Performance analysis of real-time and general-purpose operating systems for path planning of the multi-robot systems

Article *in* International Journal of Electrical and Computer Engineering (IJECE) · February 2022

DOI: 10.11591/ijece.v12i1.pp285-292

CITATIONS
5

READS
1,509

2 authors:

Seçkin Canbaz
Istanbul Sabahattin Zaim University
3 PUBLICATIONS **6** CITATIONS
SEE PROFILE

Gokhan Erdemir
University of Tennessee at Chattanooga
36 PUBLICATIONS **151** CITATIONS
SEE PROFILE

International Journal of Electrical and Computer Engineering (IJECE)
Vol. 12, No. 1, February 2022, pp. 285~292
ISSN: 2088-8708, DOI: 10.11591/ijece.v12i1.pp285-292

Performance analysis of real-time and general-purpose operating systems for path planning of the multi-robot systems

Seçkin Canbaz[1], Gökhan Erdemir[2]
[1]Department of Information Technologies, İstanbul Sabahattin Zaim University, Instanbul, Turkey
[2]Department of Electrical and Electronics Engineering, İstanbul Sabahattin Zaim University, Instanbul, Turkey

Article Info

Article history:

Received Mar 31, 2020
Revised Jun 17, 2021
Accepted Jun 30, 2021

Keywords:

GPOS
Mobile robotics
Multi-robot systems
Path planning
RTOS

ABSTRACT

In general, modern operating systems can be divided into two essential parts, real-time operating systems (RTOS) and general-purpose operating systems (GPOS). The main difference between GPOS and RTOS is the system is time-critical or not. It means that; in GPOS, a high-priority thread cannot preempt a kernel call. But, in RTOS, a low-priority task is preempted by a high-priority task if necessary, even if it's executing a kernel call. Most Linux distributions can be used as both GPOS and RTOS with kernel modifications. In this study, two Linux distributions, Ubuntu and Pardus, were analyzed and their performances were compared both as GPOS and RTOS for path planning of the multi-robot systems. Robot groups with different numbers of members were used to perform the path tracking tasks using both Ubuntu and Pardus as GPOS and RTOS. In this way, both the performance of two different Linux distributions in robotic applications were observed and compared in two forms, GPOS, and RTOS.

Corresponding Author:

Gökhan Erdemir
Department of Electrical and Electronics Engineering, Istanbul Sabahattin Zaim University
Halkali, Kucukcekmece, Istanbul 34303, Turkey
Email: gokhan.erdemir@izu.edu.tr

1. INTRODUCTION

In recent, almost all of electronic devices are designed for the devices to operate independently from human beings and autonomously have operating systems to perform a specific task [1]-[5]. Commonly used operating systems can be listed as android for mobile devices, windows for PCs, Linux and iOS for both platforms. The main purpose of operating systems is to use the hardware most effectively and also to fulfill the tasks in the fastest way [6]. On the other hand, operating systems have some dependencies while performing described tasks. The time is one of the important dependencies for operating systems. The calculation of the processing time is crucial parameter in especially time depended on operations and tasks. It is an important parameter that determines the performance of an operating system and minimizes the possibility of error [6]. Therefore, all operating systems have time dependencies. Two different time calculation methods have been used which were named based on their kernel structures which depend on operating systems type [4], [7]-[13]. These are general purpose operating system (GPOS) and real-time operating system (RTOS).

GPOS is a common operating system type such as windows, Linux, android. that is designed to fulfill personal use. The main purpose of GPOS, which is an operating system design that can be used by everybody, is high efficiency. Efficiency is directly related to the number of tasks completed per unit cycle. It means that GPOSs must support multi-tasking [8], [14]. On the other hand, task scheduling and allocation are performed without priority levels of the tasks in GPOSs [15]. It is possible to execute lower priority tasks instead of higher priority tasks, primarily. In this way, system performance can be increased which depends on the task-scheduling algorithms. It means that system efficiency can increase at the same time. However, it is possible to occur delays in processing time in GPOS. This is a tolerable error, and it can be solved by using effective task-scheduling and allocation algorithms. But, this situation can cause crucial errors if the task has time dependency [4], [16]-[19]. For example, in the obstacle avoidance for mobile robots, when an obstacle is detected, an obstacle avoidance task must be performed simultaneously. It means that the task must have higher priority. Otherwise, the robot may hit the obstacle and an accident occurs. This is an undesirable and unexpected situation. In order to avoid such an error, the task with high priority should be executed quickly.

