقي (الفعلي)  
نظام التشغيل VxWorks  
نظام التشغيل Windows CE  
الأنظمة المضمنة

### الملخص

تلعب أنظمة التشغيل في الوقت الحقيقي (الفعلي) (RTOS) دورًا محوريًا في التطبيقات الحرجة مثل الفضاء، الأتمتة الصناعية، والرعاية الصحية. ومن بين المنصات واسعة الاستخدام، تبرز VxWorks و Windows CE لتعدد استخداماتهما وقدراتهما في الوقت الفعلي. تقدم هذه الورقة تحليلًا مقارنةً بين VxWorks و Windows CE، مع التركيز على بنيتهما، آليات الجدولة، وملاءمتها للتطبيقات في الوقت الفعلي (الزمن الحقيقي). تدرس هذه الورقة الميزات الأساسية لكل نظام تشغيل، مقاييس الأداء في الوقت الفعلي (الزمن الحقيقي)، وتنفيذ التطبيقات الخاصة باستخدام بيانات من معايير موثوقة ودراسات حالة. يكشف التحليل أن VxWorks يتفوق في بيئات الوقت الفعلي الصارمة بسبب جدولته الحتمية، بينما يوفر Windows CE مرونة أكبر وقدرات تكامل لمهام الوقت الفعلي اللينة. توفر هذه الدراسة رؤى قيمة للمطورين والمهندسين لاختيار نظام التشغيل في الوقت الفعلي المناسب بناءً على متطلبات التطبيق المحددة.

## Introduction

The rapid advancement of technology has led to the proliferation of real-time systems in critical applications, ranging from aerospace and medical devices to industrial automation and consumer electronics. Real-time operating systems (RTOS) form the backbone of these systems by managing hardware resources and ensuring timely and deterministic execution of tasks. Among the many RTOS available today, VxWorks and Windows CE have garnered significant attention due to their proven capabilities in handling diverse real-time requirements.

VxWorks, developed by Wind River Systems, is widely recognized for its high-performance capabilities and robust support for hard real-time systems. It has been a preferred choice in mission-critical domains such as avionics, space exploration, and industrial control systems. On the other hand, Windows CE, a Microsoft product, offers a versatile platform designed for embedded applications, providing integration with the broader Windows ecosystem while

supporting soft real-time requirements in consumer electronics and industrial applications [1].

Despite their widespread use, there is a lack of comprehensive studies comparing these two RTOS across key performance and design parameters. This paper seeks to address this gap by analysing the architectures, scheduling mechanisms, and real-time capabilities of VxWorks and Windows CE. Additionally, the paper evaluates their suitability for various real-time applications, supported by examples from real-world implementations.

The remainder of this paper is structured as follows: Section 2 provides a background on RTOS and reviews related work. Section 3 delves into the technical features of VxWorks and Windows CE, followed by a comparative analysis in Section 4. Section 5 discusses challenges and limitations, while Section 6 concludes with insights and future directions.

## Background and Related Work

Background on Real-Time Operating Systems

Real-time systems are designed to perform specific tasks

\*Corresponding author

[https://doi.org/10.63318/waujpasv3i2\\_03](https://doi.org/10.63318/waujpasv3i2_03)

within strict timing constraints, making them essential in applications where delays could lead to system failure or significant performance degradation. Real-time operating systems (RTOS) are specialized software platforms that provide the necessary scheduling, resource management, and timing mechanisms to ensure deterministic behaviour.

Two primary categories of real-time systems are hard real-time and soft real-time systems. Hard real-time systems require tasks to be executed within strict deadlines, with failure being unacceptable (e.g., aerospace or medical devices). In contrast, soft real-time systems tolerate occasional deadline misses but strive to maintain acceptable performance (e.g., video streaming or consumer electronics).[1]

The field of embedded real-time systems has seen significant advancements over the years, with multiple real-time operating systems (RTOS) being developed to cater to diverse application requirements. Below is an overview of the most prominent RTOS in the industry, highlighting their features, use cases, and relevance to embedded real-time applications.

