DOI: 10.2507/36th.daaam.proceedings.xxx

ENERGY-AWARE TASK SCHEDULING FOR SECURE AUTONOMOUS VEHICLES: AN ANALYSIS OF CURRENT SYSTEM DESIGNS

Suad Nesimi & Ervin Domazet

This Publication has to be referred as: Nesimi, S[uad]. & Domazet, E[rvin] (2025). Energy-Aware Task Scheduling for Secure Autonomous Vehicles: An Analysis of Current System Designs, Proceedings of the 36th DAAAM International Symposium, pp.xxxx-xxxx, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-xx-x, ISSN 1726-9679, Vienna, Austria
DOI: 10.2507/36th.daaam.proceedings.xxx

Abstract

The adoption of heterogeneous architectures is driven by these demands, integrating CPUs, GPUs, and FPGAs to perform diverse tasks such as perception, planning, and control on different hardware. Such platforms promise enhanced performance and greater flexibility. This paper presents an overview of how contemporary industry and research systems act in task and energy scheduling on heterogeneous AV platforms. We analyse the solutions at hand - including NVIDIA DRIVE, Tesla's Full Self-Driving Computer, Mobileye EyeQ and AUTOSAR Adaptive to examine how they coordinate processing units and enforce security isolation. At the same time, we investigate how they balance energy consumption with real-time performance. The paper also highlights the trade-offs that these systems make and identifies key limitations like scheduling of mixed criticalities being lacklustre, for lack of a better phrase, integrated into the current schemes, as well as the absence of standardised thermal-aware mechanisms. Additionally, we must consider fragmented security models. By synthesising these existing findings and observations, as well as our research conclusions, we point out future research directions that aim to bring us closer to the system-level designed applications of energy-saving and secure autonomous driving platforms.

Keywords: autonomous vehicle; heterogeneous components; secure systems; task allocation; thermal management.

1. Introduction

Many computation-intensive tasks need to be conducted in modern AVs, including but not limited to multi-sensor fusion, perception, real-time control and decision making. These are high-performance, low-latency responsibilities that require task determinism to operate reliably and safely in dynamic environments. To satisfy these demanding conditions, AV platforms are more equipped with heterogeneous computation engines including general-purpose CPUs: massively parallel GPUs and hard- or software reconfigurable FPGAs [1], [3], [6]. The structure offers flexibility and processing capability to perform simultaneous parallel computation with scalability and real-time performance. But there is a catch to this flexibility. In such environments, the project of task scheduling, energy efficiency and system security is very

difficult. Energy-aware task scheduling has recently emerged as an important research thrust to reconcile the trade-off between performance requirements of safety-critical applications plans with limited energy budgets and heat dissipation constraints [2], [4], [5]. Such scheduling is charged with the obligations of consuming minimal energy while meeting time deadlines for critical perception and control tasks under unknown workloads and hardware.

Recent advances in the industry, such as NVIDIA DRIVE AGX Orin [8], Tesla's Full Self-Driving (FSD) computer [9], and Mobileye EyeQ5 processors are also examples of the trend of heterogeneous architectures in AV, which integrates CPUs, GPUs and specialized accelerators for deep learning inference and sensor fusion [10]. These systems highlight the benefits of tightly integrated heterogeneous processing for enabling real-time decision-making, but they also open new challenges that come with sharing computing resources between workloads and ensuring isolation and security. Projects such as AUTOSAR Adaptive Platform further highlight the need for secure task coordination and communication in distributed automotive systems [11].

Although extensive academic research and industrial adoption have been achieved, to the best of our knowledge, there exists no comprehensive application which unifies the aspects (i.e., energy-efficiency; real-time scheduling; secure operation) during AV-heterogeneous SoC design. Previous work has investigated methods such as task-structure-aware scheduling [6] or GPU-based deep learning optimization [7], as well as dynamic system reconfiguration under energy constraints [2]. Nevertheless, the practical and theoretical challenges in incorporating such approaches into end-to-end autonomous driving stacks persist.