RTOSs have been designed to solve high-time dependency tasks [9], [16], [17], [20]. RTOSs are divided into three subcategories according to their time dependency. These are soft real-time, hard real-time, and firm real-time, respectively [3], [9], [21]. The task processing and completion times are absolutely certain in RTOS [22]. Since the task that is processed will finish at the end of a predetermined maximum scheduling time, it is terminated or completed in a certain time [11], [23]. However, in some possible delays, the responses of the system are various which depends on the type of RTOS. Delays are tolerable in soft real-time systems, poorly tolerable in hard real-time systems, and no tolerance in firm real-time systems [12], [24]-[26]. If the task is not completed on time, it is terminated or canceled in firm real-time systems [12], [24]-[26]. It is crucially important to have highly precise decision ability for autonomous robot applications in a dynamic and unknown environment [27]-[29]. The decision process is one of the high-time dependency tasks, especially, detection and identification of obstacles [28], [30], [31].

RTOSs are divided into two types according to their core architectures which are monolithic kernel and microkernel [32]-[35], as well. In a monolithic kernel, all processes in the operating system are defined in the kernel. All processes such as file management and networking. run in the kernel. On the other hand, applications work on the user side. The disadvantage of this design is the entire system can be affected by any negativity in the kernel [32], [33]. In this structure, the whole kernel must be recompiled for a change that can be performed [32], [33]. And, this process is a cause of the time loss [32], [33]. The whole operating system processes work on the user side in microkernel systems [32]-[34]. All processes communicate with each other. This approach increases the message traffic and decreases performance. But, the microkernel architecture is more secure than the monolithic kernel architecture. An occurred error during task-processing does not affect the whole system. It is not more complex because it contains less code [32]-[34].

In this study, GPOS and RTOS versions of Ubuntu and Pardus which are two Linux distributions were analyzed to compare their performances on path planning of the multi-robot systems. Moreover, it is possible to compare the performance of robot operating system (ROS) on these two distributions, in this study. Robot groups with different numbers of members were used to perform the path tracking tasks using both Ubuntu and Pardus as GPOS and RTOS to analyze their performance in the turtlesim simulation environment. In this way, both the performance of two different Linux distributions in multi-robotic applications were observed and compared. Two different path planning cases were performed, and results were discussed.

2. RESEARCH METHOD

In this study, GPOS and RTOS were installed on two different Linux distributions which are Ubuntu and Pardus, and system performances were analyzed. Ubuntu is one of the most popular Linux distribution with a Linux kernel, using the Debian infrastructure and architecture [36]. This operating system, distributed as open-source, has been developed since October 2004 and is currently the most popular Linux-based system [36]. Pardus is an open-source operating system developed by Ulakbim and Tübitak using the Debian infrastructure. Its distributions have been on the market since 2005 [37]. Experimental studies on the simulation have been performed on ROS. Although ROS is perceived as a real operating system, it is actually software based on an operating system and used for robot control [38]-[43]. It is possible to test mobile robots in a virtual environment with the turtlesim package installed in the ROS noetic version. The turtlesim package is a visualization tool that allows us to observe the movements of a virtual robot by working on ROS via user codes [38]-[40]. ROS rqt plugin for turtlesim was used as a trajectory planning application of multi-robot systems [44]. Free distribution and usage permissions are given in the license file in the application [45]. The turtlesim plug-in in the rqt of [44], a selected picture as a planned path is drawn with the help of swarm robots. In drawing the picture, the number of robots is various according to the size and complexity of the picture [44]. In case studies, the computer which has an Intel Atom® N2600 1.6 Ghz processor, 2 GB DDR3 RAM, 500 GB hard disk was used.

