https://doi.org/10.52326/ic-ecco.2022/EL.02

The evaluation of the on-board computer architecture for TUMnanoSAT series of nanosatellites for carrying out missions

Viorel Bostan [1], ORCID: 0000-0002-2422-3538
Alexei Martiniuc [1], ORCID: 0000-0003-3407-3016
Nicolae Secrieru [1], ORCID: 0000-0002-9168-6049
Vladimir Vărzaru [1], ORCID: 0000-0002-5241-6265
Vladimir Melnic [1], ORCID: 0000-0003-4180-3654
Valentin Ilco [1], ORCID: 0000-0001-7210-3840
[1] Center of Space Technologies of Technical University of Moldova, Chisinau, Republic of Moldova,
viorel.bostan@adm.utm.md, alexei.martiniuc@sde.utm.md, nicolae.secrieru@cnts.utm.md,
vladimir.varzaru@cnts.utm.md, vladimir.melnic@mate.utm.md, valentin.ilco@cnts.utm.md

***Abstract*—In this paper, a brief overview of the nanosatellite on-board computer (OBC) functions will be described. The main advantages and disadvantages of the most common OBC architecture will be explained. A set of three different architecture variants for TUMnanoSAT series of CubeSat nanosatellites is proposed and described. A feature comparison is performed in order to highlight the improvements and advantages of the proposed on-board computer architecture designs over traditional CubeSat OBC architectures.**

Keywords*— *nanosatellite, CubeSat, on-board computer, TUMnanoSAT, microcontroller, OBC, architecture, scalability, reliability

I. INTRODUCTION

The on-board computer (OBC) is one of the key components of every satellite, including nanosatellites such as CubeSats. It is responsible for control and monitoring of all on-board subsystems, mission on-board scheduling control, detection and (whenever is possible) recovery of system errors, payload and system data logging and storage, telemetry data preparation, and in some cases, the attitude determination and control. Due to the many tasks assigned to and performed by the on-board computer, the design, testing and validation of an on-board computer module design and its software is very complex in order to develop a reliable key system component which will accomplish all those tasks flawlessly and with required performance.

Most common processor architecture used in today's on-board computer designs of CubeSat nanosatellites is ARM. This 32-bit architecture provide excellent combination of high computing performance at lower power consumption. Although the computing performance of ARM processors (of especially the ARM Cortex-M profile), is still lower than of traditional SPARC and x86 architectures, the ARM architecture is more power efficient, so for the same operation it consumes less power and dissipates less heat, while the speed performance is slightly lower. This makes ARM the architecture of choice for power and space constrained battery powered systems such as CubeSat nanosatellites (especially 1- and 2-unit configurations), where is no enough space for large and complex thermal control systems (TCS) and/or large energy storage and generators. Nevertheless, some semiconductor manufacturers have migrated some ARM based processor designs to radiation hardened/tolerant fabrication processes to leverage their power efficiency, scalability and large ecosystem in space applications.

The software running on most CubeSat platforms is developed on top of a real time operating system (RTOS). The RTOS allows to divide the entire application into separate, independent, simultaneously running threads or processes, each being responsible for a specific subsystem or task. On more demanding applications with heavy computational load, the full featured operating system such as a custom Linux distribution is used, running on a high performance single or multi-core processor with a memory management unit (MMU) for hardware accelerated virtual memory management.

The use of an operating system (real time or conventional) allows to create a modular software where a

fault in a subsystem and/or its specific thread does not necessarily affect other running threads. The isolation of threads for each task and implementation of a deterministic inter-process communication with a unified interface makes the development of the entire software less prone to software errors introduced during software development stage, that may affect entire system. In addition, each process can be developed mostly independent of other processes and tasks.

In this paper the emphasis will be placed on the hardware architecture of an on-board computer for a CubeSat nanosatellite, namely the TUMnanoSAT series of nanosatellites. Several design variants will be proposed, which address different levels of complexity, reliability and performance. A comparative analysis of the feasibility of each variant also will be presented.

II. DEFINITION OF OBC ARCHITECTURE REQUIREMENTS

The CubeSat nanosatellites are in fact a high-density systems, packed in a very small form-factor. Due to increasing complexity of missions, the design and integration of all CubeSat subsystems required for successful mission deployment pose a big challenge for designers of the nanosatellite subsystems. In order to ensure the proper operation of whole nanosatellite system during entire mission duration, several design principles must be followed during overall design process:

1. Minimizing the interdependences between different modules as much as possible without compromising overall system functionality and performance;
2. Avoiding centralized control of resources by a single module;
3. Multiple redundancy of critical modules;
4. Multiple failure protection mechanisms for critical components and subsystems including fault detection, isolation and recovery (for example against unexpected power system failures, latch up);
5. Use of high reliability electronic components from verified suppliers and, if available, validated during lab testing and/or flight;
6. Implementation of design-for-test (DFT) approaches and techniques at early design stages and during entire design process.