1. **FreeRTOS:** FreeRTOS is a widely adopted open-source RTOS designed for microcontrollers and small embedded systems. Its lightweight and easy-to-use architecture make it a popular choice for developers working on resource-constrained devices. FreeRTOS supports a wide range of hardware architectures and is often used in IoT devices and low-power applications. However, its simplicity may limit its suitability for more complex, high-reliability systems.[32]
2. **QNX:** QNX is a commercial RTOS known for its microkernel architecture, which enhances its performance and reliability. It is widely used in automotive and industrial applications, where real-time responsiveness and system stability are paramount. QNX's modular design allows for customization, but its licensing costs and complexity may limit its adoption in smaller-scale projects.[33]
3. **RTEMS (Real-Time Executive for Multiprocessor Systems):** RTEMS is an open-source RTOS designed for embedded systems, particularly those requiring multi-threading and real-time scheduling. Its support for multiprocessor systems makes it suitable for applications in aerospace and scientific research. While RTEMS offers a rich feature set, its learning curve and documentation challenges can be barriers for new users.[34]
4. **Micrium:** Micrium is a commercial RTOS tailored for resource-constrained devices. It is known for its high performance, modularity, and extensive library support. Micrium is often used in consumer electronics and medical devices, where efficiency and reliability are critical. However, its commercial licensing may not be feasible for all developers.[35]
5. **ThreadX:** ThreadX, developed by Express Logic (now part of Microsoft), is a compact and efficient RTOS designed for embedded systems. Its fast context switching and user-friendly API make it a popular choice for real-time applications. ThreadX is widely used in consumer electronics and industrial automation, but its proprietary nature may limit its accessibility.[36]
6. **Nucleus:** Nucleus, a scalable RTOS from Mentor Graphics, is designed for embedded systems requiring real-time performance and modularity. It offers comprehensive middleware support, making it suitable for

automotive and telecommunications applications. However, its commercial licensing and complexity may deter smaller projects.[37]

7. **ChibiOS/RT:** ChibiOS/RT is an open-source RTOS with a focus on embedded systems. Its compact size, high performance, and rich feature set make it a viable option for developers seeking a balance between functionality and resource usage. ChibiOS/RT is often used in robotics and automation, but its community support may not match that of larger projects like FreeRTOS.[38]
8. **Zephyr:** Zephyr is an open-source RTOS designed for IoT devices. Its modular and scalable architecture supports a wide range of hardware, making it a versatile choice for connected devices. Zephyr's growing community and active development make it a promising option for future IoT applications, though it may lack the maturity of more established RTOS.[39]

### Research Gap and Contribution

While numerous RTOS options are available, each with its own strengths and limitations, there is a lack of comprehensive comparative studies that evaluate their performance in embedded real-time applications. Existing research often focuses on individual RTOS or narrow use cases, leaving a gap in understanding how systems like VxWorks and Windows CE compare in terms of real-time performance, scalability, and reliability. This paper addresses this gap by providing a detailed comparative analysis of VxWorks and Windows CE, offering insights into their suitability for various embedded real-time applications.

### Overview of VxWorks and Windows CE

VxWorks, developed by Wind River Systems, is a highly reliable RTOS tailored for hard real-time applications. Its microkernel-based architecture, preemptive scheduling, and support for advanced real-time features make it ideal for mission-critical systems, including space exploration (e.g., Mars rovers) and industrial robotics. VxWorks has been a benchmark for RTOS performance due to its deterministic execution and scalability.[10]

Windows CE, introduced by Microsoft, caters to the embedded systems market, focusing on soft real-time and general-purpose applications. Its modular design, integration with the Windows ecosystem, and graphical capabilities make it a popular choice for devices such as handheld computers, automotive systems, and industrial equipment. Despite its primary focus on soft real-time tasks, Windows CE provides adequate support for applications with moderate timing requirements.[22]

### Related Work

Several studies have explored the characteristics and performance of RTOS in various domains. For instance, [1] conducted a survey of RTOS architectures, emphasizing the trade-offs between monolithic and microkernel designs. Their findings highlighted the superior scalability of microkernel-based systems, such as VxWorks, over monolithic systems.

In another study, [2] evaluated the scheduling algorithms used by popular RTOS, comparing rate-monotonic, earliest-deadline-first, and hybrid scheduling approaches. Their results underscored the significance of algorithm selection based on application-specific requirements.

Comparative analyses of VxWorks and Windows CE have primarily focused on isolated aspects such as power management [3] or user experience [4]. However, a holistic comparison covering architectural design, scheduling, and application suitability remains scarce. This paper aims to

bridge this gap by providing a comprehensive analysis of VxWorks and Windows CE, drawing insights from existing literature and real-world case studies.

### Technical Analysis

#### VxWorks Architecture

VxWorks is a microkernel-based RTOS designed for real-time, safety-critical applications. The microkernel architecture ensures minimal overhead by separating core system services (such as scheduling, interrupt handling, and inter-process communication) from application-specific components. This modular design enhances system scalability and reliability, as critical real-time tasks can be isolated from less time-sensitive operations.