3. EXPERIMENTAL STUDIES

All experimental studies were designed for path planning and formation control of multi-robot system. Official logos of operating systems were used. In all case studies, [44] were used to draw robots's paths.

3.1. Experiment 1 draw a path using Pardus logo

In this experiment, Pardus logo was drawn as simple path planning using [44]. Pardus GPOS-RTOS and Ubuntu GPOS-RTOS operating systems were used with the same parameters. The results were recorded, and performance comparison was performed. In Figure 1, Pardus official logo and selected paths were shown, separately. The code was run with its default settings without changing and the same shape was drawn in 4 different operating systems. In Figure 2, drawing paths by robots are shown by using turtlesim. Thus, processor usage and processing time on the turtlesim were measured and recorded in the data table. Results were presented in Figure 3, separately. According to results which were shown in Figure 3, central processing unit (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 RTOSs were observed than GPOS versions, in Figure 3, as well.

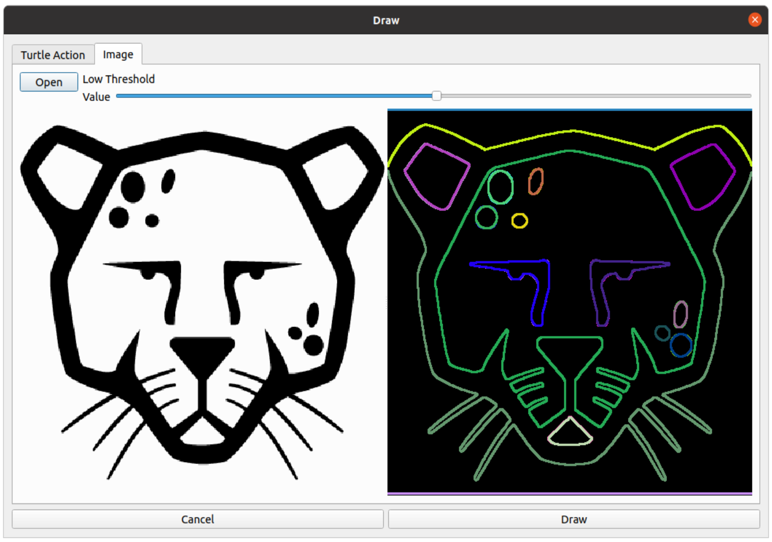


Figure 1. Pardus logo and rqt turtlesim view

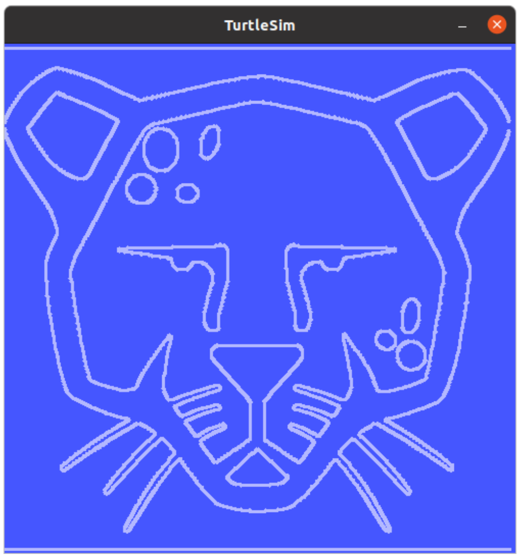
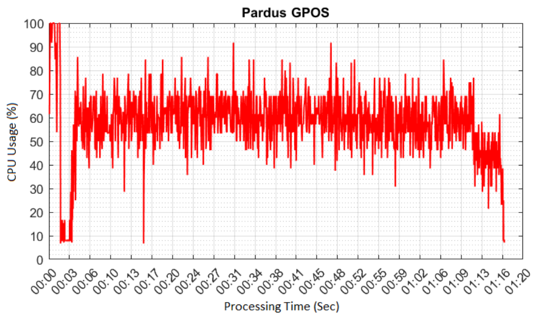
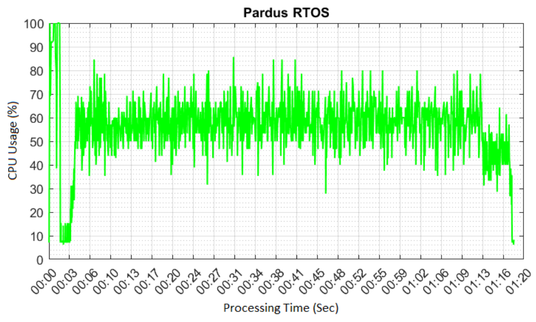
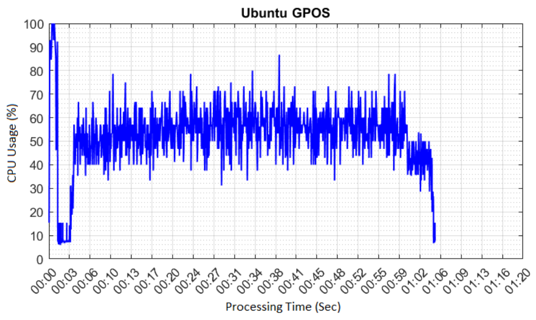


