The Impact of Using Real-Time Operating System (RTOS) in AI-Integrated Autonomous Vehicles on Efficiency

Şeyma DEĞİRMENCİ[1], Muhammed TELÇEKEN[2]*
[1] Mechatronics Engineering, Sakarya University of Applied Sciences, Turkey
[2] Computer Engineering, Sakarya University of Applied Sciences, Turkey
ORCID: 0009-0000-1501-5086, 0000-0001-5223-2856
E-mails: b200104021@subu.edu.tr, muhammedtelceken@subu.edu.tr
*Corresponding author.

***Abstract*— This study examines the impact of Artificial Intelligence (AI) and Real-Time Operating Systems (RTOS) on the efficiency and performance of autonomous vehicles. Autonomous vehicles operate with high precision using environmental sensors and decision-making algorithms, where the deterministic structure and low latency of RTOS are critical for timely and safe operation. The study discusses how different RTOS types Hard RTOS, Soft RTOS, and Firm RTOS affect tasks such as safety, data analysis, and route planning in autonomous vehicles, highlighting the advantages of each type. The integration of AI and RTOS enables efficient utilization of system resources, allowing autonomous vehicles to respond more quickly to environmental data and ensuring the effective activation of safety measures. Graphical analyses and experimental results demonstrate the positive effects of this integration on CPU usage, memory consumption, and system response times. These findings illustrate that AI and RTOS significantly enhance the efficiency of autonomous vehicles, contributing substantially to safety and performance.**

***Keywords*— Real-Time Operating System, AI and RTOS Integration, Autonomous Vehicles**

I. INTRODUCTION

Artificial Intelligence (AI) technology has significant potential to solve complex problems and develop intelligent systems with its ability to model human mental processes [1]. Artificial Intelligence has significant potential for engineering and technological solutions with its optimization, adaptability, and learning capacity capabilities in various applications such as industrial automation, robotics, and autonomous systems [2]. However, the success of these systems is directly related not only to the accuracy of AI algorithms but also to the compliance of the results produced with temporal requirements. In time-critical systems, the completion of a task within the specified time stands out as one of the most important factors determining the functionality of the system [3]. In this context, Real-Time Operating Systems (RTOS) provide an important infrastructure by meeting the accuracy and timely completion requirements for artificial intelligence-based applications [4]. RTOSs play an important role in the integration of AI into industrial and autonomous systems with their capabilities, such as prioritization of system resources, synchronization of tasks, and efficient management of embedded hardware. In particular, the stability and reliability of processes such as processing data from sensors, real-time decision-making, and action execution can be optimized thanks to the RTOS infrastructure [5].

Real-time operating systems (RTOS) guarantee that tasks are performed uninterruptedly and correctly in certain time periods. RTOSs ensure the system operates in a deterministic structure by completing operations within the specified time limits. RTOSs are generally divided into three different categories: hard, soft, and firm. These systems increase the reliability of autonomous vehicles by ensuring that critical tasks such as security, data analysis, and route planning are performed without delay. Since each application has its own unique requirements, choosing the right RTOS is a very critical and time-consuming process for designers. Developers need to consider many parameters during RTOS selection. These features include interrupt latency, context switching time, communication mechanisms, memory management, power management, and task management. This process creates a multi-dimensional search problem and requires finding the most appropriate solution in a large search space. In order to overcome these problems, some methods use the Genetic Algorithm (GA) in selecting the right RTOS [6]. This method offers an optimized approach for RTOS selection by considering the parameters. GA determines the most suitable RTOS in a large design space using operations such as selection, crossover, and mutation inspired by biological evolution. In addition, an interactive graphical interface has been developed to make this process more user-friendly. This system allows designers to quickly and efficiently select the RTOS that suits their needs, thus saving time and achieving better performance in the development process [6].