The OBC hardware, as other nanosatellite subsystems, represents physically a module, which in turn consist of single main printed circuit board (PCB) in a derived PC104 form-factor, with or without one or several mezzanine printed circuit boards or cards mechanically and electrically attached to it. All the nanosatellite modules, including OBC are stacked vertically in order to save space. Another issue is the limited energy storage and power budget due to space limitations.

In order to satisfy both these limitations and mission requirements, the nanosatellite modules, including OBC, must be designed to satisfy the following requirements:

1. Use of a hardware computing architecture and platform that embeds integrity self-check mechanisms, error detection and correction during runtime in all data processing components (CPU/ MCU, internal and external memories, programable peripherals etc.)

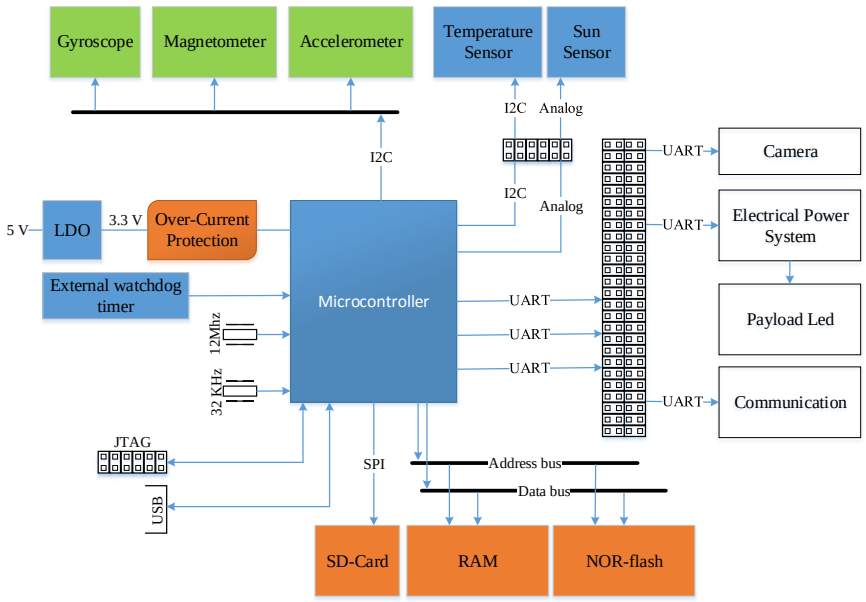

Figure 1. Common OBC structure used in CubeSat nanosatellites.

