International Journal of Advanced Smart Convergence Vol.11 No.1 28-35 (2022)
http://dx.doi.org/10.7236/IJASC.2022.11.1.28

IJASC 22-1-4

Performance Evaluation of Real-time Linux for an Industrial Real-time Platform

Yong Hwan Jo[1], Byoung Wook Choi[2*]

[1]*M.S., Department of Electrical and Information Engineering, Seoul National University of Science and Technology, South Korea*
[2*]*Professor, Department of Electrical and Information Engineering, Seoul National University of Science and Technology, South Korea*
[1]*jyh159@seoultech.ac.kr,* [2*]*bwchoi@seoultech.ac.kr*

Abstract

This paper presents a performance evaluation of real-time Linux for industrial real-time platforms. On industrial platforms, multicore processors are popular due to their work distribution efficiency and cost-effectiveness. Multicore processors, however, are not designed for applications with real-time constraints, and their performance capabilities depend on their core configurations. In order to assess the feasibility of a multicore processor for real-time applications, we conduct a performance evaluation of a general processor and a low-power processor to provide an experimental environment of real-time Linux on both Xenomai and RT-preempt considering the multicore configuration. The real-time performance is evaluated through scheduling latency and in an environment with loads on the CPU, memory, and network to consider an actual situation. The results show a difference between a low-power and a general-purpose processor, but from developer's point of view, it shows that the low-power processor is a proper solution to accommodate low power situations.

Keywords: *Xenomai, RT-preempt, General processor, Low-power processor, Multicore configuration*

1. Introduction

Advances in computers and semiconductors have led to breakthroughs in control devices and processors. As systems become more complex, however, the workload of the processor also increases. Accordingly, the use of multicore processors in various fields, such as general computing, robotics, control systems, and artificial intelligence, has become common [1-2]. The use of multicore processors brings advantages such as better work efficiency, lower power consumption, reduced heat generation, and lower prices [3].

The parallel computing of multicore processors is designed to satisfy tasks that do not require time constraints. However, real-time systems are defined by strict time constraints on tasks and processes. Commercial real-time operating systems (RTOS) such as VxWorks and Nucleus provide real-time scheduling solutions on multicore processors. However, RTOS are generally distributed as libraries and have disadvantages such as license and royalty costs, as well as technology dependence [4].

Manuscript Received: January. 14, 2022 / Revised: January. 20, 2022 / Accepted: January. 24, 2022
Corresponding Author: bwchoi@seoultech.ac.kr
Tel: +82-2-970-6412, Fax: +82-2-970-9732
Professor, Department of Electrical and Information Engineering, Seoul National University of Science and Technology, South Korea

Open-source software can overcome these disadvantages. Many efforts have been made in the community to ensure real-time performance on Linux, the most commonly used open-source operating system. In general, real-time Linux can be categorized depending on whether a dual-kernel approach or a fully preemptive kernel approach is used [5].

The dual-kernel approach uses a common kernel architecture in which the real-time kernel runs alongside standard Linux through an adaptive domain environment for operating systems (ADEOS). In this architecture, the real-time kernel has the highest priority. A standard Linux task can be executed only when there is no real-time execution queue, and RTAI and the Xenomai project are representative dual-kernel approaches [6-8].

A fully preemptive kernel approach uses an architecture that manages all tasks in a single real-time kernel. A typical example is RT-preempt, which expands the normal Linux kernel to a fully preemptive kernel via a real-time patch that includes modifications to the timer and scheduler. In terms of real-time implementation, RT-preempt is a single real-time Linux kernel. Accordingly, it has the advantage of easy extensibility because Linux-based libraries can be used. However, there is a disadvantage in that the performance changes depending on the version [9].

Although multicore processors have become common, real-time scaling does not support real-time scheduling for multicore processors. Because there is no real-time scheduler considering task migration in multicore processors, many studies have isolated specific cores and attached real-time tasks to the isolated cores or have disabled all physical and logical cores to mimic a single-processor system. Previous studies showed a change in real-time performance due to the use of C-state, hyperthreading, multicore and CPU isolation in a low-power dual-core processor [10]. However, in real-time applications with mobility, a low-power processor is often used. Therefore, a real-time performance analysis is required not only in general processors but also in low-power processors. During the effort to achieve this requirement, however, real-time performance according to the multicore distribution in a low-power processor was not considered in previous studies. A real-time performance analysis is conducted both on Xenomai for the dual-kernel architecture and fully preemptive RT-preempt for the single kernel. With regard to experimental conditions, an analysis was also performed between a general processor and a low-power processor in an environment with experimental loads for CPU, memory, and network tasks.

