Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 58-16 (2024) 181–186

A Modular Avionics Architecture for a Planetary Rover Demonstrator for Human Assistance

Nicola di Gruttola Giardino * **Federico Fantastico** *
Leonardo Maria Festa * **Giacomo Gorgerino** *
Federico Mustich * **Fabrizio Stesina** ** **Edoardo Vacchetto** *

* *Politecnico di Torino (e-mail: name.surname@studenti.polito.it).*
** *Politecnico di Torino (e-mail: name.surname@polito.it).*

Abstract:
Current planetary rovers are designed with mission-specific monolithic architectures, limiting flexibility and increasing costs. As the space economy grows, more infrastructure will be needed on the Moon and Mars, including numerous rovers for human assistance. To reduce development costs, this work proposes a modular architecture with a common hardware and firmware infrastructure.
The work was carried out by the Electronic Division of the DIANA Student Team, whose focus is on the development of Rovers for planetary exploration.

Keywords: Space Rover, Space Robotics, Space Avionics, Student Team

1. INTRODUCTION

Since the dawn of space exploration, robots and autonomous systems have been indispensable, from the early successes of Lunokhod on the Moon (Kassel (1971)) to the recent achievements of Perseverance on Mars (NASA (2020)). As humanity intensifies efforts to establish a permanent lunar and Martian presence, the demand for advanced robotic systems to support and augment human explorers becomes paramount. This demand is further amplified by the rapid growth of the space economy, with an expanding commercial sector driving innovation and creating new opportunities for robotic applications in space (Brukardt (2022)).

Space robotics is a rapidly evolving field that presents unique engineering challenges, from developing robust locomotion systems capable of traversing diverse terrains to creating intelligent control algorithms for autonomous operation in extreme environments. However, current avionics architectures for planetary rovers often lack modularity and flexibility, making them difficult to adapt to evolving mission requirements, diverse payload configurations, and the increasing demands of a rapidly expanding space economy.

This research aims to design and implement a modular avionics architecture for a planetary rover demonstrator, focusing on improved scalability, adaptability, and ease of integration for a wide range of human assistance and exploration payloads. This modular approach aligns with the future of space exploration, which demands a paradigm shift towards cooperative rover systems (Maxime Ransan (2006)) that can adapt to the expanding range of mission requirements in a dynamic and growing space ecosystem (Post et al. (2021)).

The proposed modular avionics architecture is expected to significantly enhance the capabilities of planetary rovers, enabling them to efficiently support human explorers in a wide range of tasks and mission objectives while contributing to the continued growth and diversification of the space economy.

2. BACKGROUND

2.1 DIANA

This work is carried out by DIANA, a University Student Team from Politecnico di Torino. Since 2008, DIANA has inspired hundreds of students from diverse academic backgrounds, fostering a passion for space exploration and robotics. The Team was born under the guidance of Professor Emeritus Giancarlo Genta, with the ambitious goal of sending a rover to the moon as part of the Google Lunar X-prize challenge. During its more than fifteen years of history, DIANA evolved through multiple phases. From the initial years, in which it had to deal with an actual space mission and space-grade hardware, and higher budgets, to the difficult transition to the scope of student competitions such as the European Rover Challenge (European Space Foundation (2024)), pivoting toward a more educational-oriented goal. This change of focus did not come without difficulties, as the team gradually changed from being largely composed of PhD students and young researchers to being composed mainly by undergrad and master students, having to adapt to different budgets and technical constraints.

Throughout its history, DIANA has focused on developing technological demonstrators of space robotic systems, striving to adhere to rigorous rules and methodologies for space development. To achieve this, DIANA now adopts an

10.1016/j.ifacol.2024.08.483

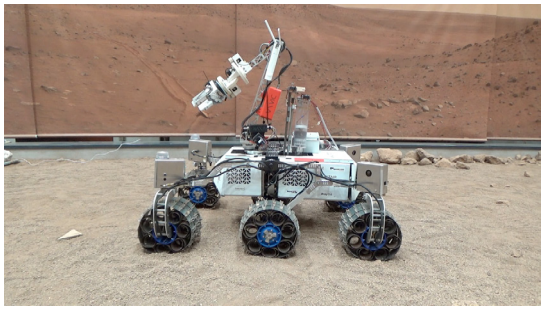

Fig. 1. ARDITO Rover (This photo was taken at the ALTEC Mars Terrain Simulator - MTS Facility - which was developed for ESA ExoMars program under Thales Alenia Space Italy contract)

highly structured organization that allows for both vertical development, based on each member's area of study, and horizontal integration, ensuring seamless collaboration across all rover subsystems. The rover development process is guided by a multi-V model, complemented by the utilization of various software tools for requirement identification and work package management.