Autonomous vehicles can operate in complex driving scenarios without the need for human intervention through the combination of environmental sensors, AI algorithms, and RTOS. These vehicles process large amounts of data from the environment to perform tasks such as recognizing traffic signs,

avoiding obstacles, controlling speed, and managing emergencies. When the ability of AI to make rapid decisions based on environmental data is combined with the low latency and reliable performance of RTOS, both the efficiency and safety of autonomous vehicles are significantly increased.

In this study, the effects of AI and RTOS integration on the efficiency of autonomous vehicles will be examined. In addition, the areas of use, advantages, and limitations of different types of RTOS in autonomous vehicles will be discussed. Through the research conducted, the effects of systems in which these two technologies are integrated on CPU usage, memory consumption, and system response times will be evaluated. This way, the contributions of RTOS usage in AI-integrated autonomous vehicles to efficiency and performance will be examined.

II. RELATED WORKS

Studies are being carried out in different areas on RTOS systems with artificial intelligence integration. In this study, we will examine the use of two different systems. One of these will be AI-supported RTOS systems for autonomous vehicles, and the other will be studies on embedded systems.

A. RTOS in Autonomous Vehicles

Studies are being carried out on the development of autonomous vehicle technologies. Some of these studies are as follows. Wang et al. [7] examined the current status, performance, and safety challenges of Autonomous Vehicle (AV) technology in detail. In their study, they used road test data from California between 2014 and 2018. In the tests, AVs traveled a total of 3.7 million miles, and 159,870 manual intervention events were reported during this period. Manual interventions were generally caused by complex driving conditions and varied by manufacturer. The study emphasized that interventions and accidents were mainly due to perception, decision-making, and action errors. Of the 128 reported accidents, 63.3% occurred in autonomous driving mode, but only 6.3% were directly caused by AVs. They observed that 93.7% of the other accidents were caused by external factors such as pedestrians, cyclists, motorcycles, and other vehicles. In their analysis, they emphasized that passive accident avoidance strategies of AVs should be developed. They also noted that differences in definitions of “intervention” across manufacturers limited the full comparability of reported events. Ma et al. [8] examined the impact of AI technologies on AVs and the role of AI in the development of these vehicles. They examined existing research to understand how AI is used for its basic functions, such as environmental perception and accurate decision-making. The main goal of their study is to analyze existing approaches to the application of AI in AVs and to reveal the challenges and opportunities encountered in achieving the goal of full autonomy. In this context, they examined the use of AI in three main application areas in AVs, namely perception, localization, and mapping. In addition, according to the results of this analysis, they focused on the potential areas where AI can be used together with other new technologies. Verma et al. [9] studied the effects of AI and RTOS integration on industrial robots. In their study, they worked on how to combine the advantages of RTOS, such as time cycle accuracy, synchronization, and performance improvement, with artificial intelligence-based data analysis and decision-making mechanisms. They observed that AI and RTOS integration achieved positive results on the performance of industrial robots. By using an advanced RTOS such as LynxOS together with artificial intelligence algorithms, they optimized the system's processing time, CPU usage, and memory consumption. In their experimental studies, they observed that six-axis industrial robots could analyze environmental data in real-time while following a specific path plan, and their motion accuracy was increased. In terms of performance, they improved the system's response time and accelerated decision-making processes. Jinglu and Lina [10] used the IEC61850 protocol in their work to combine various communication protocols of devices from different manufacturers and to ensure the unification of the information model and communication protocol. In this system, they used specially designed and built-in integrated control technologies to increase interoperability between devices. They developed a ZigBee-based information processing terminal using ARM Cortex-M0 and GPRS technologies. They optimized the application development and data processing processes with this structure they proposed. In the virtualization test carried out for 10 minutes, they obtained the packet loss rate in the RTOS service system with their proposed method as 0.396%. In a 120-minute simulation, the packet loss rate of this method was measured at 0.343%. The study results show that the system designed with this method attracts attention due to its lower time cost, higher stability, and lower packet loss rate. In addition, they showed that the cloud computing-based virtualized artificial intelligence RTOS service system is better than existing methods in terms of automatic management and processing capacity. Rahul et al. [11] used RTOS to perform the motion control of a 4-axis parallel robot arm in real-time and synchronously. They generated actuation signals by solving the kinematic equations with RTOS in a dual-core microcontroller. In this way, they managed the calculation of position coordinates and the generation of signals simultaneously. They evaluated the feedback from the limit switches with the G-code-based motion planning they wrote with RTOS. In this way, they managed to operate the synchronous motion of the robot arm reliably and at high speed. They state that this system is quite efficient for autonomous ground vehicles and unmanned aerial vehicles. In his study, Altan [12] examined the use of RTOS in mini unmanned aerial vehicles (MUAS). RTOS offered an effective solution by addressing problems such as processing load, hardware/software limitations, and resource constraints in the flight control systems of MUAS. He criticized the traditional standards (DO-178B) in the development of MUAS. He stated that these standards do not apply to the development of small systems and are far from practical and time-consuming. Yaşar et al. [13] worked on autonomous mobile robot navigation in structured rough terrain. 2D simultaneous localization and mapping, 3D mapping, path planning and path

