Secure Operating System- IE2032

Implementation of Embedded Real-Time Operating System and Application Software based on Smart Chip

Submitted Date- 29/09/2024

Campus-Malabe

Group Member Details

Name with initials	Registration Number
Amantha M.A	IT23184312
Perera P.A.J.M	IT23169708
Rajasooriya D.G.C.H	IT23170520
M.R.A.Peramunugama	IT23187832

Terms of Reference

A report submitted in fulfilment of the requirement for the module IE2032. Second Year First semester, Cyber Security, Faculty of Computing, Sri Lanka Institute of Information Technology.

This report provides a detailed analysis of RTOS architecture, smart chip design, and the co-development of hardware and software. The analysis will focus on the performance, challenges, and applications of these systems in modern technology.

Acknowledgement

The successful completion of this project is a result of the collective effort and support of many individuals whom we wish to acknowledge. Foremost, we extend our heartfelt gratitude to Ms Suranjini Silva whose unwavering guidance and encouragement have been invaluable throughout this work. Next, we would like to show our gratitude to the resources which helped us to create this report. We sincerely thank the hardworking team members whose dedication, knowledge and teamwork were essential in finishing this report. Each member's unique contributions have improved and broadened the scope of our research.

Table of Contents

Table of Figures

Abstract

The design and implementation of an embedded real-time operating system (RTOS) based on smart chip technology are covered in the research paper. Its main goal is to provide an improved RTOS framework that improves system efficiency and functionality by leveraging computer network technology. The micro-kernel architecture is used, relocating certain operations to user space and maintaining critical OS functions within the kernel. The features and implementation of the integrated real-time operating system μCOS-II, together with its interaction with the NIOS II soft core processor, receive particular emphasis.

The study places a strong emphasis on co-designing hardware and software to guarantee resource allocation that is balanced and to enhance productivity and job management. It also emphasizes how crucial real-time functionality is for embedded systems to fulfill stringent timing and performance standards. The system is demonstrated to prolong the CPU's service life by experimental analysis, and the suggested network connection protocol enhances data handling. The benefits and future prospects of embedded RTOS in smart chip applications are examined in the report's conclusion.

1.Introduction

Embedded Real-Time Operating Systems (RTOS)

An RTOS is a specialized type of operating system that is developed to manage resources of embedded devices and perform tasks with hard time constraints. [1] Unlike general-purpose OSes, which emphasize multitasking and flexibility, RTOS focuses on real-time performance, guaranteeing critical tasks' deadlines. In other words, timing errors in applications like aerospace, automotive, and industrial control systems can be disastrous.

With the rapid increase of intelligent devices and the IoT, RTOS has gained much significance. Many such intelligent devices are based on embedded systems, which have very limited resources, and RTOS make efficient use of these limited resources while maintaining high performance. In the case of autonomous vehicles, for instance, software must be able to integrate seamlessly with sensors, control systems, and other elements in real time.

Real-Time Performance and Resource Management

RTOS is designed to efficiently manage the scarce resources of an embedded system, such as memory, processing power, and energy. It does this by:

• Scheduling Tasks: normally done by prioritizing tasks by their importance and time to complete.

• Interrupt Management: it deals with interrupts from hardware devices efficiently.

• Resource Allocation: The system apportions resources like memory and CPU time according to task requirements

Key Industries and Applications

Some of the broad industries in which RTOS find applications include the following.

• **IoT**: In IoT, RTOS enables devices to collect data and process it in real time with no delay, thus facilitating autonomous operations, intelligent decision-making, and more.

• **Automotive**: The primary need for an RTOS is found in the implementation of ADAS, where timely response is crucial to avoid potential hazards. Examples include collision avoidance or notifications in case of lane departure.

• **Aerospace**: Avionic systems controlling aircraft flight, navigation, and communication depend on RTOS for dependability and safety in these life-critical applications.