Figure 2. Drawing Padus logo path result drawn using turtlesim







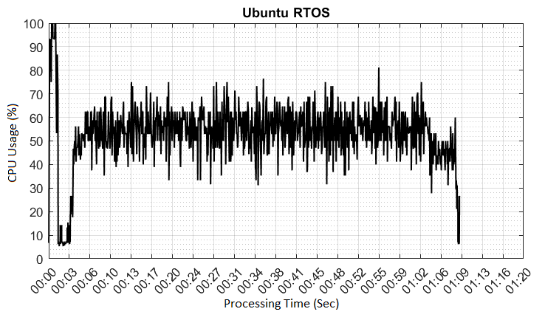


Figure 3. Processor usage graphs for experiment 1

3.2. Experiment 2 draw a path using Ubuntu-Linux logo

In this experiment, Ubuntu-Linux logo was drawn as simple path planning using [44]. Pardus GPOS-RTOS and Ubuntu GPOS-RTOS operating systems were used with the same parameters. The results were recorded, and performance comparison was performed. In Figure 4, Ubuntu-Linux official logo and selected paths were shown, separately. The code was run with its default settings without changing and the same shape was drawn in 4 different operating systems. In Figure 5, drawing paths by robots are shown by using turtlesim. Thus, processor usage and processing time on the turtlesim were measured and recorded 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 RTOSs were observed than GPOS versions, in Figure 6. Low CPU 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.