following with respect to the 3D map are implemented on Robot Operating System (ROS) in this work.

B. RTOS in Embedded Systems

The areas of use of embedded systems have expanded considerably today, and they have become a fundamental component, especially in autonomous vehicles and robotic systems. Various optimization studies have been carried out to increase performance, reliability, and energy efficiency in these systems. Some of these studies are as follows. Mohammed et al. [14] conducted a comprehensive study addressing the role of RTOS in embedded systems. In the study, they focused on using RTOS in embedded systems and emphasized that these systems are critical in performance and reliability. The basic features of RTOS include deterministic behavior, fast response, and effective task planning, and they stated that these features ensure that tasks in embedded systems are completed on time. They demonstrated the necessity of RTOS, especially in time-sensitive applications such as industrial controllers and medical devices. The ability of RTOS to isolate tasks in the system and meet strict timing requirements was shown among the unique features of these systems. Sindhwani et al. [15] discussed various acceleration techniques to reduce the load on the CPU of RTOS used in embedded systems. In this context, they discussed four main acceleration techniques used to lighten RTOS loads in the study. First, they mentioned co-processor-assisted acceleration. A co-processor aims to reduce the load on the CPU by delegating some RTOS tasks to an external microcontroller. The second is hardware-assisted acceleration. This method aims to reduce the CPU load by implementing RTOS functions at the hardware level. Third, hardware RTOS aims to eliminate the loads introduced by software-based RTOS by implementing the entire RTOS in hardware. Finally, instruction set customization aims to shorten the processing time by implementing critical RTOS functions in hardware with special instructions. Their study emphasized that these techniques are generally implemented using hardware such as advanced processors, FPGA space, and configurable System-on-Chip (SoC). They accelerated the functions of RTOS, such as scheduler, task management, and event control, by using instruction set customization on software-based processors such as Altera NIOS and OpenRISC and significantly reduced the load on the CPU. Hassel et al. [16] used RTOS as an abstract model to design real-time embedded systems. With this model, they studied the modeling of parallel and concurrent behaviors of designers by representing the typical features of RTOS at a high level of abstraction. They created an abstract RTOS model using the SystemC language during system design. Then, they refined this model to lower levels of abstraction and transformed it into an application model. They applied this approach to a telecom system consisting of 50 tasks and four priority levels, allowing different scheduling strategies to be tested quickly. He et al. [17] modeled RTOS as an abstract state machine based on SystemC and designed a model that can be configured to represent different RTOS structures. They created the model they designed from components including task management, operating system kernel, input/output module, timing, and event synchronization protocols. With this system, they simulated the behavior of tasks, interrupts, and timing with a model representing seven predetermined basic states. In this way, they enabled the modeling and testing different RTOS policies in the early design stages. They scheduled the tasks using different scheduling policies, such as priority-based or FIFO/round-robin. Since dynamic threading is not supported, they created a static thread pool before the simulation and assigned the threads in this pool to specific tasks. They used this model to accelerate the design and evaluation process of real-time embedded systems by user-configuring it to represent different RTOS architectures.

