Evaluation of General-Purpose and Real-Time Operating Systems Performance for Multi-Robot Systems Path Planning

Nishma A S[1], Krishna Priya K S[2], Minu Marshal[3], Jeshika S[4]

Department of Computer Science and Engineering, Arunachala College of Engineering for Women

DOI: https://doi.org/10.51584/IJRIAS.2024.90335

Received: 15 February 2024; Revised: 07 March 2024; Accepted: 12 March 2024; Published: 17 April 2024

ABSTRACT

Real-Time operating systems (RTOS) and General-Purpose Operating Systems (GPOS) are the two main components of contemporary operating systems, in general. The main differentiation between RTOS and GPOS lies in the urgency factor, signifying that a high-priority thread in GPOS cannot interrupt a kernelcall. However, with RTOS, even while a low-priority process is carrying out a kernel call, it might be preempted by a high-priority task if needed. When kernel modifications are made, the majority of Linux distributions can function as both GPOS and RTOS. Two Linux distributions, Ubuntu and Pardus, were examined in this study, and their functions as GPOS and RTOS for multi-robot system route planning were compared. Robot groups with varying membership counts were employed to carry out the path tracking.

Key Words: GPOS, Mobile robotics, Multi-robot systems, RTOS, Path Planning

INTRODUCTION

Nowadays, Electronic gadgets are increasingly developed to run autonomously with operating systems to fulfill specific tasks [1-5]. Commonly used operating systems include Android for mobile devices, Windows for PCs, Linux, and iOS for both platforms. Operating systems aim to optimize hardware utilization and speed up tasks [6]. Operating systems require certain dependencies to execute specific tasks. One of the most significant dependencies for operating systems is time. The computation of processing time is an essential quantity, particularly when dealing with time-dependent tasks and operations. It is a crucial factor that establishes how well an operating system performs and reduces the potential for error [6]. Time dependencies exist in all operating systems as a result. Two distinct time-calculation techniques have been employed, and they were given names according to the types of operating systems on which they depend [4], [7]. They are real-time operating systems (RTOS) and general purpose operating systems (GPOS).

Like Windows, Linux, and Android, GPOS is a popular operating system type intended for personal use. High efficiency is the primary goal of GPOS, an operating system design that is accessible to anyone. Efficiency and the quantity of jobs finished in a unit cycle are directly correlated. This implies that multitasking needs to be supported by GPOS [8]. However, in GPOS, task scheduling and allocation are carried out without regard to the tasks’ priority levels. This Increases system performance, which is contingent upon the task-scheduling algorithms. It implies that simultaneous increases in system efficiency are possible. Nonetheless, delays in GPOS processing times are conceivable. This is a reasonable mistakethat may be fixed by applying Efficient algorithms for work scheduling and allocation. However, if the task involves a time dependency, this could lead to significant errors [4]. For instance, when an obstruction is recognized in mobile robot obstacle avoidance, an obstacle avoidance task needs to be completed concurrently. It implies that the task has to be given more importance. If not, the robot can run into the obstruction and have an accident. This is an unforeseen and undesired circumstance. The high-priority task should be completed promptly to prevent such an error.

High-time dependency jobs have been the focus of RTOS design [9]. Based on their dependence on time, RTOS can be further classified into three classes. They are known as firm real-time, hard real-time, and soft real-time, in that order [9], [3]. The processing and completion times for tasks are unquestionably RTOS is certain. The activity is stopped or completed in a specific amount of time since it will conclude at the conclusion of a predetermined maximum scheduling time [11]. Nonetheless, depending on the kind ofRTOS, the system may respond differently in some potential delays. In soft real-time systems, delays can be tolerated; in hard real-time systems, they are not tolerant; and in firm real-time systems, there is no tolerance. In firm real-time systems, a task is canceled or terminated if it is not finished on time. Highly accurate decision-making is essential for applications involving autonomous robots in dynamic, uncharted environments. One of the jobs with a high time dependency is the decision-making process, particularly the identification and detection of impediments.

Two Linux distributions, Pardus and Ubuntu, were examined in this study for their GPOS and RTOS versions in order to compare how well they performed when it came to multi-robot system path planning. Additionally, the Robot Operating System's (ROS) performance on these two distributions can be compared in this Research. Using Ubuntu and Pardus as GPOS and RTOS, robot groups with varying numbers of members completed the route tracking tasks, and their performance was examined in the turtlesim simulation environment. Thus, two distinct Linux distributions' performances in multi-robotic applications were noted and contrasted. The outcomes of two distinct path planning situations were examined andanalyzed.