The paper aims to present indicators through a performance evaluation of real-time characteristics so that developers who aim to implement industrial real-time applications can properly choose a processor and real-time Linux. Xenomai and RT-preempt with real-time Linux with a multicore distribution on Intel multicore processor-based systems are considered. A real-time performance analysis is conducted between a general processor and a low-power processor on both architectures. As multiple cores are used and hardware performance capabilities are increasing, the real-time performances of the two architectures tested here differ. Therefore, the results present a useful guideline for applications, especially real-time control applications.

2. Real-time Linux Implementation

The Linux kernel was originally developed for Intel-based processors. For a comparison with the results in earlier work, an i7-6700 octa-core processor of the same generation as the processor in that study was used. The i7-6700 consists of four physical cores and four logical cores running at 4.00 GHz. Table 1 shows the specifications of the processor. By using the identical sixth generation Intel i7, a real-time performance analysis of a low-power processor and a general processor can be conducted. The two processors differ in terms of the number of cores and threads, but the greatest difference is the thermal design power (TDP). This is the maximum power used to cool the system as required to mitigate the heat inside the computer. To minimize the impact of the integrated graphics controller, the multicore system used Lubuntu 18.04, a Linux distribution.

Linux 4.14.134, which is the most stable version, was selected.

Table 1. Processor Specifications

Heading level	General processor	Low-power processor
Processor	Intel i7-6600	Intel i7-6600U
Cores	4	2
Threads	8	4
Processor base frequency	3.4GHz	2.6GHz
TDP	65W	15W

2.1 Xenomai

Xenomai is an interface for real-time tasks. Xenomai requires a hardware abstraction layer called ADEOS to utilize Xenomai and the Linux kernel simultaneously to implement a dual-kernel environment [11]. In this study, ipipe-core-4.14.134-x86-8 is used. Xenomai chose v3.1 as the most recent version in the GIT repository. In order for the Linux kernel to be capable of real-time performance after patching with ADEOS, several kernel options must be enabled/disabled.

Table 2. Xenomai Kernel Options

Kernel options	Enable/Disable
HIGH_RES-TIMERS	Enable
CONFIG_MCORE2	Enable
CONFIG_TRANSPARENT_HUGEPAGE	Disable
CONFIG_COMPACTION	Disable
CONFIG_CMA	Disable
CONFIG_MIGRATION	Disable
CONFIG_X86_SMAP	Disable
CONFIG_CPU_FREQ	Disable
CONFIG_ACPI_PROCESSOR	Disable
CONFIG_INTEL_IDLE	Disable
CONFIG_CPU_IDLE	Disable
CONFIG_KGDB	Disable

Power management is a critical factor related to latency in a real-time system and should be disabled. In addition, the kernel debugging function is another cause of latency that affect real-time requirements. In particular, the debugger KGDB function used to examine variables, the call stack information and the memory usage all affect latency; accordingly, KGDB should be disabled.

After the above settings are completed, the subsequent steps are to compile and install. If the bootloader is updated after compilation, the Xenomai development environment as shown in Figure 1(a) is built.

2.2 RT-preempt

RT-preempt patches the Linux kernel to support hard real-time tasks and fully preemptive scheduling. It has the advantage of being able to use Linux-based libraries (e.g., ROS, IgH EtherCAT) [12-14].

The RT-preempt patch is in the Linux kernel repository. We use version 4.14.134-rt63, which compatible

with the kernel used here. First, to make a fully pre-emptible kernel, CONFIG_PREEMPT_RT must be enabled. It is also necessary to enable/disable the Intel processor, memory pages and power management options, as described for Xenomai.

• Table 3. RT-preempt Kernel Options

Kernel options	Enable/Disable
HIGH_RES-TIMERS	Enable
CONFIG_MCORE2	Enable
CONFIG_SCHED_MC	Disable
CONFIG_TRANSPARENT_HUGEPAGE	Disable
CONFIG_CPU_FREQ	Disable
CONFIG_ACPI_PROCESSOR	Disable
CONFIG_INTEL_IDLE	Disable
CONFIG_CPU_IDLE	Disable
CONFIG_KGDB	Disable

After the above settings are completed, the next steps are to compile and install. If the bootloader is updated after compilation is completed, the RT-preempt development environment as shown in Figure 1(b) is built.