1. **Kernel-Design:** VxWorks employs a pre-emptive, priority-based scheduler that ensures real-time tasks are given CPU time based on priority and timing constraints. The kernel also supports a priority inheritance protocol to avoid priority inversion, a common issue in real-time systems.[3,5]
2. **Real-Time-Scheduling:** VxWorks offers several real-time scheduling policies, including rate-monotonic and earliest-deadline-first (EDF) algorithms. These allow the system to meet hard real-time constraints by ensuring that critical tasks are executed within their deadlines. VxWorks' scheduler is highly deterministic, making it ideal for applications where timing is critical, such as aerospace and defense.[5,10]
3. **Inter-Process Communication (IPC):** VxWorks provides efficient IPC mechanisms such as message queues, semaphores, and shared memory, enabling communication between real-time tasks. These mechanisms are designed to minimize latency and ensure predictable behavior, a crucial requirement in real-time environments.[5]
4. **Interrupt Handling:** VxWorks supports interrupt latency minimization, a critical feature for hard real-time systems. The system allows direct handling of interrupts with low latency, which ensures that time-sensitive tasks are executed without delay. [5,10]

#### Windows CE Architecture

Windows CE is a modular, multitasking operating system designed primarily for embedded systems. Unlike VxWorks, Windows CE is not strictly a microkernel OS; it uses a hybrid kernel that integrates the advantages of both monolithic and microkernel designs. Its flexibility and extensibility make it a popular choice for applications requiring a balance between real-time and general-purpose computing.

1. **Kernel-Design:** The Windows CE kernel offers preemptive multitasking, allowing multiple applications to run concurrently while providing real-time scheduling

features for time-sensitive tasks. While the kernel's design allows for better integration with Windows-based systems, it may introduce non-deterministic behavior in certain scenarios, particularly in systems with heavy general-purpose workloads.[22]

2. **Real-Time-Scheduling:** Windows CE primarily supports time-slice scheduling, which divides CPU time into intervals for each running process. While this approach is suitable for many embedded applications, it is not inherently deterministic. To accommodate real-time tasks, developers must implement priority-based scheduling or use third-party real-time extensions. The real-time extension (RTE) in Windows CE allows for soft real-time capabilities but may not meet the stringent requirements of hard real-time applications.[24, 25]
3. **Inter-Process Communication (IPC):** Windows CE provides standard IPC mechanisms, including named pipes, mail slots, and event objects. These are designed to facilitate communication between processes in embedded systems, though they can introduce latency compared to the more streamlined IPC mechanisms in VxWorks.[23,24]
4. **Interrupt Handling:** Windows CE supports interrupt handling, but its interrupt latency is generally higher than VxWorks, which can impact time-sensitive operations. The system allows developers to define interrupt priorities, but the inherent complexity of its kernel architecture means that interrupt handling is not as deterministic as in VxWorks.[25,26,27]

### Key Comparison

Table 1 provides a structured comparison between the two systems, VxWorks and Windows CE, focusing on their architectural and performance characteristics for real-time applications. It highlights critical differences in kernel design, scheduling mechanisms, interprocess communication (IPC), interrupt handling, and real-time capabilities.

#### Suitability for Real-Time Applications

- VxWorks is well-suited for hard real-time applications where strict deadlines must be met, such as in aerospace, defense, and medical systems. Its low interrupt latency, deterministic scheduling, and high reliability make it ideal for critical systems that cannot tolerate failure or delay. [10].
- Windows CE, while capable of handling soft real-time requirements, is better suited for applications that require a balance between real-time and general-purpose functionality. It is widely used in consumer electronics, automotive systems, and industrial automation, where time constraints are important but not as stringent as those in hard real-time systems [23].

**Table 1:** Comparison between VxWorks and Windows CE features

Feature	VxWorks	Windows CE
Kernel Type	Microkernel	Hybrid kernel (monolithic and microkernel)
Scheduling	Preemptive, priority-based (hard real-time)	Time-slice, preemptive (soft real-time)
IPC Mechanisms	Message queues, semaphores, shared memory	Named pipes, mail slots, event objects
Interrupt Handling	Low-latency, high-priority interrupts	Higher latency, priority-defined interrupts
Real-Time Support	Hard real-time (deterministic)	Soft real-time (through RTE extensions)
Scalability	Highly scalable for critical applications	Scalable, but less efficient for hard real-time tasks
Flexibility	Optimized for dedicated, mission-critical systems	Flexible and suitable for embedded systems with moderate real-time requirements

## Case Studies and applications

### VxWorks in Aerospace and Defense

One of the most well-known applications of VxWorks is in the aerospace and defense industry. VxWorks is used in mission-critical systems where failure to meet deadlines can result in catastrophic consequences. Its deterministic behavior and high reliability make it an ideal choice for such applications.