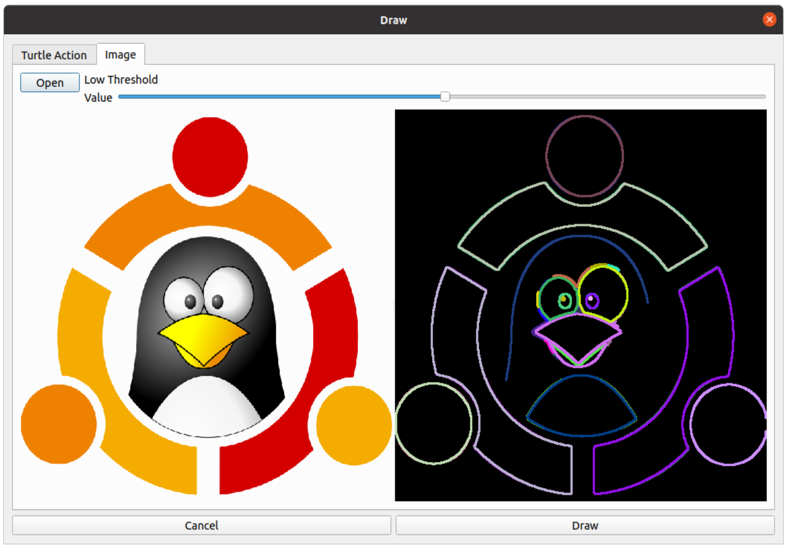


Figure 4. Ubuntu-Linux logo and rqt turtlesim view

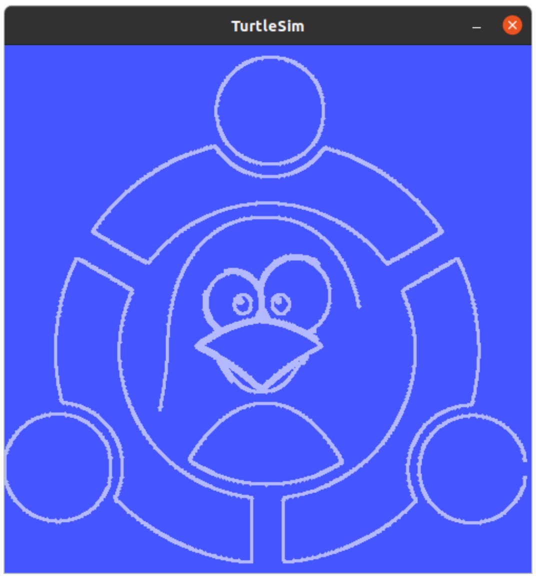


Figure 5. Drawing Ubuntu-Linux logo path result drawn using turtlesim

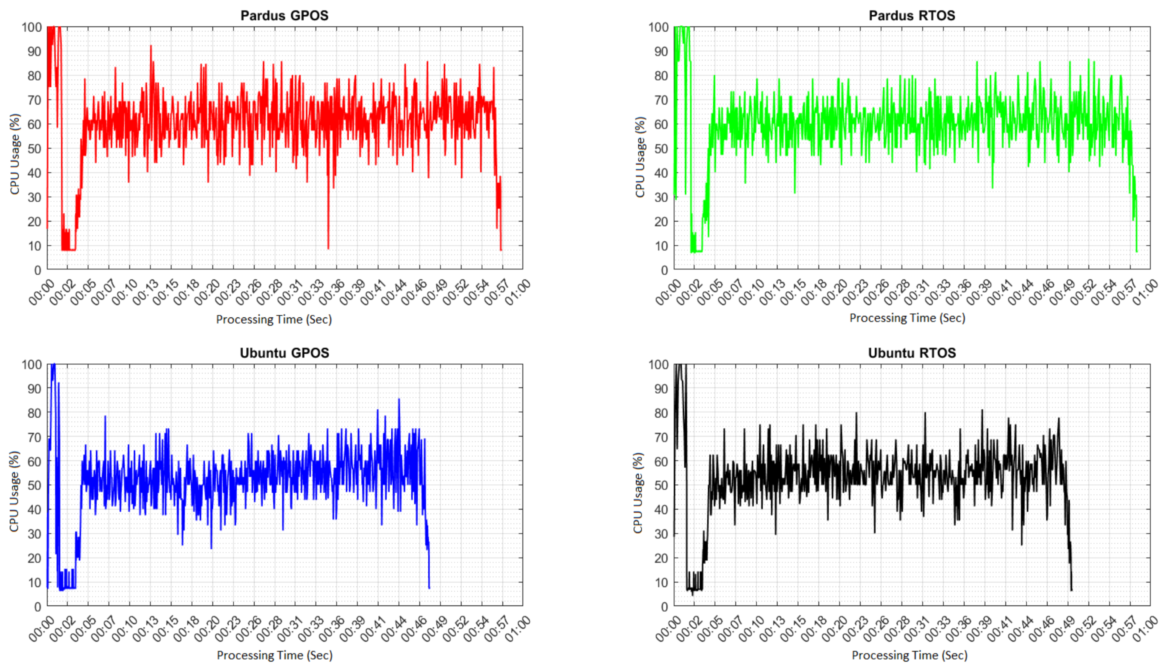


Figure 6. Processor usage graphs for experiment 2

4. RESULTS AND DISCUSSION

In experiment 1 and 2, complex trajectory tracking and formation control for multi-robot systems were performed by using [44] on all operating systems. The processing times for each operating system are presented in Table 1. As can be seen from the results in Table 1, in the comparison of RTOS and GPOS, GPOSs give better results than RTOS in processing time. The main reason for this is while drawing a picture in GPOS, when each robot's task is finished during drawing, the operating system reads the value and the process of the next robot begins. But before beginning of the process in RTOS, the beginning and ending times of the process is determined. If the processing time is set as 100 milliseconds for each process, even if robot completes its task before 100 milliseconds, the resources will be available for the other robot at the end of the 100 milliseconds. Although, this situation increases system security and reduces the margin of error, it causes delays in processing times as it extends the processing time.