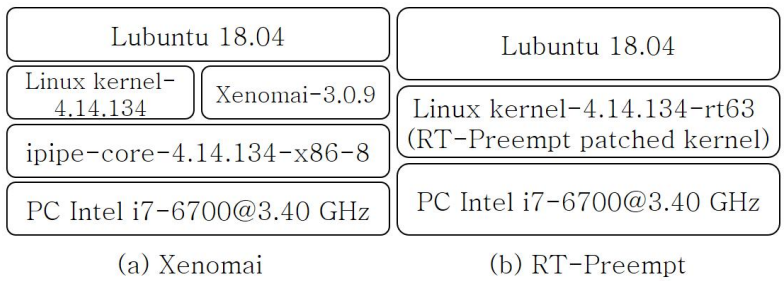

Figure 1. Real-time Linux architecture

2.3 Multicore deployment

There are numerous studies related to the effects of multicore configurations on the real-time performance of the entire system [15-16]. It is said that a multicore architecture adversely affects the periodicity of real-time tasks in RT-preempt. Also, in Xenomai, hyper threading was found to affect the real-time performance. In a low-power processor, it is necessary to disable C-state, disable hyperthreading, enable the multicore option, and enable CPU isolation. This leads to better results than in other multicore distributions [10].

3. Performance evaluation

In this article, we analyze the real-time performance capabilities of a general-purpose processor and a low-power processor in terms of scheduling latency. Scheduling latency represents the difference between the task's intended wake-up time and the actual wake-up time [17]. This is a consideration when designing real-time tasks. To measure the scheduling latency, use the *cyclictest* benchmark tool. This benchmark requires that the timer be calibrated to measure the ideal time in Xenomai. In Xenomai 3, we correct the gravity with the correction tool autotune. Using the cyclictest benchmark, the cycle was set to *100us* and the highest priority, 99, was assigned. Also, to prevent page errors, *mlockall()* was used, as was nanosleep. To ensure accuracy of

the experiment, measurements were taken for 100 seconds and 1,000,000 sufficient samples were obtained.

3.1 Idle environment

First, we gained the result when measuring the general processor in the absence of any task apart from the cyclictest benchmark.

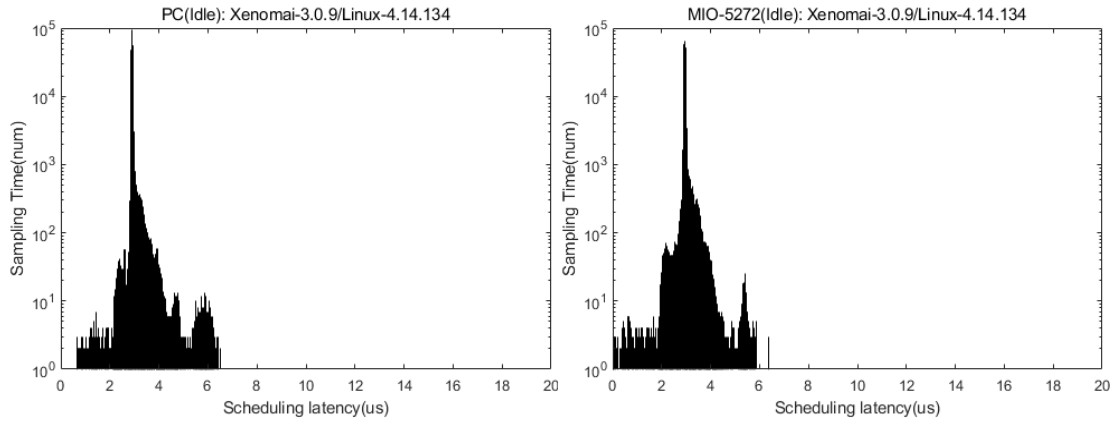

Figure 2. Histogram of scheduling latency in Xenomai (Idle)

The scheduling latency in Xenomai is displayed as a histogram, as shown in Figure 2. For Xenomai in the idle environment, it shows a similar distribution for the general-purpose processor and the low-power processor. In RT-preempt, the scheduling delay time is displayed as a histogram, as shown in Figure 3. For RT-preempt in the idle environment, a similar distribution is found for the general-purpose processor and for the low-power processor.