Figure 1:Avionic system

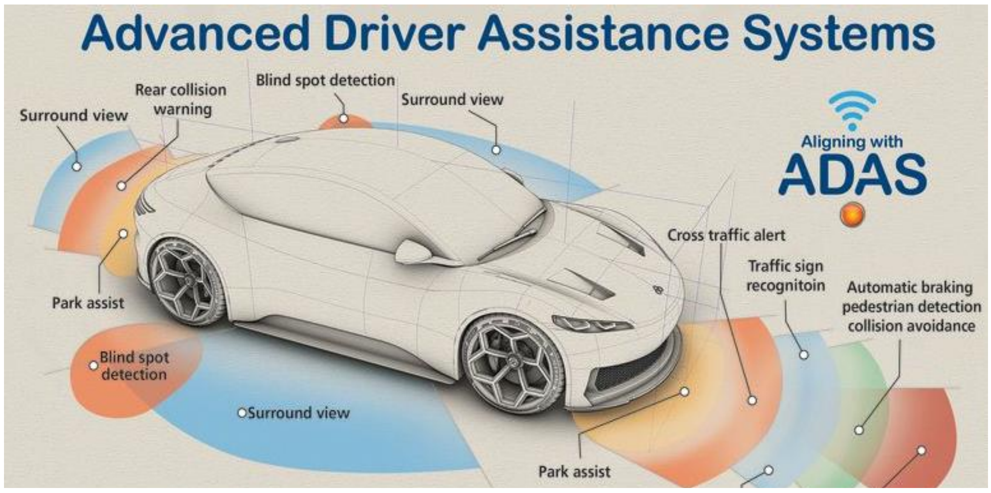

Figure 2:ADAS

Micro-Kernel Architectures

There is one very typical design technique for RTOS, namely the Micro-kernel architecture. It separates the core operating system functions, such as task scheduling and memory management, from other services like file systems and device drivers. [2] The advantages of this modular type of design are many and include:

• Flexibility: Non-essential services are easier to modify than change.

• Efficiency: Reduced overhead improves real-time performance.

• Reliability: Separation of critical from non-critical improves the system stability.

2.Problem and solution

In this research paper there are three methodologies proposed. They are Smart Chip Embedded Design, Embedded Real-Time Operating System and Embedded Real-Time System and Application Software Implementation. In each methodology mention identified problems and author proposed solution for those problems.

Smart Chip Embedded Design

Complexity in Hardware and Software Integration

An embedded system is a combination of hardware, software and other mechanical parts designed to perform a specific function. It contains processor and software. [3] In traditional embedded system design software and hardware were designed separately. This method leads to coordination errors, high design cost and long design cycles, design errors and inefficiencies. To solve these problems, the suggested solution is hardware and software co-design. Co-design is the concept that concurrent designs of hardware and software components of complex electronic systems. [4] This approach balances the resources allocated between hardware (FPGA- Field Programmable Gate Array that consists of internal hardware blocks, capable of parallel operations) and software, reducing errors and manufacturing cost and improving system efficiency and performance. [5]

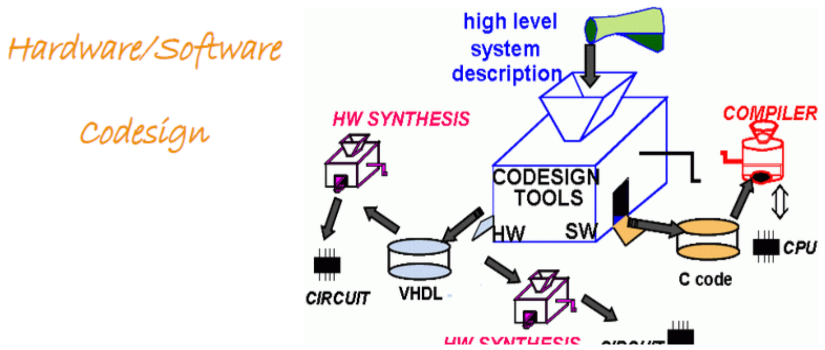

Figure 3:Hardware and Software co-design

Embedded Real-Time Operating System

Resource Constraint in Embedded System

In generally, embedded real-time operating systems have limited resources such as memory and processing power. These operating system's main goal is performing tasks on time. [6] To complete tasks successfully system have to enrich high performances like achieve high reliability and responsiveness and manage multiple tasks effectively. For that author suggests the micro-kernel architecture. The microkernel runs the minimum necessary functions. [7] This means it has light core with reduced complexity and has to consider few dependencies when adding functionality. This makes extending the OS easier because all new services are added to the user space without the modifying the kernel. This allows more efficient use of limited resources.