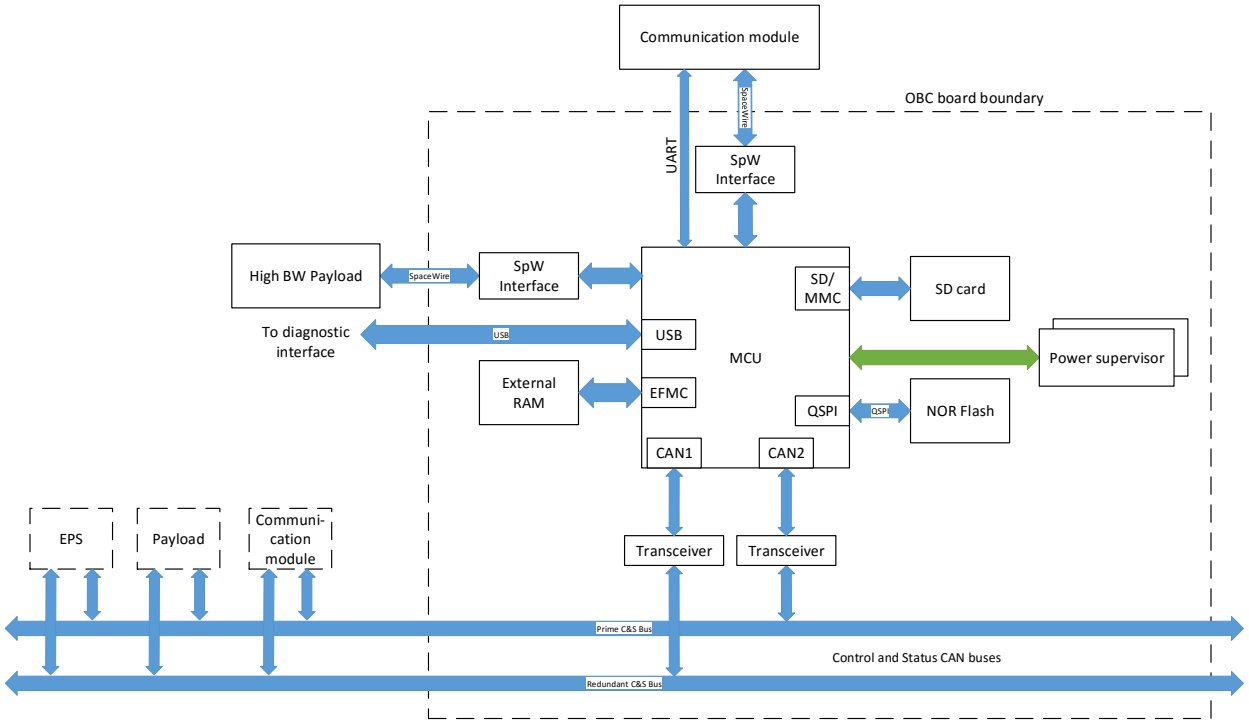

Figure 2. Basic variant of OBC architecture for TUMnanoSAT series of nanosatellites with one MCU.

2. Redundancy on critical components such power protection, control and monitoring communication interfaces, data storage units;
3. Lowest possible power consumption while maintaining required performance;
4. Use of high-density integrated circuits in high density, small packages;
5. Use of communication interfaces and protocols

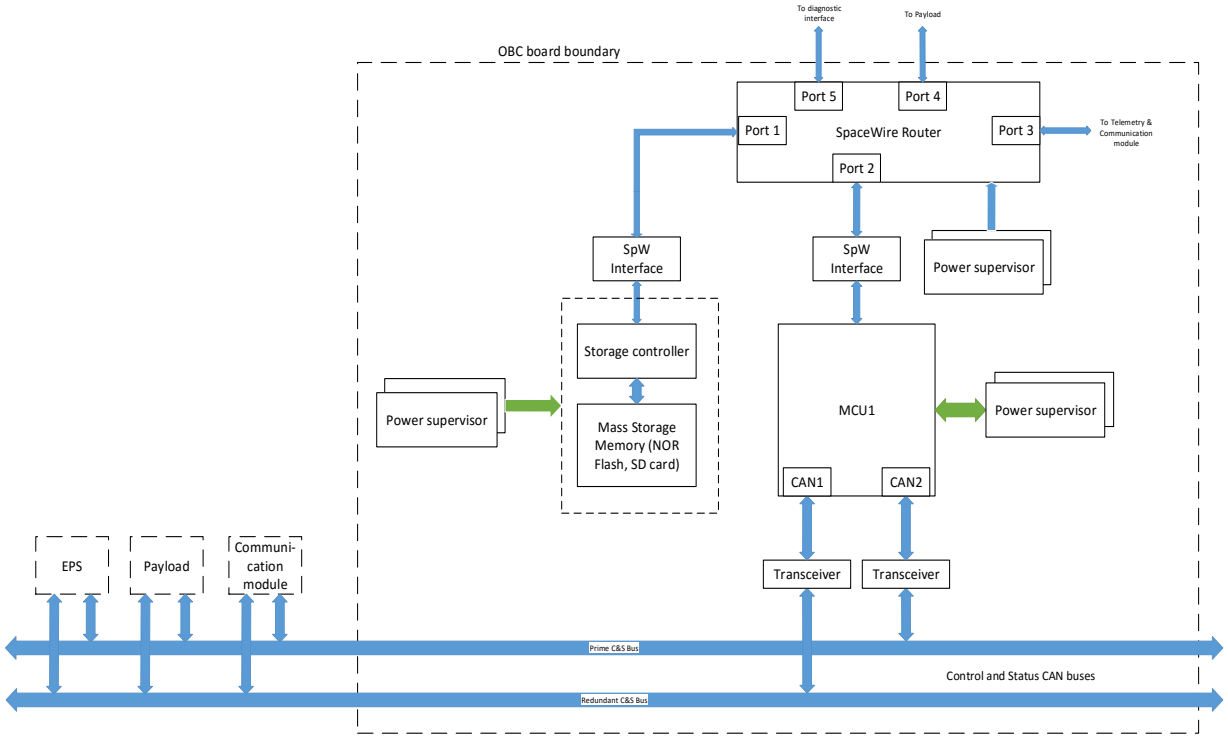

Figure 3. OBC architecture with a single MCU, a SpaceWire router and independent data storage unit.