III. MAETRIALS & METHODS

A. Artificial Intelligence

Artificial intelligence (AI) is a branch of science that aims to imitate certain aspects of human intelligence. This field aims to improve the thinking, learning, problem-solving, and decision-making abilities of computers. Sub-disciplines of AI include machine learning (ML), deep learning (DL), natural language processing (NLP), and computer vision (CV) [18]. At the core of AI are mathematical models, algorithms, and data. Today, AI has many application areas, such as healthcare, autonomous systems, and natural language processing. Autonomous vehicles, unmanned aerial vehicles (UAV), and robotic systems use AI's control mechanisms. These systems process data from sensors to make real-time decisions [19]. Autonomous systems are one of the most striking application areas of AI. These systems are technologies that can make independent decisions by processing data from the environment and performing their tasks without human intervention. Autonomous systems can be divided into subcategories as follows:

Autonomous Ground Vehicles: Today, many automobile manufacturers are developing autonomous driving technologies. These vehicles analyze the environmental situation and make driving decisions by combining data from sensors such as radar, lidar, camera, and GPS [20].

Unmanned Aerial Vehicles (UAV): UAVs are used especially in military and civilian areas. AI algorithms make these vehicles effective in tasks such as air traffic management, route planning, and object detection [21].

Robotic Systems: While industrial robots can move autonomously in production processes, domestic robots help with tasks such as cleaning and maintenance [22,23].

The development of autonomous systems brings the following advantages:

Increased efficiency: Speeds up operational processes by minimizing human errors [19].

Cost reduction: Reduces labor costs, especially in the manufacturing and logistics sectors [20].

Safety: Increases occupational safety by replacing humans in dangerous tasks [18].

However, the spread of autonomous systems also raises ethical and legal questions. For example, questions such as who is responsible for autonomous vehicle accidents are still a matter of debate [24]. The visual of the general structure of artificial intelligence is given in Figure 1.

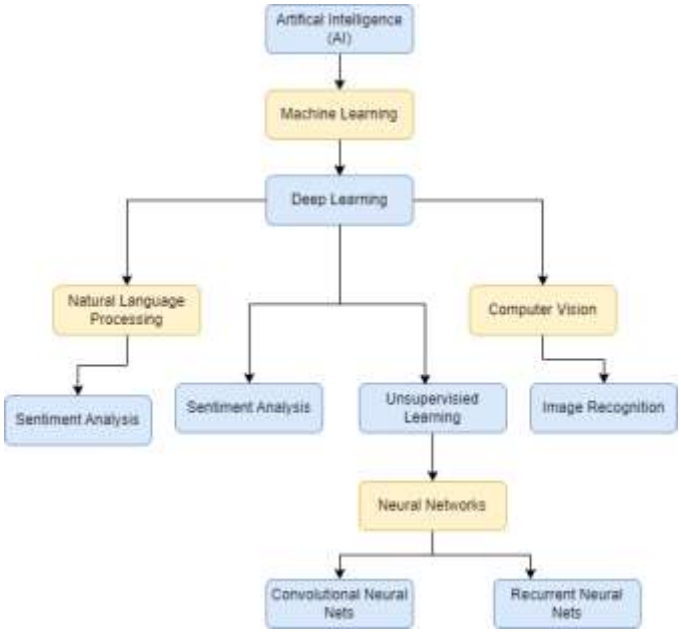

Figure 1. Artificial Intelligence Diagram

B. Real-Time Operation System (RTOS)

A real-time operating system (RTOS) is a type of operating system that guarantees the completion of tasks within a certain period. RTOSs are widely used in time-critical applications, such as industrial automation, medical devices, multi-sensor systems, and the defense industry [5]. RTOS is a system in which tasks are managed according to their priorities, sensitivity to interruptions is emphasized, and resources are used effectively. The main goal of these systems is to work within the framework of specified deadlines [25].