1. NASA's Mars Rovers: VxWorks has been used extensively by NASA in the operation of the Mars rovers, including the Spirit, Opportunity, and Curiosity rovers. These rovers rely on VxWorks for their onboard operating systems to manage real-time communication with Earth, control robotic movements, and process scientific data. The real-time scheduling capabilities of VxWorks ensure that the rover's actions are performed without delay, even in the harsh and unpredictable environment of Mars. VxWorks' ability to meet hard real-time constraints is essential for the success of these missions.[5]
2. Military Systems: VxWorks is also a trusted platform for many military applications, including Unmanned Aerial Vehicles (UAVs) and missile guidance systems. These systems require the highest levels of reliability and precision, which VxWorks provides through its low-latency interrupt handling and preemptive scheduling. In these environments, any failure to meet timing constraints could result in mission failure, making VxWorks a preferred choice for these applications.[10]

### Windows CE in Consumer Electronics and Industrial Automation

While VxWorks dominates in hard real-time applications, Windows CE is widely used in consumer electronics and industrial automation, where the timing constraints are often softer and more flexible.

1. Automotive Systems: Windows CE is commonly used in in-car infotainment systems, where it manages multimedia and navigation features. For example, the Ford SYNC system, which allows drivers to interact with their vehicles using voice commands, utilizes Windows CE to handle both real-time requirements (such as processing voice commands) and general-purpose tasks (such as running media applications). The system's flexibility, integration with other Windows-based applications, and ease of use make it a strong contender in the automotive industry, where real-time performance is necessary but not as strict as in safety-critical systems.[16,24]
2. Industrial Control Systems: In industrial automation, Windows CE is employed in Programmable Logic Controllers (PLCs) and Human-Machine Interfaces (HMIs). These systems benefit from Windows CE's ability to support soft real-time capabilities, allowing them to process data from industrial sensors and control machinery while also providing an interface for operators. For instance, in manufacturing plants, Windows CE-based systems are used to control assembly lines, process data, and provide real-time feedback to operators, all while ensuring a balance between real-time processing and user interaction.[24]
3. Medical Devices: Windows CE has also found a place in medical devices, such as patient monitoring systems and diagnostic equipment. These devices often require real-time data processing to monitor patient vitals, such as heart rate or blood pressure, and to trigger alarms if abnormal conditions are detected. Windows CE provides the necessary flexibility to integrate with other medical systems and software, while also supporting soft real-time

capabilities to meet the performance demands of these applications [26].

### Comparison of Use Cases:

Table 2 provides a side-by-side comparison of VxWorks and Windows CE across critical industries, highlighting their suitability for real-time embedded applications based on performance, reliability, and use-case requirements. By examining their adoption in sectors like aerospace, military, automotive, industrial automation, and medical devices, this analysis reveals how each operating system's design philosophy aligns with domain-specific demands.

Table 2: Comparison between VxWorks and Windows CE in Industry applications.

Industry/Application	VxWorks	Windows CE
Aerospace (Mars Rovers)	Critical real-time control and communication, requiring hard real-time capabilities.	Not suitable due to the need for hard real-time guarantees.
Military Systems (UAVs)	High reliability and low latency for mission-critical applications.	Typically, not used due to lack of deterministic scheduling.
Automotive (Ford SYNC)	N/A	Handles soft real-time tasks like multimedia and voice commands while supporting general-purpose features.
Industrial Automation (PLC)	N/A	Manages control systems and real-time data processing with flexibility and ease of integration.
Medical Devices (Patient Monitoring)	N/A	Soft real-time processing of patient data with seamless integration into hospital networks.

### Conclusion of Case Studies

The case studies demonstrate the strengths of both VxWorks and Windows CE in real-world applications. VxWorks is clearly the choice for hard real-time applications where meeting strict deadlines is non-negotiable, such as in aerospace and defense systems. On the other hand, Windows CE shines in environments where soft real-time requirements are sufficient, such as in automotive systems, industrial control, and medical devices. Each system excels in its respective domain, and the choice between them largely depends on the specific real-time demands of the application at hand.

### Challenges and limitations

#### Challenges with VxWorks

While VxWorks is a highly robust RTOS for hard real-time applications, it is not without its challenges and limitations.

1. Complexity in System Configuration and Integration: The modular nature of VxWorks offers great flexibility, but this can make system configuration and integration more complex. Developers often need to carefully select and configure the appropriate components, such as the kernel, device drivers, and real-time features, to ensure optimal performance. This process can be time-consuming and may require deep expertise in the platform.