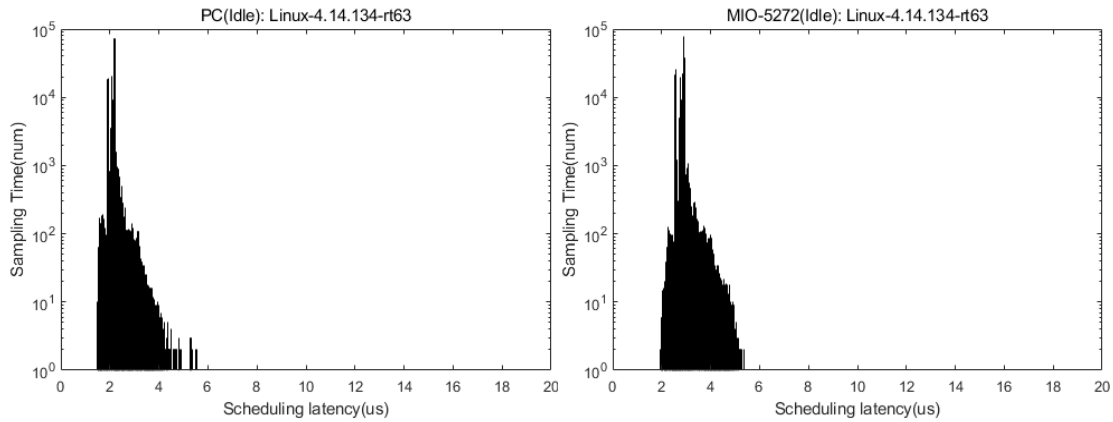

Figure 3. Histogram of scheduling latency in RT-preempt (Idle)

Table 2. Scheduling Latencies of Xenomai and RT-preempt in an Idle Environment

	Xenomai(us)		RT-preempt(us)	
	General	Low-power	General	Low-power
AVG	2.9177	2.9597	2.1393	2.8404
MAX	8.849	20.878	19.128	40.857
MIN	0.0532	0	0.1482	0.1925
STD	0.1372	0.1481	0.1696	0.2101

Table 2 shows the average, maximum, minimum, and standard deviation of the measured scheduling delay

in the idle environment. For Xenomai in the idle environment, an average time of 2.9177us and standard deviation 0.1372us were found for the general-purpose processor, and an average time of 2.9597us and standard deviation of 0.1481us were found for the low-power processor. In the RT-preempt case, the average time was 2.1393us and the standard deviation was 0.1696us with the general-purpose processor. The corresponding outcomes were 2.8404us and 0.2101us for the low-power processor.

3.2 Stress environment

To establish an environment suitable for industrial controllers, the scheduling delay time is measured in an environment affected by a load. The impact of the load was considered for CPU, memory, and network operations.

Here, stress-ng is used to load the CPU and memory, providing an extensive CPU-specific stress test in a very tight infinite loop. It was set to provide a 100% load to the CPU and to use 70% of the system memory.

For network operation, *iperf* is used. *iperf* runs as a server and client architecture. In this experiment, the real-time system acts as a server connected to another PC. By simulating 100 clients each requesting 64KB of data from the load generator, we obtain throughput of 1Mbps/client.

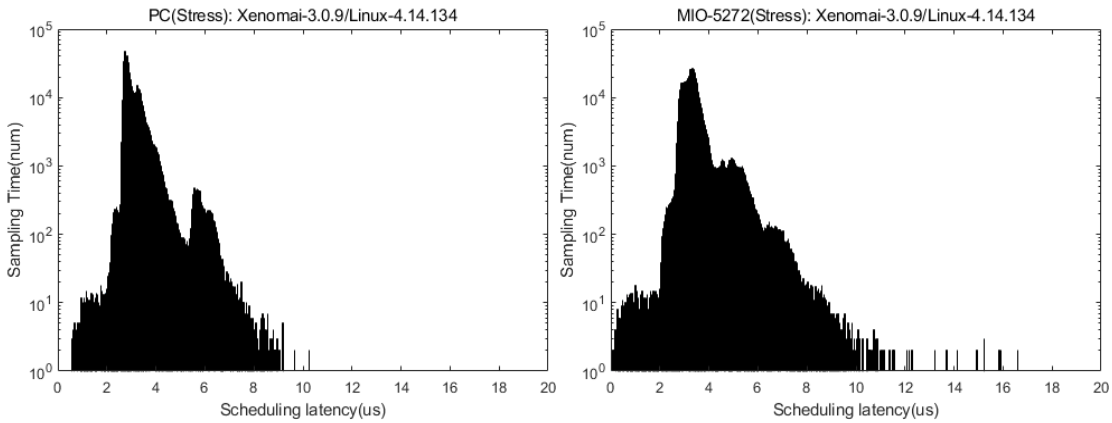

Figure 4. Histogram of scheduling latency in Xenomai (Stress)