The essential components of an RTOS are:

Real-Time Scheduler: Responsible for scheduling various tasks according to their priority levels and operating conditions.

Interrupt Management: RTOS effectively manages the situations in which the hardware sends signals to the operating system.

Resource Management: Provides efficient use of system resources such as CPU, memory, and output units.

Concurrency Mechanisms: Provides structures such as locks and semaphores to ensure data sharing and synchronization between tasks.

RTOS uses various algorithms to determine the order and timing of tasks to run [26].

- First Come First Serve (FCFS) Algorithm:

FCFS is a basic scheduling algorithm in which the order of processes is determined by the order in which they enter the system. Processes are sorted using the FIFO (First In First Out) principle, and the first process that arrives is processed [27].

Arrival Time (AT): The time the process enters the system.

Burst Time (BT): The time it takes for the process to process on the CPU.

Completion Time (CT): The time at which the process's processing time is completed is calculated as seen in Equation 1.

$$CT_i = \max(CT_{i-1}, AT_i) + BT_i \quad (1)$$

Turnaround Time (TAT): The total time spent by the process throughout the process is calculated as seen in Equation 2.

$$TAT_i = CT_i - AT_i \quad (2)$$

Waiting Time (WT): The time the process waits to be processed on the CPU is calculated as seen in Equation 3.

$$WT_i = TAT_i - BT_i \quad (3)$$

Response Time (RT): The time it takes for the process to first access the CPU is calculated as seen in Equation 4.

$$RT_i = WT_i \quad (4)$$

- Priority Scheduling Algorithm:

Priority Scheduling is a scheduling algorithm in which each process is associated with a priority value and the CPU processes processes according to this priority value. Processes with higher (or lower, as per system definition) priority are processed first. It can be implemented as Preemptive or Non-Preemptive based on the priority value [28].

Priority (P): The value that determines the order in which each process is processed by the CPU. A lower value usually indicates a higher priority (this depends on the system definition).

Arrival Time (AT): The time at which the process enters the system.

Burst Time (BT): The time it takes for the process to process on the CPU is calculated as in Equation 5.

$$CT = Start\ Time + BT \quad (5)$$

Turnaround Time (TAT): The total time the process spends in the entire process is calculated as in Equation 6.

$$TAT = CT - AT \quad (6)$$

Waiting Time (WT): The time the process waits to be processed on the CPU as seen in Equation 7.

$$WT = TAT - BT \quad (7)$$

- Round Robin (RR) Algorithm:

Round Robin (RR) is a scheduling algorithm that is widely used in time-sharing systems and allocates equal time to each process. This algorithm aims to distribute CPU resources to all processes in a fair and simple manner. Processes are placed in a queue (FIFO - First In First Out) and each process is processed in order. If a process cannot complete in its allocated time slot, the process is added to the end of the queue with its remaining time [29].

Arrival Time (AT): The time the process enters the system.

Burst Time (BT): The total time it takes for the process to process on the CPU.

Time Quantum (TQ): The processing time (time slice) allocated to each process.

Completion Time (CT): The time at which the process's processing time is completed.

Turnaround Time (TAT): It is calculated as seen in Equation 8.

$$TAT = CT - AT \tag{8}$$

Waiting Time (WT): It is calculated as seen in Equation 9.

$$WT = TAT - BT \tag{9}$$

- Shortest Job Next (SJN) Algorithm:

Shortest Job Next (SJN), also known as Shortest Job First (SJF), is a scheduling algorithm that allocates CPU according to the processing time required for each process. The process with the shortest processing time is run first. SJN can be implemented as Non-Preemptive or Preemptive. The Preemptive version is called Shortest Remaining Time First (SRTF) [30].