The goal of this paper is to present a systematic literature review and critical study on the recent energy-aware task scheduling approaches for autonomous vehicular systems running on heterogeneous computing architectures. Rather than introduce new algorithms, we are concerned with observing how existing solutions, both from commercial trends and academic concepts, tend to organize task allocation and processor coordination AND secure execution under constraints of efficiency and real-time guarantee. The observations reported here are intended to recognize current challenges and suggest directions for designing future energy-aware and secure heterogeneous computing platforms in autonomous vehicle systems.

2. Background

As autonomous vehicles (AVs) have become smarter, the computational complexity of their software systems has grown with each new advance. These systems must process an enormous quantity of data from cameras, Light Detection and Ranging (LiDAR) sensors, radars and other sensors. For these systems to do this in a timely and safe manner, as well as (more difficult yet) meeting strict real-time criteria, is crucial, especially for brake control, obstacle avoidance and lane-keeping tasks. However, to meet these performance demands, today's AV platforms use a variety of processing elements. These range widely from more typical CPUs like ARM-architecture microprocessors to GPUs and often FPGAs (or ASICs). Each processor has its advantages:

1. High-performance sequential control tasks, logic-heavy processing and system orchestration are all handled well by CPUs.
2. With their massive parallelism, GPUs provide a great fit for computationally intensive tasks such as deep learning inference and point cloud processing.
3. FPGAs are commonly used in preprocessing sensor data or safety-critical modules. They offer low latency along with highly customised data path control facilities.

This wide range of processing units not only brings flexibility and power but also raises fundamental questions of how to match tasks effectively across them in ways which strive for a balance between categories such as performance, real-time constraints, and security.

2.1 Task Scheduling in Autonomous Systems

Task scheduling is the process of determining which tasks should run on which processor at any given time. In the context of AVs, computational tasks differ not only in complexity but also in their urgency and timing. Scheduling decisions that are made poorly will mean that a task misses its deadline for whatever reason; this usually results in unsafe behaviour. Worse still, if tasks are not isolated or lack security, then faults and cyber threats can spread from module to module unimpeded.

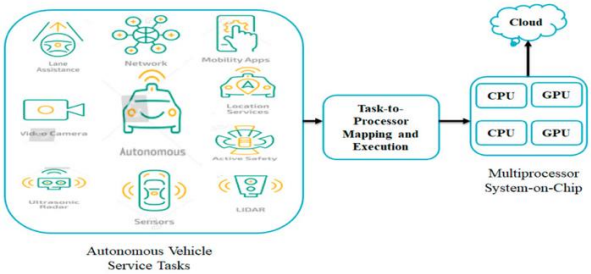

Fig. 1 . Task scheduling in autonomous vehicles

The task-to-processor assignment used in this work is shown in Fig. 1, adapted from [14]. On the left, perception and actuation workloads generate input flows that enter the central Task-to-Processor Mapping and Execution stage, where they are scheduled onto heterogeneous units. The right-hand side shows the on-board multiprocessor SoC (CPUs/GPUs) executing the tasks, while a cloud uplink is reserved for non-safety-critical analytics and storage. Arrows indicate data flow and motivate why the placement fits the scheduling discussion.

2.2 Energy-Aware Scheduling

For AVs, especially those on electric vehicle (EV) platforms, energy efficiency is a crucial consideration. Each kind of processing unit is a glutton when it comes to power, and poor scheduling gives rise to overheating, diminished battery range or the requirement of under-clocking to respond to thermal pressure. Energy-aware scheduling involves organising tasks onto processors to minimise their energy consumption without degrading performance or safety. Techniques include dynamic workload balancing, power gating and keeping track of hot sections that are temperature sensitive.

2.3 Task Criticality and Mixed-Criticality Systems

The tasks in AV systems may be classified as critical or non-critical according to their influence on safety. Critical tasks include real-time control, braking and sensor fusion modules. System breakdowns or accidents are the direct result of malfunctions in these areas.