Embedded Real-Time System and Application Software Implementation

Overhead from Network Communication Protocols

The problem with using simple function calls for sending and receiving data in an embedded RTOS is that it can make the system unpredictable. When messages arrive at unexpected time, it causes delays and make harder to meet strict timing constraints. This affect to slow down the system, block important tasks and increase overall processing overhead leading to inefficient use of resources. The author suggests to optimize embedded network communication protocol by using a modular approach for solve this problem.

Task Scheduling and Memory Management

Managing multiple tasks simultaneously while meeting strict time constraints is a challenge. Poor task scheduling can lead to miss deadlines which critical for real-time applications. Author suggests to use µC/OS-II kernel for task scheduling which support multitasking. The RTOS schedules tasks allowing high priority to run first.

Memory management algorithms lead to memory fragmentation over time. Memory fragmentation reduce system efficiency and cause unpredictable performance making difficult to meet time constraint of RTOS. The author suggests to use dynamic memory allocation method supported by µC/OS-II such as malloc and free. This method reduces memory fragmentation by freeing up memory after use.

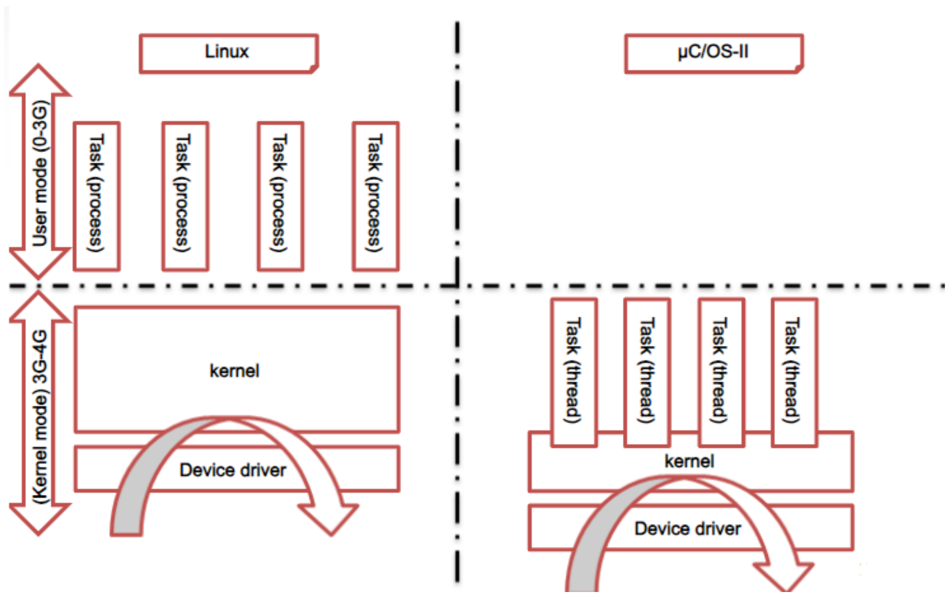

Figure 4:µC/OS-II

Task Communication and Synchronization

In embedded system there need to be a good communication and synchronization between tasks. Poor task synchronization cause to deadlocks, race conditions or delays which reduce the performance and cause system instability. The proposed solution is using semaphores, message queues and mailboxes. These mechanisms help to communicate and share data without causing conflicts.

3.Results and Implications

1. Successful Outcomes of the System’s Implementation

- Improved CPU performance: µCOS-II real-time operating system allowed better multitasking with lower CPU usage.
- Micro-kernel architecture: Key OS services were kept in the kernel, while less critical ones moved to user space, improving efficiency.
- Optimized network communication: Reducing unnecessary protocol layers minimized system overhead.
- Faster response times: Optimized the communication process, making real-time data transfer faster.
- Reduced network latency: Streamlined communication between components for quicker data exchanges.
- Maximized data throughput: More data could flow through the network without causing delays.