Table 1. The processing times for each operating system (milliseconds)

Experiment	Ubuntu		Pardus	
	GPOS	RTOS	GPOS	RTOS
1	61.890	68.940	76.700	78000
2	48.130	49.960	57.160	58.210

As a result of this application, there are two results that can be observed clearly in the performance evaluation of multi-robots in path planning and formation control. The first one is, GPOSs completed their task earlier than RTOSs in two different operating systems. The second one is, Ubuntu GPOS and RTOS completed processes faster than Pardus with limited resources. At the same time, it has been observed that all operating systems completed their tasks without any problems.

5. CONCLUSION

In this study, open-source operating systems, Ubuntu and Pardus, were used as operating systems in order to freely modify in the core codes and structure. The main aim of this study is to observe and to compare usability of RTOS and GPOS in multi-robot systems. RTOSs were completed tasks later than the GPOSs as expected. Because, even if the processes were completed, the operating system read the value at the end of the exact processing time determined for the processes. The results vary due to the structural features of the operating systems. When choosing an operating system for the multi-robot applications, features such as possible failure conditions, process security, processing time speeds should be taken into consideration. A defined process is terminated or completed at certain time when a process run on RTOS. This is very important feature, if robot is used in time dependency applications. Since the processing times are clearly defined, that task is canceled in the delay that occurs in a task and the whole system is not affected by an error that occurs and the system remains alive. GPOS and RTOS were compared for path planning and formation control of multi-robot systems using two different Linux distributions. According to the experimental results, it was seen that GPOS concluded all defined task faster than RTOS. GPOS can be preferred in a way that performance is in the foreground in studies to be carried out in the field of multi-robotic systems. If the task has time dependency or the task requires certain processing time, RTOSs must be used, absolutely.

ACKNOWLEDGEMENTS

This study was supported by the Istanbul Sabahattin Zaim University Scientific Research Program [grant number: BAP-1000-62, 2021].

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," *Networks Parallel 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] A. Murikipudi, V. Prakash, and T. Vigneswaran, “Performance analysis of real time operating system with 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] C. 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.

[12] H. Wei, Z. Huang, Q. Yu, M. Liu, Y. Guan, and J. Tan, "RGMP-ROS: A real-time ROS architecture of hybrid RTOS and GPOS on multi-core processor," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 2482-2487, doi: 10.1109/ICRA.2014.6907205.

[13] H. Wei *et al.*, "RT-ROS: A real-time ROS architecture on multi-core processors," *Future Generation Computer Systems*, vol. 56, pp. 171-178, 2016, doi: 10.1016/j.future.2015.05.008.

[14] C. 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," *SAE Technical Paper*, vol. 1, 2014, doi: 10.4271/2014-01-0200.

[15] S. Zouaoui, L. Boussaid, and A. Mtibaa, "Priority based round robin (PBRR) CPU scheduling algorithm," *International Journal of Electrical and Computer Engineering (IJECE),* vol. 9, no. 1, 2019, doi: 10.11591/ijece.v9i1.pp190-202.

[16] Nandana V., Jithendran A. and Shreelekshmi R., "Survey on RTOS: Evolution, types and current research," *International Journal of Computer Applications*, vol. 121, no. 21, pp. 28-31, 2015, doi: 10.5120/21825-5077.

[17] A. Murikipudi, V. Prakash, and T. Vigneswaran, "Performance analysis of real time operating system with 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.