2.4 Non-critical Tasks

Non-critical tasks include user-interface processing, infotainment, and logging. If the system allows both types of tasks to share a hardware platform, it is a mixed-critical system. Careful scheduling and isolation are necessary to avoid situations where non-critical faults or delays interfere with the critical operations of safety.

2.5 Secure Execution and Isolation

Security is a major concern. Autonomous vehicles are subject to both physical and network-based threats. This necessitates secure execution through task isolation. Hardware features such as memory protection, privilege separation or trusted execution environments (e.g., ARM TrustZone, Intel SGX). Software containers and microkernels that enforce task separation and restrict access to shared resources. Effective task scheduling should take these requirements into account, ensuring that tasks from different security domains do not interfere with one another.

2.6 Purpose of this Paper

This paper is not going to introduce a new task scheduling algorithm or framework. Rather, it focuses on: Looking at current industry and research practices for energy-aware task scheduling on AVs. How these systems succeed or fail in managing criticality, coordinating processors to satisfy energy constraints and providing protection. Highlighting limitations, gaps and new opportunities for more integrated, intelligent scheduling strategies in the future.

3. Review of Existing Systems

In this part, we will try to review a selection of prominent industry platforms and academic systems that support AV functionality. Each is titled according to the target task scheduling, power efficiency, and secure execution on substrate platforms, which contain security processing units within central processing units implicitly or explicitly as part of their design.

3.1 NVIDIA DRIVE

NVIDIA's DRIVE [8] platform is the most widely used autonomous driving solution for commercial and research AVs. It is equipped with multi-core ARM CPUs, high-throughput GPUs and dedicated Deep Learning Accelerators (DLAs). Task scheduling is usually performed by applications using NVIDIA's SDKs (e.g., DriveWorks, TensorRT), and then developers are responsible for the detailed deployment of the workload CPU, GPU and DLA. While NVIDIA's design does support high-performance inference and real-time sensor processing, its ability to schedule for energy leads to less tightly organised scheduling behaviour. Developers may use Dynamic Voltage and Frequency Scaling (DVFS) or thermal-aware power domains, but these are hardware features and cannot be considered software-level schedulers in any meaningful sense of the term. Security is handled at the system level, with support for secure boot, root of trust, memory

partitioning and so on. Little integration between the scheduling logic and isolation means little attention is paid to incorporating these principles into concrete mechanisms for practical enforcement of the security policy.

3.2 Tesla Full Self-Driving Computer

Tesla's FSD [9] computer has a custom-built SoC architecture that consists of two redundant chips. Inside are CPU clusters, GPU cores, and specialised Artificial Intelligence (AI) accelerators on each chip. The platform has been developed to be highly redundant for the parallel execution of perception and planning pipelines on different chips. Scheduling is an internal detail, but Tesla seems to do static partitioning and use the dedicated compute blocks for some workloads. Energy management is probably implemented at the level of silicon, with dedicated logic for reducing power consumption under idle or partial-load conditions. Security mechanisms such as fault detection and watchdogs are found, but due to the closed-end nature of the system, it is hard to evaluate how dynamic or adaptable the task scheduling model is, especially when different driving situations or processor loads are taken into consideration.

3.3 Mobileye EyeQ

Mobileye's EyeQ [10] series is dedicated to vision processing and deployed in many Levels 2–3 AVs. Its design is based not on GPGPUs but on highly optimised ASICs. This is very energy efficient, but less flexible. EyeQ systems are deterministically assigned hardware-pipelined (statically) such that performance can be guaranteed. Although it provides fine-grained control over when actions execute and how much energy they consume, it doesn't provide a way to either schedule actions dynamically during execution or to adapt to changes in the environment or the availability of resources. The platform is tailored for use in the context of ISO 26262 ASIL-D requirements with high safety integrity levels, which is less appropriate for full-stack autonomous systems making on-board decisions which require the reconfiguration of tasks

3.4 AUTOSAR Adaptive Platform

AUTOSAR Adaptive [11] is a software platform designed for new-generation ECUs, which are used in ADAS & AVs. It is capable of service-oriented communication, POSIX-compliant OS environments and deterministic application execution. AUTOSAR Adaptive does not support energy-aware scheduling. On the contrary, power management is commonly devoted to the ECU as OEM-dependent approaches. Also, task scheduling policies depend on the RTOS and the middleware selected by the designer. From the security point of view, AUTOSAR Adaptive provides some degree of support for role-based access control, secure communication and application separation. But these capabilities are not coordinated well with scheduling logic, and coordination between energy, timing, and security becomes suboptimal.