2. Extending CPU Life and Improving System Reliability

The research paper discusses about extending CPU life and increasing system reliability.
The system's efficient task scheduling and multitasking mechanism was designed with the goal of enhancing CPU life and reliability. Tasks were able to share CPU resources without overloading the processor thanks to the µCOS-II real-time kernel, which decreased wear and increased the processor's operating lifespan. Furthermore, the embedded real-time operating system of the system minimized the necessity for frequent interruptions and enhanced memory management, thereby lowering the total computational load. Better heat management and longer hardware lifespan were made possible by these advancements, which made the system more dependable and durable for embedded applications. [8]

3. Contribution to the Field of Real-Time Operating Systems in Embedded Environments

- Optimized task management -increased the effectiveness of embedded systems.
- Resource allocation that was balanced: Memory and CPU were used more wisely.
- Customized for certain applications: The system might be adjusted for use with industrial control systems and Internet of Things devices.
- Energy efficiency: Better power usage in real-time systems was made possible by the flexible architecture.
- Upgraded smart chip technology -contributed to the enhancement of embedded software's hardware integration for improved performance.
- Resource-constrained environments: Demonstrated how real-time performance might be achieved by systems with limited resources.

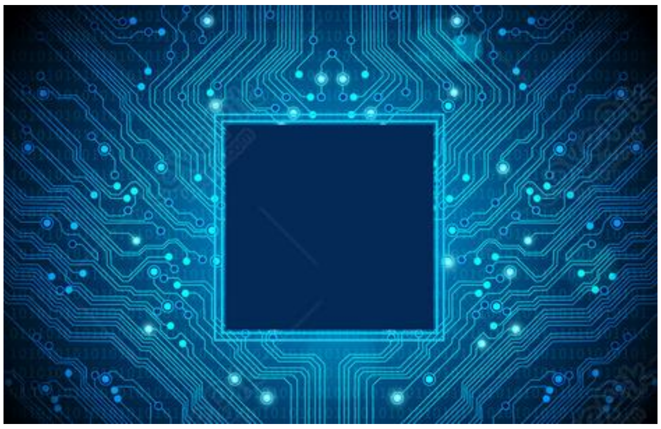

Figure 5:Smart chip

4. Implications for Future Embedded System Designs

- Growing demand for real-time systems: Effective real-time processing will become more and more necessary as devices get more sophisticated.
- Devices that are more compact and sophisticated: The system's architecture encourages the trend of technology becoming smaller.
- Hardware and software co-design: This co-design strategy can be used in future systems to increase functionality.
- Future systems will continue to prioritize key characteristics like task management and memory optimization.
- Applications for medical gadgets and driverless cars: The system's developments have applications in vital domains such as healthcare and transportation.
- Infrastructure for smart cities: In order to manage future smart cities and increase performance and efficiency, real-time systems will be essential.

Figure 6:Driverless car

4.Critical Evaluation and Relevance to Modern OS

Critical evaluating of Micro kernel Architecture and tailored TCP/IP protocol in Modern Operating Systems.

Strengths

1. Modularity and Flexibility

- ❖ One of the primary strengths of the micro kernel architecture is its modular design. [9] By running minimal services in the kernel and shifting others (such as file systems and device drivers to user space the micro kernel improves system stability and security.

2. Security

- ❖ Due to the separation of critical services from the kernel micro kernels inherently provide a more secure operating environment. Fewer services in the kernel reduce the attack surface compared monolithic kernels, which is particularly beneficial for modern systems where security is a top concern

3.TCP/IP Tailoring

- ❖ A tailored TCP/IP protocol within the micro kernel environment allows for more efficient handling of network traffic by streamlining the protocol stack, unnecessary overhead can be avoided, leading to reduced latency and better performance. This is especially valuable for RTOS and IoT systems where time sensitive communication is crucial. [10]

Weaknesses

1. Performance Overhead

- ❖ Despite its advantages micro kernels can suffer from performance overhead due to increased communication between user space and kernel space, also known as message passing. [11]

2.Scalability

- ❖ Micro – Kernels may face scalability issues when applied to large scale system with many services.

3.Lack of Benchmarking

- ❖ One of the significant criticisms is the absence of robust benchmarking data without comprehensive comparison between micro kernels and other architectures.

Areas For Further Research

1.Securuty Enhancements

- ❖ Given the evolving cyber threats, further research into enhancing the security mechanisms of Micro – Kernel is crucial. Exploring the integration of micro kernel with modern cryptographic protocols and secure enclaves may also prove valuable.

2.Task Scheduling

- ❖ Task scheduling is another area that requires attention, especially in RTOS and multicore environments. Existing scheduling algorithms need to be optimized for the micro kernel's architecture to reduce context switching overhead and improve overall system responsiveness.