with built-in error and fault detection and correction;

6. Performance scalability and modularity for a wide range of mission complexity.

III. PROPOSED OBC ARCHITECTURES

The basic OBC structure commonly used in educational CubeSats is shown in Figure 1. It incorporates both general OBC functionality and Attitude Determination and Control System (ADCS) on same microcontroller (MCU). For communication with other system modules, such as payloads, electrical power system (EPS) and RF communication module, it uses simple embedded interfaces and protocols such as UART, I2C and SPI, commonly used in standard embedded applications. All the electrical connections with other system modules of the satellite are routed to a PC104 header, which also serves as a backbone system bus that carries both power and shared signals from all nanosatellite modules. For debug purposes a JTAG or similar debug interface connector is also present.

The advantage of this architecture is a low cost and simple implementation. The drawbacks of this design are the centralized approach, in which the OBC microcontroller controls exclusively the access to data storage and other critical modules such as EPS and RF communication, the use of non-fault-fault-tolerant communication interfaces (SPI, I2C, UART) and the ADCS system, including the inertial data acquisition shares same CPU/MCU that has exclusive access inertial sensors. In case of failure of one communication channel or entire MCU, the system loses both OBC, data storage and ADCS functionalities. This example of a OBC design also has limited or no possibility to scale to different mission requirements because of a lack of unified standardized fault tolerant high interface which can connect multiple similar modules in a decentralized system network. Thus, this design cannot be used for future missions without partially or even fully redesigning it. Finally, it does not provide a high-speed reliable interface for high bandwidth data transfers between OBC and potentially high bandwidth data producers (for example high performance payloads such high resolution camera) or data consumers (for example high bandwidth downlink RF transmitter).

The three proposed OBC architectures for future TUMnanoSAT missions mitigate the disadvantages of the simpler common OBC design, although they are more complex. The primary goals for these designs are the improved reliability, performance, scalability and reusability. The first variant is the least complex among all three and is shown in Figure 2. It also has the lowest cost of implementation. The main distinctive features of this architecture are the use of double-redundant shared multidrop fault tolerant CAN based buses for system control and monitoring, multiple (two) high speed SpaceWire interfaces for high bandwidth data transfers up to 100 Mbps between OBC and payloads or communication subsystem, separate interfaces for commands, status and data between OBC and communication subsystem, and double-redundant power supervisors for protection against latch up and safe state backup in case of short duration power interruption. The