[18] L. Das, S. Mohapatra, and D. P. Mohapatra, "Schedulability of rate monotonic algorithm using improved time demand analysis for multiprocessor environment," *International Journal of Electrical and Computer Engineering*, vol. 8, no. 1, pp. 429-440, 2018, doi: 10.11591/ijece.v8i1.pp429-440.

[19] P. Dong, Z. Jiang, A. Burns, Y. Ding, and J. Ma, "Build real-time communication for hybrid dual-OS system," *Journal of Systems Architecture,* vol. 107, 2020, doi: 10.1016/j.sysarc.2020.101774.

[20] T. H. Lin, Y. Kinebuchi, and T. Nakajima, "Robust lightweight embedded virtualization layer design with simple hardware assistance," in *IEICE Transactions on Information and Systems*, 2012, vol. E95-D, no. 12, pp. 2821-2832, doi: 10.1587/transinf.E95.D.2821.

[21] B. S. Kim, H. S. Park, K. H. Kim, D. Godfrey, and K. Il Kim, "A survey on real-time communications in wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 2017, 2017, Art. no. 1864847, doi: 10.1155/2017/1864847.

[22] U. C. Devi, "Soft real-time scheduling on multiprocessors," Ph.D. dissertation Department of Computer Science, University of North Carolina, Chapel Hill, North Carolina, 2006.

[23] M. Barabanov, "A linux-based real-time operating system," M.S. thesis, New Mexico Institute of Mining and Technology Socorro New Mexico 1997. [Online]. Available: http://luz.cs.nmt.edu/~rtlinux.Acknowledgements (accessed Mar. 27, 2021).

[24] A. Damm, J. Reisinger, W. Schwabl, and H. Kopetz, "Real-time operating system of MARS," *ACM SIGOBS Operating Systems Review,* vol. 23, no. 3, pp. 141-157, 1989, doi: 10.1145/71021.71029.

[25] J. Kacur, "Real - time kernel for audio and visual applications," *Linux Audio Conf.*, 2010.

[26] F. Reghenzani, G. Massari, and W. Fornaciari, "The real-time linux kernel: A survey on PREEMPT_RT," *ACM Computing Surveys*, vol. 52, no. 1, pp. 1-36, 2019, Art. no. 18, doi: 10.1145/3297714.

[27] S. K. Das, A. K. Dutta, and S. K. Debnath, "OperativeCriticalPointBug algorithm-local path planning of mobile robot avoiding obstacles," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 18, no. 3, pp. 1646-1656, 2020, doi: 10.11591/ijeecs.v18.i3.pp1646-1656.

[28] A. M. Azri, S. Abdul-Rahman, R. Hamzah, Z. A. Aziz, and N. A. Bakar, "Visual analytics of 3D LiDAR point clouds in robotics operating systems," *Bulletin of Electrical Engineering and Infromatics (BEEI)*, vol. 9, no. 2, pp. 492-499, 2020, doi: 10.11591/eei.v9i2.2061.

[29] Y. Li, Y. Chen, Y. Yang, and Y. Li, "Soft robotic grippers based on particle transmission," in *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 3, pp. 969-978, June 2019, doi: 10.1109/TMECH.2019.2907045.

[30] B. Rahmani, A. Harjoko, and T. K. Priyambodo, "A vision-based real-time obstacle avoidance's rules utilising grid-edge-depth map," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 19, no. 1, pp. 513-525, 2020, doi: 10.11591/ijeecs.v19.i1.pp513-525.

[31] A. S. Handayani, S. Nurmaini, I. Yani, and N. L. Husni, "Analysis on swarm robot coordination using fuzzy logic," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 13, no. 1, pp. 48–57, 2019, doi: 10.11591/ijeecs.v13.i1.pp48-57.

[32] R. I. Mutia, "Inter-Process Communication Mechanism in Monolithic Kernel and Microkernel," unpublished, 2014. [Online]. Available: http://www.eit.lth.se/fileadmin/eit/project/142/IPC_Report.pdf.

[33] D. Du, Z. Hua, Y. Xia, B. Zang, and H. Chen, "XPC: Architectural Support for secure and efficient cross process call," *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*, 2019, pp. 671-684.