METHOD OF RESEARCH

In this study, system performance was examined after GPOS and RTOS were installed on Ubuntu and Pardus, two distinct Linux distributions. One of the most widely used Linux distributions, Ubuntu uses the Debian architecture and infrastructure and is based on the Linux kernel. The Linux-based operating system known as Pardus, which is currently the most widely used distributed operating system, has been in development since October 2004. It is an open-source system created by Ulakbim and Tubitak, utilizing the Debian framework. Since 2005, its distributions have been available for purchase. Research has been conducted on ROS using simulated experiments. ROS, although not considered a true operating system, is a software built on an operating system and is primarily used for robot control. By installing the turtlesim package in the ROS noetic version, mobile robots can be tested in a virtual environment. The turtle-like Package serves as a visualization tool that allows users to work on ROS and observe the movements of a virtual robot through user programs. Additionally, a trajectory planning application for various robot systems was developed using the ROS rqt plugin for Turtlesim. The application's license file grants permissions for free distribution and usage.

Swarm robots assist in drawing, a planned path from a selected picture using the turtlesim plug-in in the rqtof the number of robots used to draw the image varies depending on its size and intricacy. The machine in case studies has an Intel Atom® N2600 1.6 Ghz processor with 2 GB of RAM, 500 GB hard drive andDDR3 RAM were used.

EXPERIMENTAL STUDIES

Every experimental study was created with the goal of controlling the formation and path planning ofseveral robot systems. Operating system official logos were utilized. All case studies used to depict the pathways taken by the robots.

Experiment 1: Use The Pardus Logo to Draw a Path.

The Pardus logo was created in this project as a straightforward path planning using the Ubuntu GPOS-RTOS and Pardus GPOS-RTOS operating systems were both utilized with identical settings. Performance comparison was done and the results were noted, Figure 1 shows the official Pardus logo along with a few chosen pathways displayed individually. The code was executed with no modifications made, and four distinct operating systems produced the same shape. Turtlesim is used to depict robots drawing routes in Figure 2. As a result, measurements and records of CPU use and processing time on the Turtlesim were made in the data table. The individual results were displayed in Figure 3. Based on the findings displayed in Figure 3, when Ubuntu GPOS is utilized, Central Processing Unit (CPU) utilization is reduced and processing time is shortened compared to other operating systems. Conversely, Figure 3 also shows that RTOS used less CPU power than GPOS versions.