2. **High Cost of Licensing:** VxWorks is a commercial RTOS, and its licensing costs can be quite high, particularly for small to medium-sized projects. This can be a significant barrier for organizations with limited budgets, especially when compared to open-source or less expensive embedded operating systems. The cost structure may also limit the adoption of VxWorks in industries that do not require hard real-time guarantees.
3. **Limited Support for General-Purpose Applications:** While VxWorks excels in hard real-time environments, it lacks the same level of support for general-purpose applications as other more widely used operating systems. It is not as well-suited for applications that require heavy multitasking, complex graphical user interfaces (GUIs), or integration with a wide range of third-party software libraries and frameworks.
4. **Longer Development Time:** Due to its specialized nature and focus on real-time performance, developing and debugging applications on VxWorks can take longer compared to more common platforms like Windows or Linux. The need for custom configuration and optimization can lead to increased development cycles, which can be a limitation in time-sensitive projects.

### Challenges with Windows CE

Windows CE, while versatile and widely used, has its own set of limitations, especially when dealing with more stringent real-time requirements.

1. **Lack of True Deterministic Real-Time Support:** While Windows CE does support real-time extensions (RTE), it does not provide the same level of deterministic, hard real-time support that VxWorks offers. The kernel's hybrid design, while efficient for general-purpose tasks, introduces unpredictability in the execution of time-sensitive tasks. For systems where absolute deadline guarantees are critical, such as in aerospace or medical applications, Windows CE may not be suitable.
2. **Higher Interrupt Latency:** Windows CE is known for higher interrupt latency compared to VxWorks. In time-critical applications, such as those in industrial control or automotive safety, even slight delays in interrupt handling can lead to unacceptable performance degradation or failure to meet timing constraints. The higher interrupt latency can significantly impact the real-time performance of applications that require immediate responses to external events.
3. **Limited Support for Multi-Core Processing:** Although Windows CE supports multi-core processors, it does not fully optimize parallel processing in the way other modern operating systems like Windows or Linux do. This can limit the scalability and performance of applications requiring extensive parallel computation, such as in advanced robotics or high-performance computing in embedded systems.
4. **Deprecation and Limited Updates:** Windows CE has been largely phased out in favor of more modern operating systems like Windows Embedded Compact and Windows 10 IoT Core. As a result, developers working with Windows CE may face challenges related to limited support and fewer updates. Many newer hardware devices and technologies are not fully compatible with Windows CE, leading to potential integration challenges for new projects.
5. **Security-Concerns:** As an embedded OS designed for relatively less critical applications, Windows CE does not provide the same level of security features found in full-

fledged operating systems. Security is often a secondary concern in the design of Windows CE, which may pose risks for applications in fields where data integrity and protection are paramount, such as in healthcare or financial systems.

### Comparison of Challenges

Table 3 compares VxWorks and Windows CE across critical embedded system challenges. VxWorks offers superior hard real-time performance with low interrupt latency and strong security, but requires complex configuration and carries high licensing costs. Windows CE provides easier integration and lower costs, but lacks deterministic real-time capabilities and has higher interrupt latency. While VxWorks excels in mission-critical applications, Windows CE suits general-purpose embedded systems with softer timing requirements. Both face limitations in multi-core support, though VxWorks remains the choice for safety-critical environments.

**Table 3:** Challenges in VxWorks and Windows CE.

Challenge	VxWorks	Windows CE
System Configuration	Complex system configuration required due to modular architecture.	Easier integration but lacks real-time determinism.
Licensing Cost	High licensing costs.	Relatively low cost but limited real-time capabilities.
Real-Time Support	Excellent for hard real-time but less suited for soft real-time.	Soft real-time capabilities, but lacks hard real-time determinism.
Development Time	Longer development cycles due to complexity.	Faster development for general-purpose applications.
Interrupt Latency	Low interrupt latency (ideal for hard real-time).	Higher interrupt latency can affect real-time performance.
Security	Strong security features for critical systems.	Limited security features, not ideal for sensitive data applications.
Multi-Core Support	Limited multi-core processing.	Limited support and optimization for multi-core processors.

Both VxWorks and Windows CE have unique strengths and limitations, which make them suitable for different application domains. VxWorks is ideal for hard real-time systems in industries such as aerospace and defense, but its complexity and high licensing cost may limit its adoption in smaller or budget-constrained projects. On the other hand, Windows CE offers flexibility and cost-effectiveness for embedded systems with less stringent real-time requirements, but its lack of true deterministic real-time support and higher interrupt latency make it unsuitable for mission-critical applications.