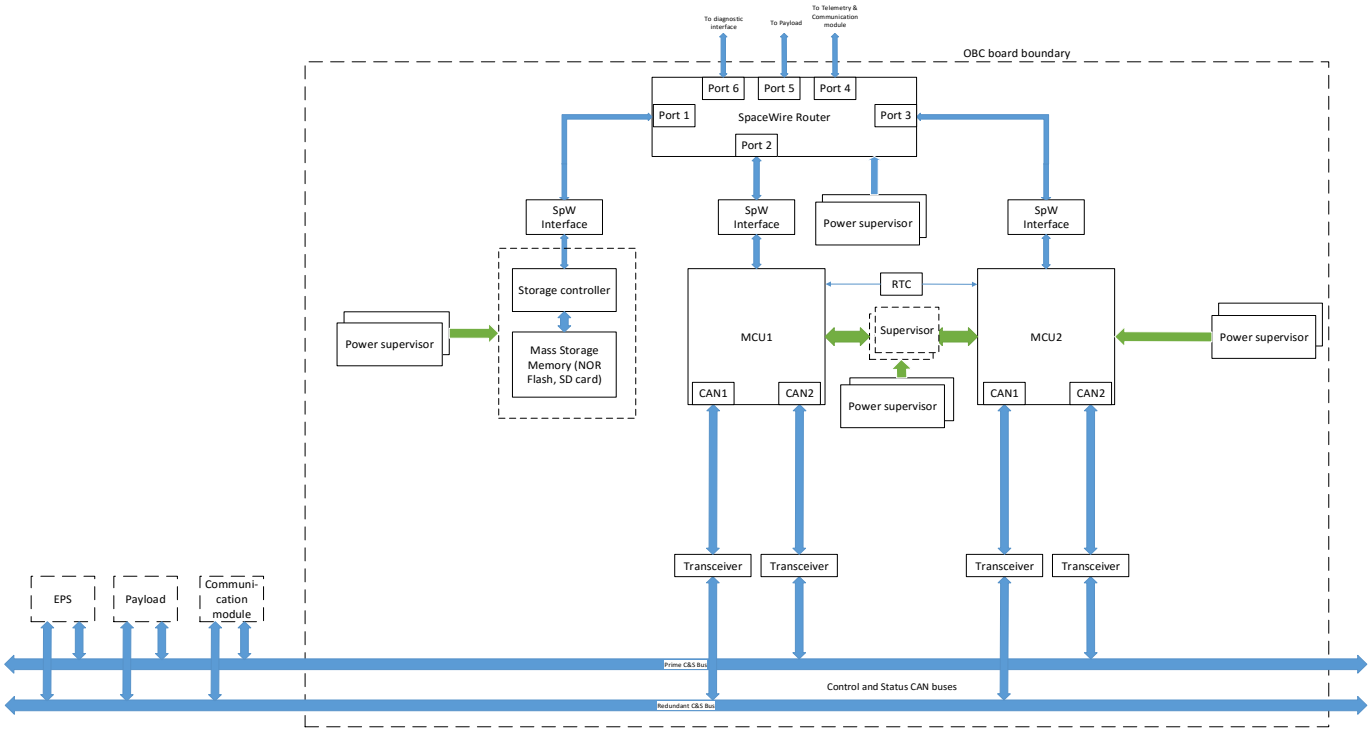

Figure 4 OBC architecture with two MCUs, a SpaceWire router and an independent data storage unit.

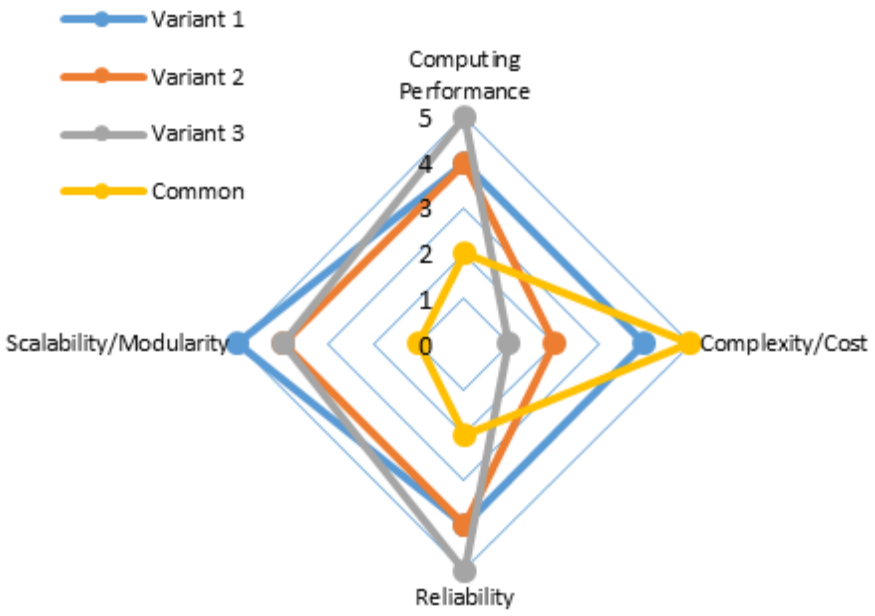

Figure 5. Comparison of features for all described OBC architectures on a scale from 1 to 5.

presence of multiple CAN control and status data exchange interfaces (which are multidrop) allows a unified standardized control interface for system monitoring, regardless of the system configuration or hardware module design. The number of modules connected to the unified control and status CAN buses is limited only by bus physical specifications. All control and status information are broadcasted on both buses and each subsystem module filters the messages and collects only the information it needs. Even if a bus is blocked by a faulty module connected to it, the secondary redundant bus can be used for data exchange. All the control commands and status information are exchanged on the bus using CAN messages, eliminating the need for separate control and status signal lines routed from one module to other via backbone connector. This allows to make the system decentralized and hence to minimize the impact of one module failure over another subsystems (except main power system). The SaceWire ports of the OBC may be connected directly to another subsystems such as payload or communication module, or via an external SpaceWire router as an auxiliary module.