The team is now supervised by Professor Fabrizio Stesina, in the STAR Laboatory of Politecnico di Torino (Stesina and Corpino (2021)).

2.2 ARDITO Rover

The ARDITO project, launched in 2019, was built upon the team's experience prototyping three rovers. ARDITO (Fig. 1) is a technology demonstrator of a Rover for supporting astronauts in planetary missions. The Rover is composed of a Platform, which comprises all the components needed for Command and Data handling, Electric power supply and the mobility system; Due to its modular nature ARDITO can be equipped with many kinds of payloads. The locomotion system is based on the rocker-bogie suspension model (Bickler (1989)). It features six driven wheels, four of which are capable of steering. The rover has manipulation abilities thanks to the six-degree-of-freedom payload arm, equipped with a sensorised claw. It is also capable of mounting additional scientific payloads, such as a soil sampler, to analyse the soil, and a box to collect, weight and analyse rocks. The system aims at autonomy by means of front mounted stereocameras and custom navigation algorithms (Mustich et al. (2023)), as well as complex algorithms to manipulate objects without human input.

2.3 Avionics architecture in current planetary missions

All current planetary missions being held on foreign bodies, such as the Moon and Mars, are built specifically purposed for the missions to be carried out. Because of this, their avionic architecture is mission-specific, and has no requirement concerning its re-use for future rovers.

An avionic comparison can be made on the publicly available information about three Mars missions: Curiosity (NASA (2011)), Perserverance (NASA (2020)) and the future ExoMars Mission (ESA (2028)), with the Rosalind Franklin rover. Zhurong (CNSA) has few available information, and will not be taken into consideration.

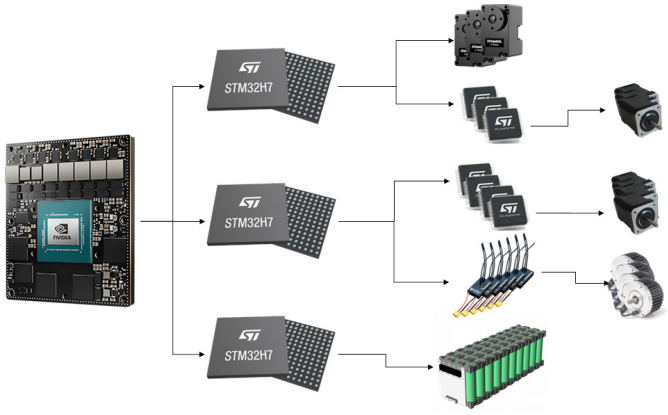

Fig. 2. ARDITO's Electronic Architecture, featuring Platform and DIANArm

The three missions share similar core functionalities. They have three main subsystems:

- The central avionics, whose CPUs are radiation-hardened, managing overall operations, using redundant unities for fail-safe.
- A Power Management Unit to regulate and distribute electrical power throughout the rover
- The Communication System, with high-gain antennas for Earth communication and lower-gain ones for relaying data through orbiters.

Curiosity and Perseverance share many architectural elements. The same dual-computer system, using two Radiation-Tolerant Avionics Computer (RTAC) units, is on the rovers. This avionic handles the main operation of the rovers. On Perseverance, dedicated computers control the catching and retrieval of rock samples.

ExoMars comprises a European-built computer, called Martian Surface Unit (MSU), managing the rover. The communication system, in its last iteration, prior to the last contract by ESA, was handled by the Russian Communications Module. Additional avionics modules are used for payloads, such as instruments analysing atmospheric gasses.

From the trend presented, all the avionics of the rovers are built by different contractors and require lots of R&D to be designed, bringing up the development costs of the system.
The architecture presented in this work, instead, aims at reducing the development costs by having a standard core architecture which can be reused for both the central platform and its payloads. This allows future mission to be more cost-effective, while also reducing maintenance costs, in case of human presence on the foreign body.

3. PROPOSED ARCHITECTURE

ARDITO's electronic architecture (Fig. 2) follows the same concept of modularity outlined in Section 2.2 for both the hardware and software sides.