Understanding these challenges is key to selecting the right operating system for a given application, and this paper has highlighted the trade-offs developers must consider when choosing between VxWorks and Windows CE for real-time systems.

### Conclusion

In this paper, we have provided a detailed comparison between VxWorks and Windows CE, two widely used operating systems in the realm of real-time applications. Both systems

are designed to meet the demands of time-sensitive environments, but their strengths and limitations make them suitable for different use cases.

VxWorks excels in hard real-time applications, offering deterministic scheduling, low interrupt latency, and high reliability. Its microkernel architecture and robust support for critical systems have made it the preferred choice for industries such as aerospace, defense, and medical devices, where meeting strict timing constraints is paramount. However, its complex configuration, high licensing costs, and longer development times can pose challenges, particularly for smaller projects or those requiring general-purpose functionality.

On the other hand, Windows CE provides a more flexible and cost-effective platform, ideal for soft real-time applications that demand a balance between real-time processing and general-purpose computing. It is commonly used in automotive systems, consumer electronics, and industrial automation, where timing constraints are important but not as stringent. Despite its advantages in these domains, Windows CE lacks true deterministic real-time support and suffers from higher interrupt latency and limited multi-core optimization, which make it less suitable for hard real-time tasks.

Ultimately, the choice between VxWorks and Windows CE depends on the specific requirements of the application. For mission-critical systems where meeting deadlines is non-negotiable, VxWorks remains the top choice. However, for applications that require a more versatile platform with moderate real-time requirements, Windows CE offers significant benefits in terms of cost, development speed, and integration with Windows ecosystems.

Future research could explore hybrid approaches that combine the strengths of both operating systems, allowing developers to tailor solutions for even more diverse and complex real-time applications. By understanding the unique features and trade-offs of these operating systems, engineers and system designers can make more informed decisions and optimize the performance of their real-time systems.