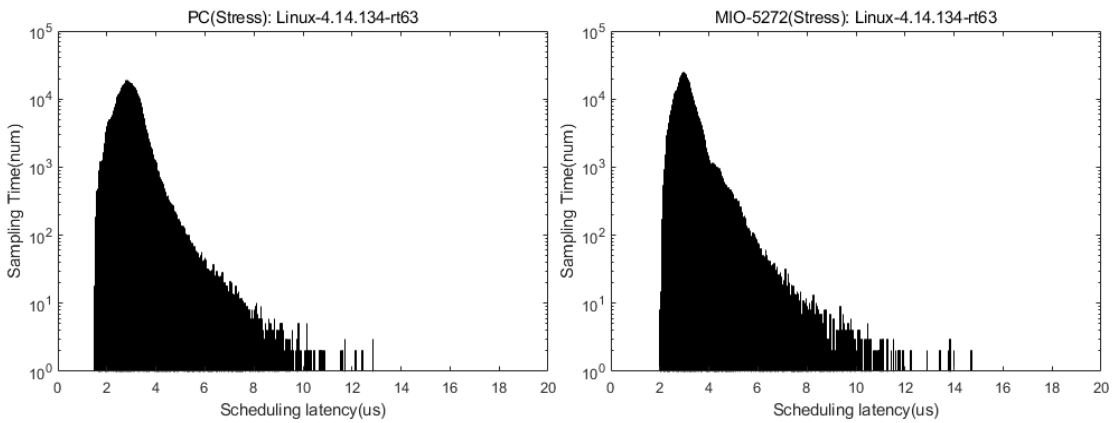

Figure 5. Histogram of scheduling latency in RT-preempt (Stress

Table 3. Scheduling Latencies of Xenomai and RT-preempt in a Stressed Environment

	Xenomai(us)		RT-preempt(us)	
	General	Low-power	General	Low-power
AVG	3.1005	3.4234	2.8976	3.1002
MAX	11.922	50.626	27,065	37.212
MIN	0.0515	0.024	0.1487	0.1998
STD	0.5372	0.6836	0.5537	0.5596

The scheduling latency in Xenomai and in RT-preempt is displayed as a histogram in Figure 4 and 5, respectively. And Table 3 shows the average, maximum, minimum, and standard deviation of the measured scheduling latency in a stressed environment. For Xenomai in a stressed environment, an average time of 3.1005us and standard deviation of 0.5372us were found for the general-purpose processor, while an average time of 3.4234us and standard deviation of 0.16836us were measured for the low-power processor. In the RT-preempt case, an average time of 2.8976us and standard deviation of 0.5537us were found on the general-purpose processor, while an average time of 3.1002us and standard deviation 0.5596us were measured for the low-power processor.

4. Conclusion

In this paper, we presented performance evaluation results for real-time industrial applications when choosing a processor and real-time Linux. To help with processor selection, real-time performance evaluations of the low-power processor and the general processor were conducted. The Xenomai and RT-preempt architectures were adapted for the real-time Linux architecture. The evaluation was also performed in idle and stressed environments to verify that the controller operates stably even in real environment.

As shown in Table 1, the general-purpose processor has better computing power than the low-power processor in terms of the number of cores, the number of threads, and the processor frequency. The experimental results also show that the general-purpose processor outperforms the low-power processor. However, for real-time application of Xenomai in an idle environment, the average latency of them were nearly identical, with only a 0.01us difference in the standard deviation. In the RT-preempt case, the mean differs by 0.7us and the standard deviation differs by 0.4us. In a stressed environment, the average latency for both of Xenomai is identical at 0.3us, and the standard deviation of both is 0.15us. In the RT-preempt case, the mean values for both cases are the same as 0.2us and the standard deviation is nearly identical as well.

In general, the real-time tasks used in industry operate with periodicity of 1 ms or more [10, 18]. In such applications, a time of less than 1us does not significantly affect the real-time performance. Therefore, even if the specifications of the low-power processor are lower than those of the general-purpose processor, the former can be used real-time applications considering deterministic tasks.

In conclusion, this paper provides very useful design parameters by comparing and analyzing the real-time performance capabilities of both processors when implementing an industrial real-time platform through various experiments. Low power has become a very important design specification for a mobile controller, and accordingly this paper shows useful results for real-time applications when considering battery consuming.