Arrival Time (AT): The time at which the process enters the system.

Burst Time (BT): The time it takes for the process to process on the CPU is calculated as in Equation 10.

$$CT = Start\ Time + BT \tag{10}$$

Turnaround Time (TAT): The total time the process spends in the entire process is calculated as in Equation 11.

$$TAT = CT - AT \tag{11}$$

Waiting Time (WT): The time the process waits to be processed on the CPU as seen in Equation 12.

$$WT = TAT - BT \tag{12}$$

The design and implementation of RTOS depends on factors such as complex scheduling algorithms and dynamic prioritization of tasks. Therefore, mathematical models and equations play an important role in the development of RTOS systems.

IV. RESULTS AND DISCUSSION

This study examined the RTOS and artificial intelligence integrated systems used in embedded systems and autonomous vehicles. In this direction, we will examine the results of the obtained data. Jinglu and Linna [5], from the analyses they obtained as a result of their studies, observed that the method they used to virtualize the intelligent information visualization artificial intelligence RTOS service system in the cloud computing environment had less time cost and higher stability and improved the automatic management ability of the intelligent information visualization RTOS system. The results they obtained are given in Table 1.

Table 1: Test analysis results of the proposed method

Similasyon süresi/dk	Jinglu ve Linna methodu
10	0,396
40	0,343
80	0,347
120	0,343

Verma et al. [9] explained the various application areas and usage of AI and reviewed how AI-based RTOS can be integrated into industrial robots and manipulators to increase the efficiency of the system. The details of their review are shown in Table 2.

Based on these studies, it was seen in the research that RTOS has time-constrained methods, and tests were conducted on how these methods can give results. Time constraint methods are Preemptive Scheduling, Non-preemptive Scheduling, Rate-monotonic Scheduling (RMS), and Earliest Deadline First (EDF). In the preemptive scheduling method, tasks are ranked according to their priorities, and higher-priority tasks can be interrupted even while a lower-priority task is running. Non-preemptive scheduling or cooperative scheduling does not leave the processor until a task is completed. In this case, the order of the tasks continues in the order they were started, and they leave the processor only when the tasks decide to terminate themselves. Rate-monotonic scheduling is a common algorithm for scheduling periodic tasks. This method ranks shorter-period tasks as having higher priority. In other words, more frequently repeated tasks are run first. Earliest Deadline First (EDF) ensures that each task is ranked according to its closest finish time. That is, the task that needs to be completed first is run first. EDF is a dynamic method because the due dates of tasks may change over time. Based on these methods, the graph of time constraint methods defined in the Python programming language and task completion rates is shown in Figure 2.

In this context, as examined in the graphical output, it was seen that Rate-monotonic scheduling, the method used for scheduling periodic tasks, gave more efficient results.

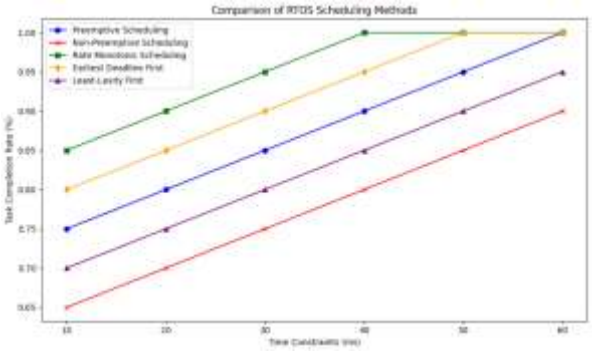

Figure 2: Time Blocking Methods and Task Completion Rate Chart

I. CONCLUSION

According to the results obtained from the studies, we examined the integration of AI and RTOS, which provides significant improvements in systems that require time sensitivity, such as autonomous vehicles and industrial robots. RTOSs allow autonomous systems to process environmental data quickly and accurately, ensuring that tasks are performed precisely in certain time periods. This allows autonomous vehicles to respond quickly to environmental changes and provide a safer driving experience. AI algorithms, combined with the RTOS infrastructure, enable autonomous systems to not only process data quickly but also to draw meaningful conclusions from this data.