3.5 ROS2-Based Research Platforms

ROS2 [12] is commonly employed both in academic and open-source AV projects. It facilitates the distributed, modular design of AV functions, with nodes executing on various processing nodes. Real-time execution is also partially supported in ROS2 using the OpenSplice DDS middleware, and with Real-Time Operating Systems (RTOS) on systems such as RT Linux. There is no built-in support for energy-aware scheduling in ROS2. All energy optimisations must occur from the OS up or at a hardware level. Likewise, task criticality and fault isolation must be imposed manually, e.g. using containerization or external safety supervisors. Despite its flexibility, ROS2 is still constrained in system-level integration of scheduling, energy and security, deciding between the two rather than being a prototyping platform rather than a production-level AV solution.

4. Analysis and Observations

Based on these platforms, some trends and deficiencies can be identified, which indicate the status and immaturity of autonomous vehicle integrated energy-aware secure scheduling to a large extent.

4.1 Lack of Dynamic Scheduling Intelligence

Most of the platforms are based on a static or semi-static task assignment model. Real-time dynamic scheduling that considers the load on the network, available power, or the criticality of the task has not been well addressed. This may result in over-utilisation (and thermal strain) or under-utilisation (and wasted performance).

4.2 Disjoint Energy and Security Management

A lack of consideration is given to energy and security as a joint issue. Although energy principles can exist (DVFS, idle modes) and security features like secure boot can be supported, few systems integrate these during scheduling decisions. There is no known system that automatically schedules tasks to achieve both power efficiency and an isolation domain.

4.3 Minimal Support of Mixed-Criticality Systems

Even when these assumptions are not posed, most systems that are aware of mixed-criticality requirements (e.g., AUTOSAR) provide very limited actual runtime separation and priority management of such tasks. This is dangerous for system stability and functional safety in case of high calculating load, or partial failure.

4.4 Limited Interoperability and Openness

Closed-source providers such as Tesla and Mobileye provide minimal visibility into how they make scheduling decisions, so it is challenging to draw comparisons between solutions. At the same time, open platforms such as ROS2 are not yet mature and integrated enough for commercial deployment of AV (Table 1).

To facilitate a qualitative comparison of the five AV platforms in this paper, Table 1 summarises them. The table compares each system according to 5 main characteristics: task scheduling mechanism, energy-awareness, security and isolation functions, support for mixed-criticality workloads and openness to the research/development community. The following table summarises the portability and integration maturity of these platforms.

For example, to schedule in the power and task domain of developer-level control on NVIDIA DRIVE; however, it does not include something for criticality-based scheduling. Meanwhile, Mobileye's EyeQ supports extremely efficient computation, but is inflexible because of its fixed-function nature. This table aims to provide readers and reviewers with an overall view of the strengths, weaknesses and limitations for each system immediately, facilitated by detailed discussion presented in the previous section.

Platform	**Task Scheduling Approach**	**Energy-Aware Support**	**Security & Isolation**	**Mixed-Criticality Support**	**Openness/Availability**
NVIDIA DRIVE	Manual/ developer-driven	Hardware-level DVFS, power domains	Secure boot, memory partitioning	Partial (developer managed)	Partial (SDK access)
Tesla FSD	Likely static, internal only	Custom chip design, likely hardware-managed	Hardware watchdogs, dual redundancy	Via chip separation	Closed source
Mobileye EyeQ	Static/hardware pipeline	Extremely efficient ASIC-based design	Meets ASIL-D, strict isolation	Limited	Closed source
AUTOSAR Adaptive	OS/middleware dependent	Vendor-specific implementations	Role-based access, secure comms	Theoretical support	Open source
ROS2-based Systems	OS-level & node-based (flexible)	None natively	Optional via SROS2, containers	Developer-defined only	Fully open source

Table 1. Comparison of task scheduling, energy-awareness, security, and system openness across representative autonomous vehicle platforms.