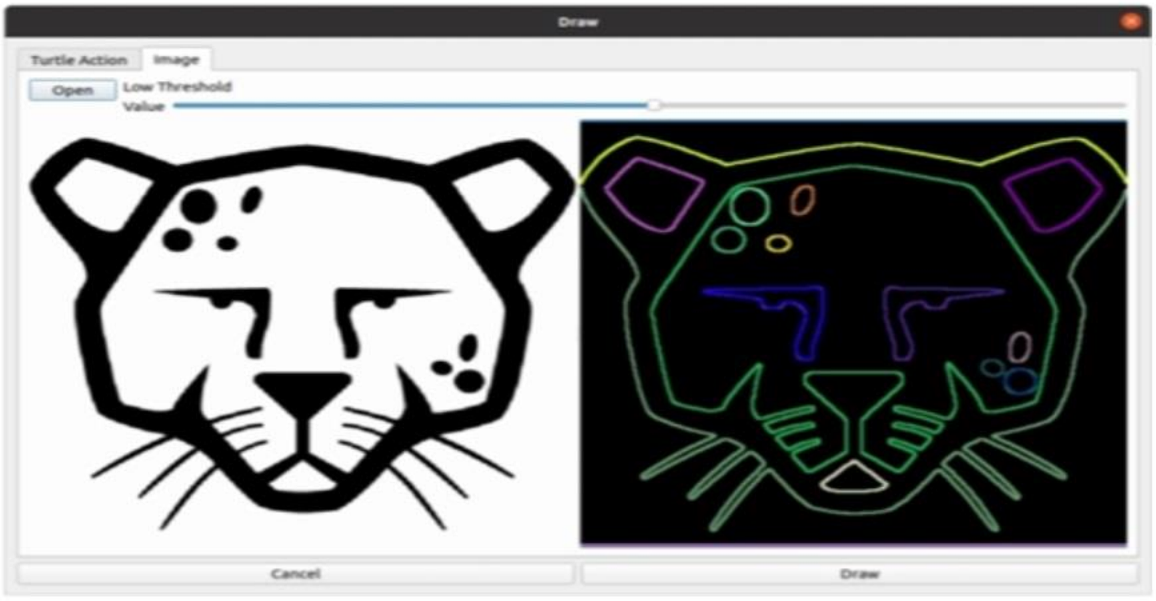

Figure 1. RQT Turtlesim View and the Pardus logo

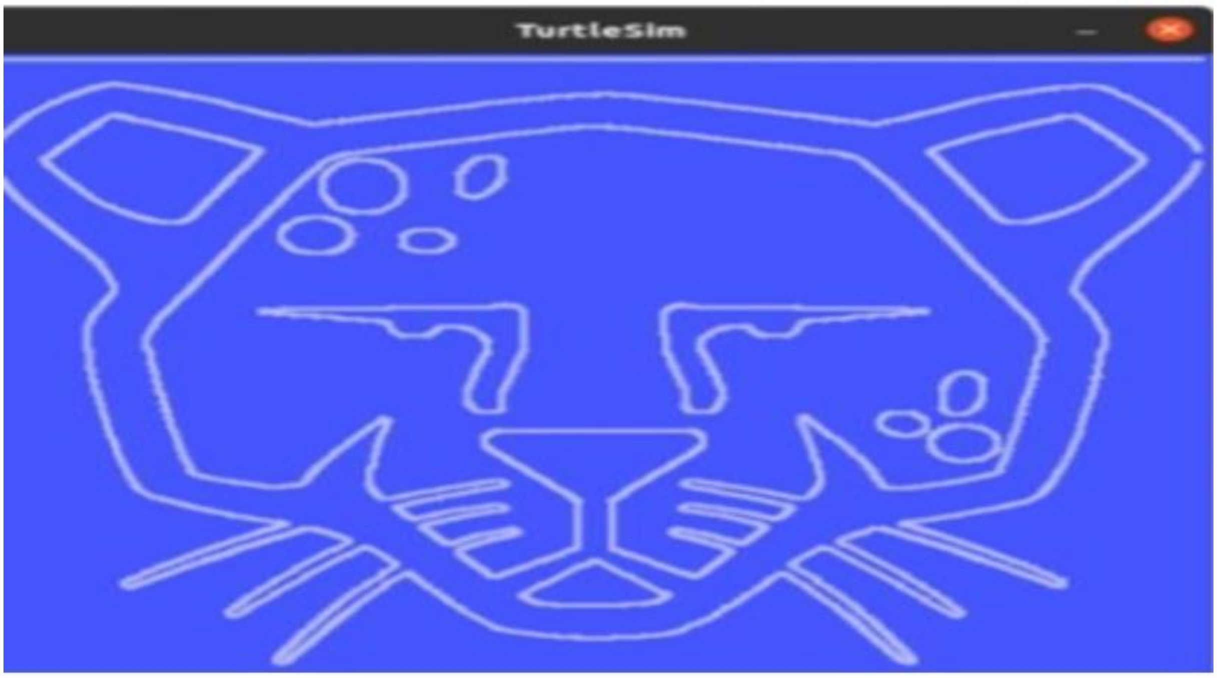

Figure 2. Sketching the Pardus logo route outcome with Turtlesim

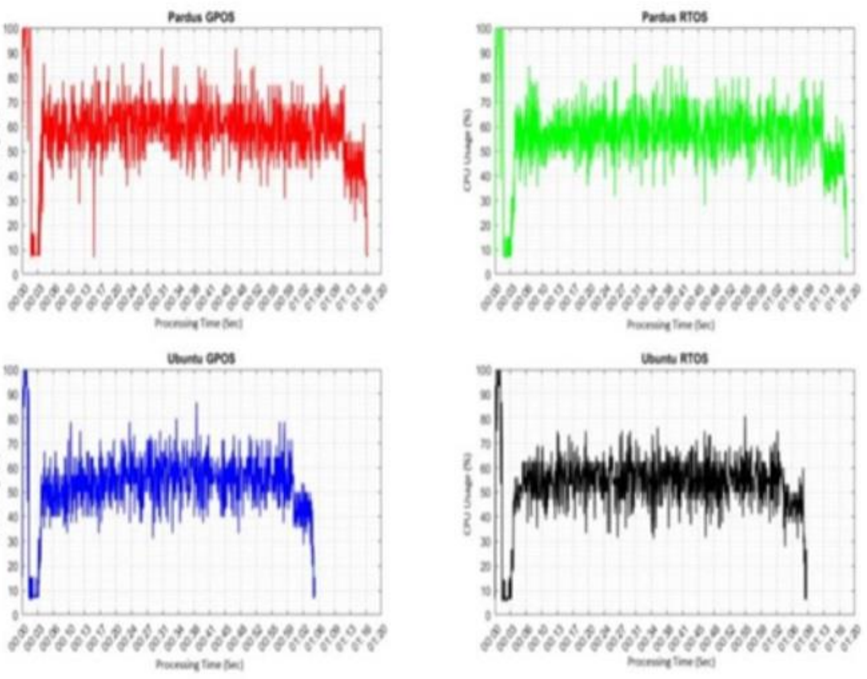

Figure 3. Processor usage graphs for experiment 1

Experiment 2: Use The Ubuntu-Linux Logo to Draw a Path

The Ubuntu-Linux logo was created in this attempt using for basic path planning. The Ubuntu GPOS-RTOS and Pardus GPOS-RTOS operating systems were both utilized with identical configurations. Performance comparison was done and the results were noted. Figure 4 displays the official Ubuntu-Linux logo and a few pathways were displayed individually. Four distinct operating systems displayed the same form when the code was ran with its default settings unaltered. Turtlesim is used to depict robots drawing routes in Figure
5. As a result, measurements and records of CPU use and processing time on the Turtlesim were made in the data table. Results were presented in Figure 6, separately. According to results which were shown in Figure 6, CPU usage is lower and processing time is shorter than other operating systems when Ubuntu GPOS was used. On the other hand, low CPU usage of RTOS were observed than GPOS versions, in Figure 6. LowCPU usage of RTOS was observed in both experiments. We can say that the main reason for this is that the long processes are terminated, and the other process is executed, in RTOS.