Acknowledgement

This work has been financially supported by SeoulTech (Seoul National University of Science and Technology).

References

[1] D. Cho, "A Study on Effect of Code Distribution and Data Replication for Multicore Computing Architectures," International Journal of Advanced Culture Technology, vol. 9, no. 4, pp. 282–287, Dec. 2021. DOI: https://doi.org/10.17703/IJACT.2021.9.4.282.

[2] S.-H. Jeon, C.-G. Lee, J.-D. Lee, B.-S. Kim, and J.-M. Kim, "Implementation of AIoT Edge Cluster System via Distributed Deep Learning Pipeline," International journal of advanced smart convergence, vol. 10, no. 4, pp. 278–288, Dec. 2021. DOI: https://doi.org/10.7236/IJASC.2021.10.4.278.

[3] G. Kronaros. Multi-Core Embedded Systems, CRC Press, Boca Raton, 2010.

[4] Real-time-operating-system-rtos, https://www.geeksforgeeks.org/real-time-operating-system-rtos/

[5] T .Bijlsma, M. Kwakkernaat, M. Mnatsakanyan, "A real-time multi-sensor fusion platform for automated driving application development," IEEE 13th Int. Conf. Ind. Inform. (INDIN) pp. 1372–1377, Jul. 2015, DOI: https://doi.org/10.1109/INDIN.2015.7281935.

[6] RTAI ,http://www.rtai.org.

[7] Xenomai, https://xenomai.org/

[8] J.H. Koh, B.W. Choi, "Performance Evaluation of Real-time Mechanisms for Real-time Embedded Linux," Journal of Institute of Control Robotics and Systems, vol. 18, no 4, p. 337-342, Apr. 2012. DOI: https://doi.org/10.5302/J.ICROS.2012.18.4.337.

[9] G. K. Adam, N. Petrellis, and L. T. Doulos, "Performance Assessment of Linux Kernels with PREEMPT_RT on ARM-Based Embedded Devices," Electronics, vol. 10, no. 11, p. 1331, Jun. 2021, DOI: https://doi.org/10.3390/electronics10111331.

[10] R. Delgado and B. W. Choi, "New Insights Into the Real-Time Performance of a Multicore Processor," in IEEE Access, vol. 8, pp. 186199-186211, 202. DOI: https://doi.org/10.1109/ACCESS.2020.3029858.

[11] Life with Adeos, https://xenomai.org/documentation/branches/v2.4.x/pd f/life-with-adeos.pdf

[12] J. Park, R. Delgado and B. W. Choi, "Real-Time Characteristics of ROS 2.0 in Multiagent Robot Systems: An Empirical Study," in IEEE Access, vol. 8, pp. 154637-154651, 2020. DOI: https://doi.org/10.1109/ACCESS.2020.3018122.

[13] M. Cereia, I. C. Bertolotti and S. Scanzio, "Performance of a Real-Time EtherCAT Master Under Linux," in IEEE Transactions on Industrial Informatics, vol. 7, no. 4, pp. 679-687, Nov. 2011. DOI: https://doi.org/10.1109/TII.2011.2166777.

[14] J. Kim and C. Moon, "A Robot System Maintained with Renewable Energy," International journal of advanced smart convergence, vol. 8, no. 1, pp. 98–105, Mar. 2019. DOI**:** https://doi.org/10.7236/IJASC.2019.8.1.98.

[15] Litayem, Nabil, and S. Ben Saoud. "Impact of the linux real-time enhancements on the system performances for multi-core intel architectures." International Journal of Computer Applications 17.3 (2011): 17-23. DOI: https://doi.org/10.5120/2202-2796.

[16] C. Garre, " Performance comparison of real-time and general-purpose operating systems in parallel physical simulation with high computational cost," SAE Technical Paper, No. 2014-01-0200. 2014. DOI: https://doi.org/10.4271/2014-01-0200

[17] F. Cerqueira and B. Brandenburg. "A comparison of scheduling latency in linux, preempt-rt, and litmus rt." 9th Annual workshop on operating systems platforms for embedded real-time applications. SYSGO AG, 2013.

[18] D.S. Lee and H.J. Ahn, "Real-Time Characteristics Analysis and Improvement for OPRoS Component Scheduler on Windows NT Operating System," Journal of Institute of Control, Robotics and Systems, vol. 17, no. 1. Institute of Control, Robotics and Systems, pp. 38–46, Jan, 2011. DOI**:** https://doi.org/10.5302/J.ICROS.2011.17.1.38