Table 2: Artificial intelligence-based RTOS application structures

Title	**Explanation**	**The Study Conducted**	**Results**
Real Time Operating System	Time-sensitive operating system solutions for industrial robots.	Precise control of industrial robots has been achieved using LynxOS and other RTOSs.	RTOSs have improved time accuracy with high-performance processors and multi-core systems.
Trajectory Planning and Manipulators	Optimization of robot movements in terms of time and accuracy.	The speed and accuracy of the robots have been optimized using AI-supported RTOS.	Thanks to trajectory planning, the robots' movement speed has been increased and their accuracy has been improved.
Communication Control Unit	AI-powered controllers to optimize robot control.	Industrial robots are optimized using AI-based control programs.	With CCU, robot control processes have been improved and time cycles have been reduced.
Fuzzy Logic and Artificial Neural Networks	AI techniques to ensure time accuracy and stability in industrial robots and other systems.	Time accuracy has been increased in robot systems by using fuzzy logic and artificial neural networks.	The use of AI techniques has increased time accuracy in industrial robot systems.

AI's decision-making capabilities are critical in improving safety and reducing the need for human intervention. The integration of RTOS and AI not only improves system performance but also provides advantages in terms of energy efficiency and more effective use of system resources.

In autonomous vehicles, this integration provides more efficient performance by optimizing processing time, memory consumption, and CPU usage. Considering that every millisecond counts in such systems, these improvements provide great benefits in terms of both functionality and safety. In addition, the compatibility of these two technologies paves the way for new application areas. This increases the value of these technologies for industrial applications by providing faster decision-making processes and more reliable operational performance in autonomous vehicles and robotic systems. As a result, the integration of AI and RTOS offers significant advantages in time- and safety-critical applications such as autonomous vehicles. The harmonious operation of these technologies ensures that systems are more efficient, safe, and durable and plays an important role in creating a strong foundation for the future development of autonomous systems. These opportunities include high-resolution maps, big data and high-performance computing systems, simulation platforms supported by augmented reality (AR) and virtual reality (VR), and AVs connected to 5G communication technologies.

REFERENCES

[1] Goodfellow, I. (2016). Deep learning. https://doi.org/10.4258/hir.2016.22.4.351

[2] Russell, S. J., & Norvig, P. (2016). Artificial intelligence: a modern approach. Pearson.

[3] Marwedel, P. (2021). Embedded system design: embedded systems foundations of cyber physical systems, and the internet of things (p. 433). Springer Nature.

[4] Wang, K. C. (2023). Embedded real-time operating systems. In Embedded and Real-Time Operating Systems (pp. 429-503). Cham: Springer International Publishing.

[5] Jinglu, Z., & Linna, W. (2024, March). Research on Ai Technology of Intelligent Information Visualization Based on Rtos System Service. In Journal of Physics: Conference Series (Vol. 2717, No. 1, p. 012018).

[6] Reddy, S. R. (2006). Selection of RTOS for an efficient design of embedded systems. International Journal of Computer Science and Network Security, 6(6), 29-37.

[7] Wang, J., Zhang, L., Huang, Y., & Zhao, J. (2020). Safety of autonomous vehicles. Journal of advanced transportation, 2020(1), 8867757.

[8] Ma, Y., Wang, Z., Yang, H., & Yang, L. (2020). Artificial intelligence applications in the development of autonomous vehicles: A survey. IEEE/CAA Journal of Automatica Sinica, 7(2), 315-329.

[9] Verma, A., Singh, K., & Mishra, A. (2016). Artificial Intelligence Based Online Rtos System Analysis For Six Axis Industrial Manipulation. International Journal of Research in Engineering and Technology,5(6),41-44

[10] Jinglu, Z., & Linna, W. (2024, March). Research on Ai Technology of Intelligent Information Visualization Based on Rtos System Service. In Journal of Physics: Conference Series Vol. 2717, No. 1, p. 012018.