[34] X. Peng, K. Xiao, Y. Li, L. Chen, and W. Zhang, "Fast interprocess communication algorithm in microkernel," *International Journal of Perfomability Engineering*, vol. 16, no. 2, pp. 185-194, 2020, doi: 10.23940/ijpe.20.02.p3.185194.

[35] C. Main, "Virtualization on multicore for industrial real-time operating systems [from mind to market]," in *IEEE Industrial Electronics Magazine*, vol. 4, no. 3, pp. 4-6, Sep. 2010, doi: 10.1109/MIE.2010.937935.

[36] R. Petersen, "Ubuntu 20.04 LTS Desktop: Applications and administration," Surfing Turtle Press, Jun. 2020.

[37] M. M. Karakoç and A. Varol, "National distribution project and pardus operating system," *Turkish Journal of Science & Technolog*, vol. 11, no. 2, pp. 25-34, 2016.

[38] M. Quigley *et al.*, "ROS: an open-source robot operating system," *ICRA Workshop on Open Source Software*, 2009. [Online]. Available: http://stair.stanford.edu (accessed Mar. 27, 2021).

[39] J. M. Cañas, E. Perdices, L. García-Pérez, and J. Fernández-Conde, "A ROS-based open tool for intelligent robotics education," *Applied Sciences*, vol. 10, no. 21, pp. 1-20, 2020, doi: 10.3390/app10217419.

[40] F. Ellouze, A. Koubâa, and H. Youssef, "ROSWeb services: A tutorial," *Studies in Computational Intellegence,* vol. 625, pp. 263-490, 2016, doi: 10.1007/978-3-319-26054-9_18.

[41] D. R. Sulaiman, "Multi-objective Pareto front and particle swarm optimization algorithms for power dissipation reduction in microprocessors," *International Journal of Electrical and Computer Engineering (IJECE),* vol. 10, no. 6, pp. 6549-6557, 2020, doi: 10.11591/IJECE.V10I6.PP6549-6557.

[42] S. H. Abdulredah and D. J. Kadhim, "Developing a real time navigation for the mobile robots at unknown environments," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 20, no. 1, pp. 500-509, Oct. 2020, doi: 10.11591/ijeecs.v20.i1.pp500-509.

[43] S. K. Das, A. K. Dutta, and S. K. Debnath, "Development of path planning algorithm of centipede inspired wheeled robot in presence of static and moving obstacles using modified critical-snakebug algorithm," *IAES Int. J. Artif. Intell.*, vol. 8, no. 2, pp. 95–106, 2019, doi: 10.11591/ijai.v8.i2.pp95-106.

[44] Franz. "Rqt plugin for ROS (Noetic) to control turtles in turtlesim." GitHub.com. https://github.com/fjp/rqt-turtle (accessed Mar. 01, 2021).

[45] F. Pusher. “Rqt plugin License.” GitHub.com. https://github.com/fjp/rqt-turtle/blob/master/LICENSE (accessed Mar. 01, 2021).

BIOGRAPHIES OF AUTHORS

Seçkin Canbaz received his B.Sc. degree form Istanbul University, Faculty of Science, Department of Physics in 2015. He started to work at Istanbul Sabahattin Zaim University in the Department of Information Technologies in 2016. He is currently master student at Istanbul Sabahattin Zaim University, Department of Computer Science and Engineering. His research interests are open-source operating systems, real-time operating systems and ROS. He can be contacted at email: seckin.canbaz@izu.edu.tr.

Gökhan Erdemir received his B.Sc., M.Sc. and Ph.D. degrees from Marmara University, Turkey, respectively. During his Ph.D., we worked as a research scholar at Michigan State University, Department of Electrical and Computer in East Lansing MI, USA. Now, he is an assistant professor at Istanbul Sabahattin Zaim University, Department of Electrical and Electronics Engineering. His research topics include robotics, control systems, and intelligent algorithms. He can be contacted at email: gokhan.erdemir@izu.edu.tr.