3.Scalability solutions

- ❖ While scalability remains a challenge for future research could focus on distributed micro kernels or hybrid architectures that combine the modularity of micro kernels with the performance benefits of monolithic designs.

5.Relate to Broader Concept

Understand Operating Systems

This paper contributes to a more comprehensive understanding of how operating systems can be optimized for specialized real-time environments, particularly where timing, reliability, and resource constraints are critical factors.

Relevance to Modern Operating Systems

1.IoT Systems

- ❖ The micro–kernel architecture is highly relevant to IoT devices due to its minimalistic and secure design Many IoT devices are resource constrained and require secure communication with a lightweight footprint which micro kernels provide.

2.Multi–core Systems

- ❖ In the context of multi core processes the relevance of micro kernels is mixed Whie their modular design fits well with parallel processing, The IPC overhead and frequent context switching may hinder their scalability.

3.Real Time Applications

- ❖ Micro kernels are also relevant for real time systems, especially in industrial automation and critical systems like avionics. The deterministic behavior of micro kernels where only essential services run in kernel mode helps in maintaining predictability a crucial requirement in Real Time Applications.

Application to Future Operating Systems

1. Modular IoT Operating Systems

- The concept of micro – kernel can apply to develop future IoT operating systems that are both lightweight and secure. The flexibility offered by the architecture allows developer to tailor the OS to specific IoT devices. [12]

2. Real Time Operating Systems (RTOS)

- Micro kernel principles can be integrated into RTOS development particularly for applications that require high security deterministic behavior. By optimizing scheduling algorithms and minimizing IPC costs.

3.Next- Generation Multi–core OS:

- For future multi–core systems further research is needed to overcome the scalability issues in micro kernels. A hybrid approach that combines the micro–kernel's modularity with optimized parallel processing algorithms could be for the basis of next generation operating systems design for cloud computing.

6.Conclusion

The research paper on Embedded RTOS based on smart chip technology helps us to understand the concept of enhancing system performance in a resource-constraint environment. Such challenges are met by adopting a micro-kernel architecture and co-designing the hardware and software such as task scheduling, memory management, and communication protocol overhead. The solutions proposed enhance the real-time task execution and system reliability, which makes this research extremely important for modern applications in the domains of automotive, industrial and medical systems. The paper is useful in practice but points to issues that should be addressed next, related to scalability and dynamic workloads.

7. References

[1] [Online]. Available: https://www.techtarget.com/searchdatacenter/definition/real-time-operating-system.

[2] [Online]. Available: https://www.oreilly.com/library/view/software-architecture-patterns/9781098134280/ch04.html.

[3] [Online]. Available: https://www.geeksforgeeks.org/what-is-smart-chip.

[4] [Online]. Available: 2. https://digilent.com/blog/what-is-an-fpga/?srsltid=AfmBOopOwXsWGfsA7lIL80Sp6wxtOxCW2aOy4gLxyLbRMpS7nPhZ6R2v.

[5] [Online]. Available: https://iies.in/blog/what-is-the-importance-of-hardware-software-co-design-in-embedded-systems/.

[6] [Online]. Available: https://www.geeksforgeeks.org/real-time-operating-system-rtos/.

[7] [Online]. Available: https://blackberry.qnx.com/en/ultimate-guides/what-is-real-time-operating-system/microkernel-architecture#:~:text=A%20microkernel%20architecture%20has%20a,need%20to%20modify%20the%20kernel..

[8] [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-52017-5_10.

[9] A. S. Tanenbaum and M. van Steen, "Distributed Systems: Principles and Paradigms," pearson education, 2006.

[10] P. Druschel, M. B. Abbott, M. Pagels, and L. L. Peterson, "Tailoring TCP/IP for high-performance networking," 1993.

[11] J. Liedtke, "On micro-kernel construction," Copper Mountain, CO, USA, 1995.

[12] J. Shapiro, J. S. Miller, and B. N. Hardy, "Eros: A fast capability system," Charleston, SC, USA, 1999.

8.Individual Contribution

Registration number	Name	Contribution
IT23184312	AMANTHA M A	Introduction and Significance
IT23170520	D.G.C.H.RAJASOORIYA	Problem and Solutions
IT23187832	PERAMUNUGAMA M.R.A	Results and Implications
IT23169708	PERERA P.A.J.M	Critical Evaluation and Relevance to Modern OS Relate to Broader Concept