[11] Rahul, K, Raheman H, Paradkar, V. (2020). Design of a 4 DOF parallel robot arm and the firmware implementation on embedded system to transplant pot seedlings, Artificial Intelligence in Agriculture,4, pp. 172-183.

[12] Altan, Ö. (2014). Mini insansız hava sistemleri için güvenli yazılım sistemi geliştirme çerçevesi (Yüksek lisans tezi, Orta Doğu Teknik Üniversitesi).

[13] Yaşar A, Uslu E, Çakmak F, Altuntaş N, Amasyalı MF, Yavuz S. (2018). Autonomous mobile robot navigation in structured rough terrain. Journal of Intelligent Systems with Applications, 2018; 1(1): 67-74.

[14] Mohammad, A., Das, R., Islam, M. A., & Mahjabeen, F. (2023). Real-time Operating Systems (RTOS) for Embedded Systems. Asian Journal of Mechatronics and Electrical Engineering, 2(2), 95-104.

[15] Sindhwani, M., Oliver, T., Maskell, D. L., & Srikanthan, T. (2004, November). Rtos acceleration techniques–review and challenges. In Sixth Real-Time Linux Workshop (p. 131).

[16] Reddy, S. R. (2006). Selection of RTOS for an efficient design of embedded systems. International Journal of Computer Science and Network Security, 6(6), 29-37.

[17] He, Z., Mok, A., & Peng, C. (2005, March). Timed RTOS modeling for embedded system design. In 11th IEEE Real Time and Embedded Technology and Applications Symposium (pp. 448-457). IEEE

[18] Russell, S. J., & Norvig, P. (2016). Artificial intelligence: a modern approach. Pearson.

[19] Thrun, S. (2010). Toward robotic cars. Communications of the ACM, 53(4), 99-106.

[20] Goodfellow, I. (2016). Deep learning.

[21] Kendall, A., & Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision?. Advances in neural information processing systems, 30.

[22] Taş MO, Yayan U, Yavuz HS, Yazıcı A. (2018). Reliability based task allocation analysis for multi-robot systems by using fuzzy logic. Journal of Intelligent Systems with Applications, 1(1): 8-12.

[23] Taş MO, Yayan U, Yavuz HS, Yazıcı A. Reliability based task allocation analysis for multi-robot systems by using fuzzy logic. Akıllı Sistemler ve Uygulamaları Dergisi (Journal of Intelligent Systems with Applications) 2018; 1(1): 8-12.

[24] Lin, P. (2016). Why ethics matters for autonomous cars. Autonomous driving: Technical, legal and social aspects, 69-85.

[25] Mall, R. (2009). Real-time systems: theory and practice. Pearson Education India.

[26] Ramamritham, K., & Stankovic, J. A. (1994). Scheduling algorithms and operating systems support for real-time systems. Proceedings of the IEEE, 82(1), 55-67.

[27] Schwiegelshohn, U., & Yahyapour, R. (1998, January). Analysis of first-come-first-serve parallel job scheduling. In SODA, Vol. 98, pp. 629-638.

[28] Andersson, B., Baruah, S., & Jonsson, J. (2001, December). Static-priority scheduling on multiprocessors. In Proceedings 22nd IEEE Real-Time Systems Symposium (Cat. No. 01PR1420) (pp. 193-202). IEEE.

[29] Singh, A., Goyal, P., & Batra, S. (2010). An optimized round robin scheduling algorithm for CPU scheduling. International Journal on Computer Science and Engineering, 2(07), 2383-2385.

[30] Damuut, L. P., & Dung, P. B. (2019). Comparative Analysis of FCFS, SJN & RR Job Scheduling Algorithms. International Journal of Computer Science & Information Technology (IJCSIT) Volume, 11, 4.