The second variant, shown in Figure 3, is more complex. It inherits the features of the first variant, including the high speed SpaceWire interfaces and decentralized control and status CAN buses, and contains an integrated SpaceWire with two ports connected locally router and a data storage unit, both with their own double-redundant power supervisors. The advantages over previous variant is the possibility to access data storage independently of the main MCU/CPU of the OBC, even if the second fails to operate properly. SpaceWire router allows subsystems to communicate in a concurrent way with each other when the source and destination are different.

The third variant, shown in Figure 4 is the most complex. It inherits all the features from the first two variants and adds a redundant MCU/CPU to the OBC with a double-redundant supervisor for synchronization and monitoring the operation of both processors. The processor supervisor detects abnormal operation of one processor via dedicated signal lines and protocol, and switches the execution to secondary processor. To accomplish this, the supervisor monitors the execution address and state of the current running processor, and contains a data cache for runtime context. It is also possible to configure the MCU supervisor to enable the operation of each MCU in tandem, thus dividing the computational workload between them. In this case the supervisor's data cache memory are used also for shared data, and the supervisor itself maintains shared data coherence between the MCUs. This mode of operation is called hot redundancy of MCUs while the mode with only one active MCU at a time is called cold redundancy of MCUs.

The comparison chart of all 4 OBC architectures described in this paper, shown in Figure 5, summarizes the advantages and disadvantages of each architecture.

From cost and complexity perspective, the common OBC architecture has the best value. However, it lacks in scalability and modularity due to absence of unified standardized data and control interfaces. Because of its centralized approach, the reliability also is relatively low compared to the architectures proposed. The performance of the common OBC architecture is moderate because there it depends greatly upon the performance of the MCU chosen.

On the other hand, all three proposed OBC architectures feature a high level of scalability, modularity and reliability, thanks to decentralized system design, multiple standardized redundant system interfaces and built-in component redundancy. However, the complexity, and hence the cost, are higher than for common OBC architecture variant. The first proposed variant is the best tradeoff between all the features used for comparative analysis. On the other hand, the best performance and reliability is possible to achieve with third variant due to hardware parallelism and computing unit redundance.

IV. CONCLUSIONS

The reliability of the nanosatellite, hence the mission success strongly depends by the intrinsic reliability of its component subsystems including the main computing unit, the on-board computer, which has a key role in maintaining the successful operation of the entire nanosatellite. Furthermore, as the nanosatellite subsystem reusability for multiple missions allows lower design, manufacturing integration costs per mission and faster mission deployment, the modularity and scalability of the OBC become a major requirement rather than an option. This can be achieved only if the key nanosatellite subsystems, and particularly the OBC, are designed at the architecture level with these features in mind. In this paper was shown that the proposed OBC architectures embed all the required features in terms of reliability, scalability and performance. Although the design and manufacturing costs per unit of these architecture variants are higher than for the common OBC architecture, the overall mission integration cost is lower and integration time is shorter due to reusability and scalability of the proposed architectures. As the TUMnanoSAT is a series of nanosatellites with different missions, one OBC variant can be used multiple times without redesigning the key module, focusing mainly on integration of missions, not the main nanosatellite hardware platform. This leads to faster mission deployment without compromising reliability.

ACKNOWLEDGMENTS

This work was supported by the project 20.80009.5007.09 “Elaboration and launches of the nanosatellites series with research missions from the International Space Station, the monitoring and its post operation, promotion of space technologies”.

REFERENCES

[1] J. Farkas, CPX: Design of a Standard Cubesat Software Bus, California State University, California, USA, 2005.
[2] CubeSat Design Specification (CDS) Rev. 13. The CubeSat Program, Cal Poly SLO, 2013. – In: http://cubesat.org
[3] J. Wertz, W. Larson Space Mission Analysis and Design, Third Edition, Physics, 1992, *pp. 645-685*
[4] Lwabanji Tony Lumbwe. Development of an onboard computer (OBC) for a CubeSat. – Cape Peninsula University, 2013, 178 p.
[5] Gregor Dreijer The evaluation of an ARM-based on-board computer for a low earth orbit satellite, Physics, December 2002
[6] V. Bostan, N. Secrieru, V. Ilco, V. Melnic, A. Martiniuc, V. Varzaru, *TUMnanoSAT Nanosatellite and Kibocube Program*, Conference on Communications COMM 2020. Ediția a 13-a, 18-20 iunie 2020, București. România: IEEE., 2020, pp. 503-508. ISBN 978-172815611-8.