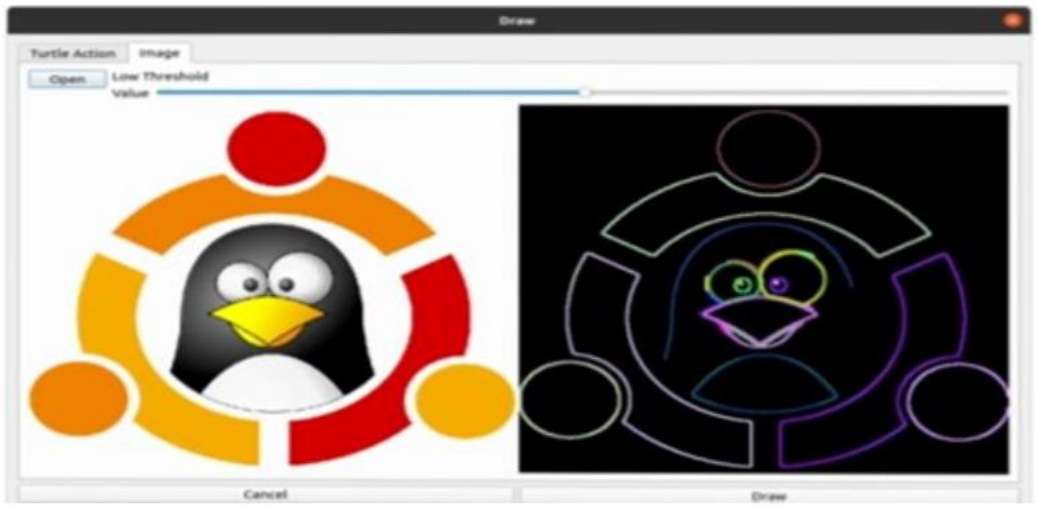

Figure 4. Ubuntu-Linux logo with the view of RQT Turtlesim

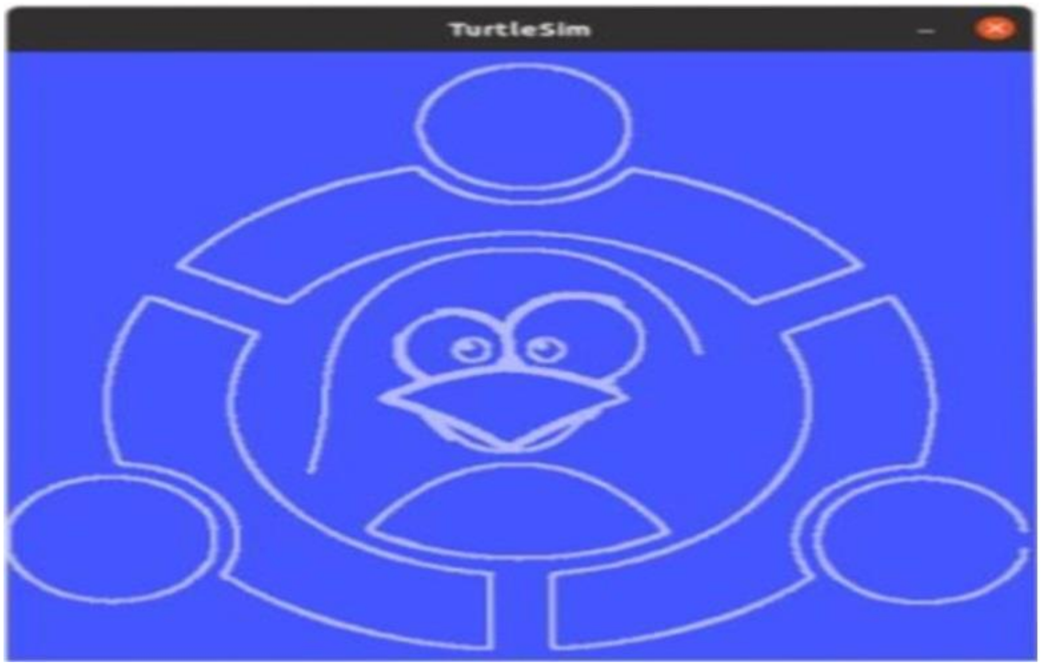

Figure 5. Using turtlesim, the Ubuntu-Linux logo path result is drawn.

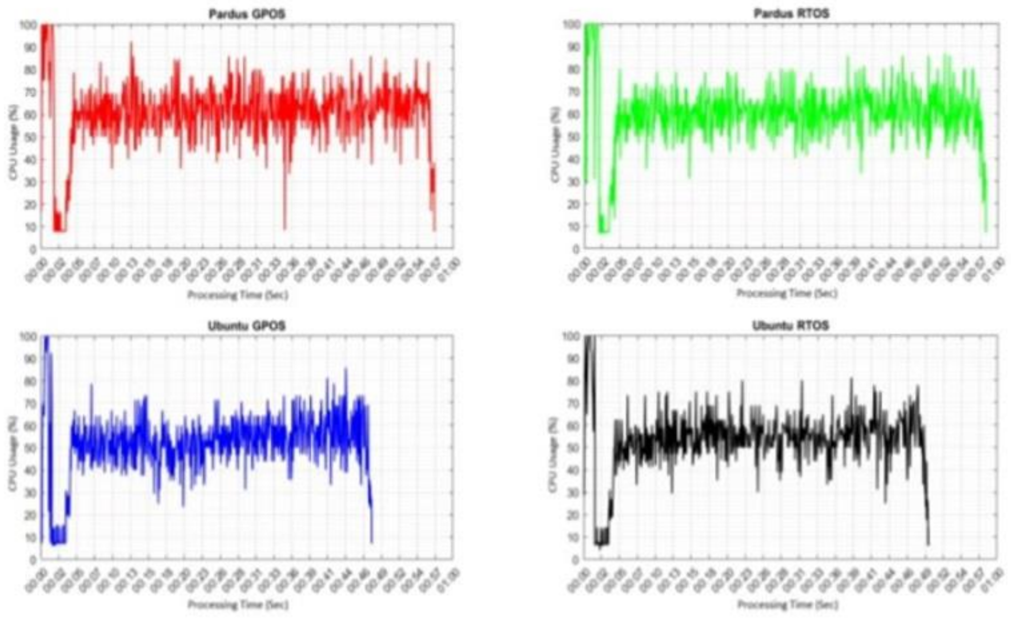

Figure 6. Graphs showing processor use in experiment 2.

RESEARCH APPROACHES

Using on all operating systems, complicated trajectory tracking and formation control for multi-robot systems were carried out in experiments 1 and 2. Table 1 shows the processing times for every operating system. Table 1 presents the results of a comparison between RTOS and GPOS. GPOS outperform RTOS in terms of processing time. The primary cause of this is because, as each robot completes a drawing task in GPOS, the operating system reads the value and the subsequent robot's process starts. However, the start and finish times of the process are set in RTOS prior to its initiation. When each process has a 100 millisecond processing time limit, resources will still be accessible for the other robot at the end of the 100 milliseconds, even if one robot finishes its mission earlier. While this lowers the margin of error and strengthens system security, it increases processing time, which results in delays in processing times.