**Author Contributions:** "It's a single-author article."

**Funding:** "This research received no external funding."

**Data Availability Statement:** "The data are available at request."

**Conflicts of Interest:** "The authors declare no conflict of interest."

## References

- [1] K. Pothuganti, A. Haile, and S. Pothuganti, "A comparative study of real-time operating systems for embedded systems," *Int. J. Innov. Res. Comput. Commun. Eng.*, vol. 4, no. 6, Jun. 2016. Available Online : 10.15680/IJIRCCE.2016.0406224.
- [2] A. Barbalace; A. Luchetta; G. Manduchi, "Performance comparison of VxWorks, Linux, RTAI, and Xenomai in a hard real-time application," *IEEE Trans. Nucl. Sci.*, vol. 55, no. 1, pp. 435-439, 2008. <https://doi.org/10.1109/TNS.2007.905231>
- [3] J. Yu, "Design and Implementation of VxWorks Compatibility Library Based on Xenomai," in *Proc. 6th Int. Conf. Communications, Information System and Computer Engineering (CISCE)*, Guangzhou, China, May 10-12, 2024. <https://doi.org/10.1109/CISCE62493.2024.10653438>.
- [4] L. Tan and J. Song, "Microkernel vs. Monolithic Kernel: Case Studies of VxWorks and Windows CE," *ACM Trans. Embed. Comput. Syst.*, vol. 17, no. 3, pp. 1-25, 2018. <https://doi.org/10.1145/3273905>
- [5] L. Shirui, G. Haifeng, and Q. Yalei, "Application of VxWorks interrupt affinity based on MPIC," in *Proc. 2nd IEEE Int. Conf. Computer and Communications (ICCC)*, Chengdu, China, Oct. 14-17, 2016. <https://doi.org/10.1109/CompComm.2016.7925074>.
- [6] M. Rajesh and B. Sreevidya, "Vulnerability analysis of real-time operating systems for wireless sensor networks," in *Proc. 3rd Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, 2020, pp. 449-460, [https://doi.org/10.1007/978-981-15-1483-8\\_38](https://doi.org/10.1007/978-981-15-1483-8_38)
- [7] G. Cao, K. Song, and J. Yang, "Adaptive low power design based on VxWorks kernel scheduler and hook mechanism," 2nd IITA Int. Conf. Geoscience and Remote Sensing (IITA-GRS), Qingdao, China, Aug. 28-31, 2010. <https://doi.org/10.1109/IITA-GRS.2010.5603226>.
- [8] W. Ruan and Z. Zhai, "Kernel-level design to support partitioning and hierarchical real-time scheduling of ARINC 653 for VxWorks," in *Proc. IEEE 12th Int. Conf. Dependable, Autonomic and Secure Computing (DASC)*, Dalian, China, Aug. 24-27, 2014. <https://doi.org/10.1109/DASC.2014.76>.
- [9] O. Hahm and E. Baccelli and H. Petersen and N. Tsiftes, "Operating systems for low-end devices in the Internet of Things: A survey," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 720-734, Oct. 2016. <https://doi.org/10.1109/IIOT.2015.2505901>.
- [10] G. Wilson, "The Decline of Windows CE and Rise of VxWorks in Modern Embedded Systems," *Embed. Comput. Des.*, 2023. [Online]. Available: <https://www.embedded.com/windows-ce-vs-vxworks-2023>
- [11] L.-P. Chen and Z. Xu, "The design and implementation of on-board data handling system based on VxWorks," *Wireless Commun. Netw.*, 2016. [https://doi.org/10.1142/9789814733663\\_0054](https://doi.org/10.1142/9789814733663_0054).
- [12] K. Soundararajan and R. W. Brennan, "Design patterns for real-time distributed control system benchmarking," *Robot. Comput.-Integr. Manuf.*, vol. 24, no. 5, pp. 606-615, Oct. 2008. <https://doi.org/10.1016/j.rcim.2007.10.002>
- [13] Y. Tao and K. Song, "Design of VxWorks-based software architecture for space optical remote sensor," in *Proc. IEEE Int. Conf. Electron. Mech. Eng. Inf. Technol. (EMEIT)*, vol. 4, 2011, pp. 1931-1934. <https://doi.org/10.1109/EMEIT.2011.6023222>
- [14] L. Zoltan, "Memory allocation in VxWorks 6.0," *Wind River Syst.*, Alameda, CA, USA, Tech. Rep., pp. 2-3, 2005. Available: <https://citeseerx.ist.psu.edu>
- [15] H. Zhou, W. Zhang, Y. Zhang, R. Ding, and F. Ba, "Intelligent SpaceWire bus controller driver design and implementation based on VxWorks," in *Proc. Int. Conf. Multimedia Technology (ICMT)*, Hangzhou, China, Jul. 26-28, 2011. <https://doi.org/10.1109/ICMT.2011.6002064>.
- [16] A.J. Kornecki and J. Zalewski and D. Eyassu, "Learning real-time programming concepts through VxWorks lab experiments," in *Proc. IEEE Conf. Softw. Eng. Educ. (CSEE)*, Austin, TX, USA, 2000. <https://doi.org/10.1109/CSEE.2000.827056>.
- [17] JY. Gao and D. Li, "Implementation of C Program Invocation in PLC Based on VxWorks Operating System," in *Proc. 11th Int. Forum Electrical Engineering and Automation (IFEAA)*, Shenzhen, China, Nov. 22-24, 2024, <https://doi.org/10.1109/IFEAA64237.2024.10878540>.
- [18] K. K. G. Buquerin, "Security evaluation for the real-time operating system VxWorks 7 for avionic systems," Bachelor thesis, Fac. Elect. Eng. Comput. Sci., Tech. Hochsch. Ingolstadt, Ingolstadt, Germany, 2018. [Online]. Available: <https://kmyr.de>
- [19] R. Li, "Computer embedded automatic test system based on VxWorks," *Int. J. Embed. Syst.*, vol. 14, no. 3, pp. 183-192,