5. Experimental Results

While the main topic of this work is a thorough overview of existing systems for energy-aware task scheduling in secure AVs, we present a simple simulated example below to demonstrate the challenges when allocating tasks to a target platform with a heterogeneous amount of processing capabilities. This is not intended as a performance benchmark. The purpose is to demonstrate the concept of how scheduling impacts energy and execution time in real AV systems.

To emulate a realistic workload in an autonomous driving context, we define three representative tasks:

- T1: Object Detection (compute-intensive, latency-tolerant)
- T2: Emergency Braking (time-critical, safety-sensitive)
- T3: Lane Keeping Control (low latency, real-time loop)

These tasks are mapped to three processor types commonly used in AV platforms:

- CPU (deterministic control, lower power)
- GPU (parallel high-throughput, high energy)
- FPGA (low latency, energy-efficient)

Simulation results show that Scenario B (Energy-aware mapping) saves 45% of the total execution time and 41% of the energy consumption compared to Scenario A, which uses an energy-unaware allocation. Placing the safety-critical Emergency Braking on the CPU improves both energy efficiency and the task determinism at runtime, while transferring the Lane Keeping Control to the FPGA provides the low-latency processing. While this is a step toward increasing the efficacy of both offline and online task scheduling policies based on complex real-world data, we posit this as an encouraging average case that should be used to vet other scheduling policies for intelligent frameworks in simulation, which, even in simulation, have the potential to provide large improvements in terms of both energy and response time in real time AV systems. Although actual results may vary according to hardware and task models, the direction demonstrated indicates the need for context-aware, dynamic scheduling schemes on heterogeneous AV platforms. This is a demonstrative simulation, not based on actual hardware experiments. However, it does help to confirm the usefulness of the design considerations presented in the earlier sections. We may also develop this simulation on a real-time middleware platform (e.g., ROS2 with DDS), and model it on machines (e.g., NVIDIA Jetson, Drive AGX, AUTOSAR Adaptive-based ECUs).

We provide a light runtime-supervision layer, which subscribes to raw and pre-processed sensor topics, and performs inference with small neural models on the CPU side. The monitor emits raised events (anomalous, drifting and missing frames) that the scheduler treats as soft signals for admission control and mode switching without rejecting components responsible for controlling any safety-critical loops. To minimize overheads, inference is rate-limited and memory-pinned and put in a separate domain. This architecture is motivated by previous results indicating that NN-based monitoring of sensor signals can be performed with an acceptable latency and resource footprint for online applications [13].

Task	Scenario A	Exec. Time (ms)	Energy (mJ)	Scenario B	Exec. Time (ms)	Energy (mJ)
Object Detection	CPU	40	35	GPU	18	60
Emergency Braking	GPU	12	50	CPU	10	12
Lane Keeping Control	GPU	10	48	FPGA	6	7
Total		62	133		34	79

Table 2. Comparison of task execution time and energy consumption under naive vs. energy-aware task scheduling strategies.

Table 2 illustrates the impact of assigning perception, safety-critical tasks, and control to the appropriate processing units, such as CPU, GPU, and FPGA. The energy-aware scenario achieves significantly lower overall energy usage and improved response times through criticality-aware task allocation and hardware.

6. Challenges and Limitations

The overview of today's systems highlights the multiple challenges that the research community and industry face in paving the way towards more dependable, secure, and energy-efficient autonomous vehicle platforms.