Two distinct outcomes in the multi-robot performance evaluation in path planning and formation control are evident as a consequence of this application. First, in two separate operating systems, GPOS finished their duty sooner than RTOS. The second one is RTOS and GPOS for Ubuntu. Finished tasks more quickly than Pardus with constrained resources. Simultaneously, it has been noted that every operating system has successfully finished its job.

CONCLUSION

To allow for free modifications to the study's basic codes and structure, Ubuntu and Pardus, two open-source operating systems, were employed as operating systems. This study's primary goal is to assess and contrast the usability of GPOS and RTOS in multi-robot systems. Tasks completed by RTOS were completed after the GPOS as anticipated. Because the operating system read the value at the conclusion of the precise processing time allocated for the processes, even if the processes were completed. The operating systems structural characteristics cause variations in the outcomes. Considerations including potential failure scenarios, process security, and processing time speeds should be made while selecting an operating system for multi-robot applications. When a process runs on an RTOS, it ends or is finished at a specifictime. If a robot is to be utilized in time-dependent applications, this capability is crucial. When a task experiences a delay, it is canceled and the system as a whole is not impacted because the processing timeframes are well defined. By a mistake that happens but the system keeps working. Using two different Linux distributions, GPOS and RTOS were compared for multi-robot system's path planning and formation control. The experimental findings showed that GPOS completed all defined tasks more quickly than RTOS. When conducting research in the field of multi-robotic systems, GPOS can be chosen in a way that prioritizes performance. RTOS must be used if a task has a time dependency or needs a specific amount of processing time.

REFERENCES

1. D. P. Watson and D. H. Scheidt, “Autonomous systems,” Johns Hopkins APL Technical Digest (Applied Physics Laboratory), Vol. 26, no. 4, 2005, doi: 10.1201/b17251-10.
2. C. S. Sharp, O. Shakernia, and S. S. Sastry, “A vision system for landing an unmanned aerial vehicle,” Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164), vol. 2, 2001, pp. 1720-1727, doi: 10.1109/ROBOT.2001.932859.
3. P. Hambarde, R. Varma, and S. Jha, “The survey of real time operating system: RTOS,” 2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies, 2014, pp. 34-39, doi: 10.1109/ICESC.2014.15.
4. K. Ghosh, B. Mukherjee, and K. Schwan, “A survey of real-time operating systems,” NetworksParallel Distrib. Process., vol. 29, No. GIT-CC-93/18, 1994.
5. H. Posadas, E. Villar, D. Ragot, and M. Martinez, “Early modeling of linux-based ROTS platforms in a system c time-Approximate co-simulation environment,” 2010 13th IEEE International Symposium
on Object/Component/Service-Oriented Real-Time Distributed Computing, 2010, pp. 238-244, doi: 10.1109/ISORC.2010.18.
6. Murikipudi, V. Prakash, and T. Vigneswaran, “Performance analysis of real time operating systemwith general purpose Operating system for mobile robotic system,” Indian Journal of Science and Technology, vol. 8, no. 19, pp. 1-6, 2015, doi: 10.17485/ijst/2015/v8i19/77017.
7. Garre, D. Mundo, M. Gubitosa, and A. Toso, “Performance comparison of real-time and general-purpose operating systems in Parallel physical simulation with high computational cost,” in SAE Technical Papers, 2014, vol. 1, doi: 10.4271/2014-01-0200.
8. M. D. Marieska, P. G. Hariyanto, M. F. Fauzan, A. I. Kistijantoro, and A. Manaf, “On performance of kernel based and embedded Real-time operating system: Benchmarking and analysis,” in 2011 International Conference on Advanced Computer Science and Information Systems (ICACSIS),

2011.

9. S. Baskiyar and N. Meghanathan, “A survey of contemporary real-time operating systems,” Informatica, vol. 29, no. 2, pp. 233-240, 2005.
10. R. Aslanian, “Real-time operating systems,” Computer Standards and Interfaces, vol. 6, no. 1, pp. 45- 49, 1987, doi:10.1016/0920-5489(87)90044-4.
11. P. Hambarde, R. Varma, and S. Jha, “The Survey of real time operating system: RTOS,” 2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies, 2014, pp. 34-39, doi: 10.1109/ICESC.2014.15.