- Aug. 2022. <https://doi.org/10.1504/IJES.2022.124839>.
- [20] Y. Yi, Q. Liu, L. Zhao, B. Wang, and Z. Zhang, "Design of the master control system of the array antenna based on the VxWorks," in *Proc. 4th Int. Conf. Intelligent Computation Technology and Automation (ICICTA)*, Shenzhen, China, Mar. 28-29, 2011, vol. 1, pp. 414-417, <https://doi.org/10.1109/ICICTA.2011.414>.
- [21] W. Lee, S. C. Kim, D.-K. Woo, Y.-S. Ma, and P. Mah, "Performance comparison of MRTOS and VxWorks in MultiBench benchmark suite," in *Proc. 19th Int. Conf. Adv. Commun. Technol. (ICACT)*, PyeongChang, Korea (South), 2017. <https://doi.org/10.23919/ICACT.2017.7890218>
- [22] S. Yang, L. Wang, S. Zhang, and J. Liu, "A Method on Extracting Registry Information from Windows CE Memory Images," in *Proc. IEEE Int. Conf. Computer Science and Applications (CSA)*, Wuhan, China, Dec. 14-15, 2013, pp. 1-5, <https://doi.org/10.1109/CSA.2013.175>.
- [23] S. Yang, L. Wang, and S. Zhang, "Exploratory study on memory analysis of Windows CE device," in *Proc. IEEE Int. Conf. Intelligent Computation Technology and Automation (ICICIP)*, Jun. 9-11, 2013. <https://doi.org/10.1109/ICICIP.2013.6568120>
- [24] J. Min, "A Design of Embedded Terminal Unit Based on ARM and Windows CE," in *Proc. 8th Int. Conf. Electronic Measurement and Instruments (ICEMI)*, Aug. 16-18, 2007. <https://doi.org/10.1109/ICEMI.2007.4350687>.
- [25] S. Xiao, D. Li, Y. Lai, J. Wan, and S. Feng, "An Open Architecture Numerical Control System Based on Windows CE," in *Proc. IEEE Int. Conf. Control and Automation (ICCA)*, Guangzhou, China, May 30-Jun. 1, 2007. <https://doi.org/10.1109/ICCA.2007.4376558>.
- [26] G. Qinlong, C. Xingmei, T. Weiwei, and Y. Minghai, "Study and application of SQLite embedded database system based on Windows CE," in *Proc. 2nd Int. Conf. Information Science and Engineering (ICISE)*, Hangzhou, China, Dec. 4-6, 2010 <https://doi.org/10.1109/ICISE.2010.5691551>.
- [27] H. Wang, Z. Wang, and H. Lin, "A universal IO controller based on Windows CE and C#," in *Proc. Int. Conf. Computer Application and System Modeling (ICCASM)*, Taiyuan, China, Oct. 22-24, 2010. <https://doi.org/10.1109/ICCASM.2010.5619286>.
- [28] X. Dong, L. Jianqun, and W. Jirong, "Research on real-time control of embedded NC system based on Windows CE 5.0," in *Proc. Int. Conf. Mechanic Automation and Control Engineering (MACE)*, Wuhan, China, Jun. 26-28, 2010. [HTTPS://DOI.ORG/10.1109/MACE.2010.5536252](https://doi.org/10.1109/MACE.2010.5536252).
- [29] Y. Zhang, D. Xue, C. Wu, and P. Ji, "Research of Portable Information Terminal Based on MIPS Processor and Windows CE," in *Proc. Int. Conf. Advanced Computer Control (ICACC)*, Singapore, Jan. 22-24, 2009. <https://doi.org/10.1109/ICACC.2009.97>.
- [30] J. Zhou and J. Yu, "Multithread serial communication model based on Windows CE," in *Proc. Int. Conf. Electric Information and Control Engineering (ICEICE)*, Wuhan, China, Apr. 15-17, 2011. <https://doi.org/10.1109/ICEICE.2011.5777858>.
- [31] L. Wu, W. Zhang, Y. Li, and F. Zhao, "Design and implementation of FPGA data communication interface driver based on Windows CE," in *Proc. Int. Conf. Advanced Computer Theory and Engineering (ICACTE)*, Chengdu, China, Aug. 20-22, 2010. <https://doi.org/10.1109/ICACTE.2010.5579864>.
- [32] R. Barry, "FreeRTOS: A Portable, Open Source, Mini Real-Time Kernel," FreeRTOS.org, 2020 <https://www.freertos.org/>
- [33] QNX Software Systems, "QNX Neutrino RTOS: A Foundation for Performance, Reliability, and Scalability," BlackBerry Limited, 2022. <https://www.qnx.com/products/neutrino-rtos/>.
- [34] RTEMS Project, "RTEMS: Real-Time Executive for Multiprocessor Systems," RTEMS.org, 2021. <https://www.rtems.org/>.
- [35] Silicon Labs, "Micrium OS: A High-Performance RTOS for Embedded Systems," Silicon Labs, Inc., 2020. <https://www.silabs.com/developers/micrium-os>.
- [36] Microsoft, "ThreadX: A Real-Time Operating System for Embedded Applications," Microsoft Corporation, 2023. <https://learn.microsoft.com/en-us/azure/rtos/threadx/>.
- [37] Mentor Graphics, "Nucleus RTOS: A Scalable Real-Time Operating System," Siemens Digital Industries Software, 2021. <https://www.mentor.com/embedded-software/nucleus/>
- [38] G. Di Marzo, "ChibiOS/RT: A Compact and Efficient Real-Time Operating System," ChibiOS.org, 2022. <http://www.chibios.org>
- [39] Zephyr Project, "Zephyr RTOS: A Scalable Real-Time Operating System for IoT Devices," The Linux Foundation, 2023. <https://www.zephyrproject.org/>