6.1 Unified Scheduling Strategies

There is an urgent demand for schedulers which take timing constraints, energy budget, processor utilisation and task criticality into consideration. A unified approach will require running on multiple hardware domains (CPU, GPU, FPGA) on the fly.

6.2 Cross-Layer Security and Scheduling Integration

In the future, security should be part of the task scheduler rather than a bolt-on. e.g., the verification of the operation's origin, the assignment of rights to be executed, or the separation during execution of potentially unsafe or faulty code parts.

6.3 Middleware-Level Support

Native middleware-level APIs for measurement of power, task profiling and dynamic load balancing should be supported by AV stacks. It would have allowed application developers to write scheduling-friendly code without having to understand much about the hardware.

6.4 Evaluation Frameworks and Benchmarks

In such systems, since users' job experience makes their experiments more reproducible and results more reliable, we must procure open benchmarking environments to perform comprehensive performance evaluation of scheduling and energy management strategies under realistic workloads and fault conditions. They would facilitate academia-industry collaboration on best practices benchmarking.

6.5 Implications / Consequences

Enforcing energy and security-aware scheduling can further enhance AV range and diminish thermal throttling by smarter DVFS and load placement, better respect of deadlines for safety-critical loops through criticality-aware mapping, reduce lifetime costs by lowering the impact of thermal stresses on components, as well as lower attack surface area by enforcing isolation domains at scheduling time. Possible downsides: Complexity and potential overhead for the scheduler of added data that needs to be limited and justified (ISO 26262). Power, thermal, and isolation telemetry may demand more observability. Portability or certification issues involving various vendor APIs, Vendor lock-in if there are proprietary power/thermal/security interfaces.

6.6 Limitations

We have built a qualitative review, relying on what is publicly available documentation for such platforms as Tesla or Mobileye; this limits the ability to verify our claims through means such as telemetry or isolation mechanisms. Our table shows a simulated system. Although the numbers are easily tested directly on real hardware, they omit memory bandwidth, data-movement overheads, interference into interconnect contention latencies and sensing/actuation lag. Changes in environmental temperature, such as heating up and cooling down processes that respond to changes in how hard you keep both power transformer windings connected to one amplifier or another--these are not considered. Of course, with individual thermal simulations, no actual temperatures or powers reached the physical object being modelled. We also do not provide a single unifying cross-platform benchmark or support, so our work may only generalise to heterogeneous SoCs with similar accelerator mixes from different vendors and power/thermal APIs. Future work in this area must include hardware-in-the-loop experiments, standardised workloads, and physically measured power/temperature fluctuation traces.

6.7 Interpretation of Results

The toy experiment in Table 2 suggests that mapping, which is conscious of criticality and energy-wise lowers both. If you assign Emergency Braking to the CPU and Lane Keeping to the FPGA, the total time falls to (62 → 34 ms), and energy also dips down along the curve (133 → 79 mJ). However, if you move heavy perception into the GPU, it keeps the throughput unchanged. These developments align nicely with earlier findings that (i) heterogeneous offloading enhances speed through acceleration paths lined up well with data structures [6], [7]; and (ii) energy saving comes when safety-critical, tight-loop control goes on low-latency, low-power sources [2], [4]. One key point is that latency is minimised for all critical tasks while levels off are reached in terms of energy/thermal. But because in these benchmarks there is no bus contention or thermal effects penetrating standard workloads, readers might form their own judgment as to the values, as a clue at least. And because isolation-aware placement of real-time control tasks on (CPU, FPGA), and

perception tasks on (GPU) is consistent with our thesis, this kind of tested hardware evaluations in Part II support our arguments.

7. Conclusion

The ongoing development of autonomous vehicles is conditioned largely on the progress made at the system level – beyond simply increasing numbers of sensors and increasing the complexity of the processing algorithms – and in the areas of task scheduling, energy efficiency, and secure execution. We had accordingly offered a systematic analysis of how state-of-the-art AVs deal with these factors in multi-processor settings. Although significant improvement of hardware performance and functionality has been achieved, our investigation discovers a huge space in integrated scheduling intelligence, energy-efficient task scheduling, and run-time security isolation. Existing systems generally address them in an isolated manner; thus, the adaptability and robustness under various practical constraints are restricted. Quantifying these gaps and surveying architectural trends, this paper establishes a foundation for future works that seek to narrow the gap between platform potential and software intelligence. Collaboration between industry and academia will be crucial in the pursuit of the subsequent generation of fast, smart autonomous systems that are also efficient, secure, and reliable. Current approaches often concentrate on improving one area while overlooking others. This can make vehicles inefficient or unsafe. The typical use of different architectures, such as CPUs, GPUs, and FPGAs, highlights the need for unified planning tools that can address power consumption, response time, and safety requirements. Dealing with these trade-offs through a combined, multi-layered design will be crucial for developing platforms that are both high-performing and reliable in real-world situations.

8. References

[1] Zou, A.; Liu, Q.; Xu, H. & Wang, Y. (2021). A survey of real-time scheduling on accelerator-based heterogeneous architectures for time-critical applications, *Journal of Embedded Systems Research*, **15**(4), 233–250.

[2] Yi, S.; Kim, T. W.; Kim, J. C. & Dutt, N. (2020). Energy-efficient adaptive system reconfiguration for dynamic deadlines in autonomous driving, *IEEE Transactions on Intelligent Vehicles*, **5**(2), 112–122.

[3] Miao, Z.; Chen, L. & Zhang, P. (2021). Review of task scheduling methods for heterogeneous chips, *Journal of Computer Science & Technology*, **36**(7), 1258–1274.

[4] Li, R.; Feng, Y. & Hu, L. (2021). Energy-aware task scheduling on heterogeneous computing systems with time constraint, *ACM Transactions on Embedded Computing Systems*, **20**(5), Article 45.

[5] Perin, G.; Rosa, L. & De Oliveira, A. (2023). EASE: Energy-aware job scheduling for vehicular edge networks with renewable energy resources, *Frontiers in Computer Science*, **7**, 122–130.

[6] Zhang, X.; Li, C. & Wen, J. (2022). TSSA: Task-structure-aware scheduling of energy-constrained parallel applications in distributed embedded systems, *Journal of Systems Architecture*, **133**, 102790.

[7] Kaur, R.; Singh, M. & Gupta, P. (2022). A survey of advancements in scheduling techniques for efficient deep learning computations on GPUs, *Journal of Parallel and Distributed Computing*, **165**, 13–27.

[8] NVIDIA Corporation (2022). *NVIDIA DRIVE AGX Orin Developer Kit: Product brief*. Santa Clara, CA, USA.

[9] Greenhill, S. (2021). Inside Tesla's Full Self-Driving computer, *IEEE Spectrum*, **58**(10), 34–41.

[10] Mobileye Ltd. (2021). *EyeQ5 system-on-chip for ADAS and AV applications: Technical overview*. Jerusalem, Israel.

[11] AUTOSAR Consortium (2021). *AUTOSAR Adaptive Platform overview*, Release R21-11. Munich, Germany.

[12] Maruyama, Y.; Kato, S. & Azumi, T. (2016). Exploring the performance of ROS 2, in *Proceedings of the 13th International Conference on Embedded Software (EMSOFT '16)*, ACM, New York, 1–10.

[13] Topalova, I. & Radoyska, P. (2023). A model of an intelligent automation system for monitoring of sensor signals with a neural network implementation, *Proceedings of the 34th DAAAM International Symposium*, 0050–0055, B. Katalinic (Ed.), DAAAM International, Vienna, Austria. DOI**:** 10.2507/34th.daaam.proceedings.007. ISBN**:** 978-3-902734-41-9. ISSN**:** 1726-9679.

[14] Balasekaran, G.; Jayakumar, S.; Pérez de Prado, R. **(2021).** An Intelligent Task Scheduling Mechanism for Autonomous Vehicles via Deep Learning. *Energies*, **14**(6), 1788. ISSN 1996-1073. DOI: 10.3390/en14061788.