3.1 Hardware Architecture

The hardware architecture of ARDITO is based on a decentralized approach. It features a central *Main On-*

Board Computer (MOBC), to which all the main Control Units (CUs) are connected via CAN Bus (Bosch (1991)). These control units are all built upon the same microcontroller: an STM32H743 (STMicroelectronics (2023b)) from STMicroelectronics. A suitable PCB has been designed and produced, which gives the I/O connections for the required peripherals of all the subsystems. It is able to use two different CAN Buses, one for the communication with the MOBC, and the second for forwarding commands to the lower level controllers, if needed. The latter can either be COTS components, such as a BMS or a BLDC controller, or a custom-made PCB.
The last layer of the architecture is made up of controllers used to interface with actuators and sensors of the subsystem they're attached to. This last layer is only added if multiple actuators are present, as for the Mobility System, on which six BLDCs and four steppers are to be controlled. This controller is based upon the STM32F446 (STMicroelectronics (2023a)) microcontroller, developed by STMicroelectronics. The PCB, seen in Figure 3, follows the same design modularity for fast replaceability. It is built on a main module which hosts only the analog devices required for the power to be routed across the board, and the signals coming out of the microcontroller. On top of it, two modules are attached, one hosting only the micro and the other for the CAN Bus transceiver. This choice was made because of how easily transceivers tend to break, with complex systems being constantly disassembled without all the necessary precautions. The final design allows the team to substitute only the damaged part when critical operations have to be performed soon after.

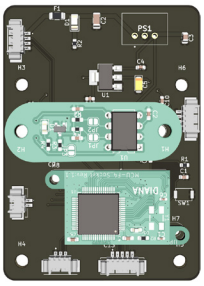

Fig. 3. Lower Level Controller PCB

The power consumption of the rover is delocalised in each control unit. The described PCBs have a measurement current sensor and then the data read is sent to the main on board computer by CAN Bus. In this way it is possible to reuse an infrastructure already existing, avoiding to wiring each actuator to a *Power Distribution Unit* (PDU) in an independent way by increasing the weight of the rover and the risk of failure.

3.2 Firmware Architecture

On both controllers described above, the firmware architecture follows the same modularity principle, by providing a standardized template from which to start, which has already implemented all the necessary features, such as Watchdogs and the CAN Bus communication interface with the *MOBC*.

The Main Control Units' firmware is developed using the Real-Time Operating System RTEMS (The RTEMS Project (2024)). The generic Board Support Packages (BSPs) were already available for the board in use. Additional ones had to be coded, such as the ones for the I2C communication. To use the CAN Bus, instead, the whole device driver had to be designed, as well as their BSP implementation for the STM32H7 microcontroller. The usage of devices instead of the available Hardware Abstraction Layer (HAL) functions gives the code an additional level of reliability and predictability, given their ability to lock the peripheral while it's being used by a single thread.
As aforementioned, a template is provided for all project to be started from. This template initializes the basic needed peripherals, such as both the CAN Interfaces, some timers, an SPI and an I2C interface, as well as a watchdog. All the interfaces are registered into the kernel of the OS, and a shell is started for debugging purposes. Finally, 7 basic threads are present: two for transmitting and receiving from the MOBC via the first CAN interface, while the other two are used to do the same, but with the lower level controllers. Libraries are provided, where all the addresses and commands are provided, so to have a standardized way of creating the frame. Another thread present in the template is the one used to run the control algorithm, if needed, such as the one for the movement of the six wheels or the robotic arm.
The sixth thread is used to execute a ping of all the underlying boards. This is needed mostly at startup, when the MOBC has to check if all the controllers are working. This thread is going to wait for an event triggered by a CAN frame with a unique ID. When the event is received, the thread checks if a message has been received by the lower level controllers. For each of them, a bit is set in the payload of the CAN Data Field, symbolizing that the CU is working. The message is then forwarded to the MOBC. Finally, the seventh thread is used to execute the checks required to allow the Watchdog to be reloaded.
Moreover, two additional safety mechanisms have been added, the first is to reboot the CU and all the lower level CUs through a message via CAN from the MOBC. The second is used to keep track of all the deadline misses of the threads, if any.
The template is then modified and adapted for the subsystem requirements, even removing threads, as can be the case if the CU works as a standalone. Threads can be added if the CU needs to interface with external sensors, following the same structure used for the already available ones.

The lower level Control Units, similarly as before, have their own template. The firmware is based upon the FreeRTOS Operating System (Amazon Web Services (2024)). STMicroelectronics proprietary software STM32CubeMX is used to initialize the peripherals and generate the basic FreeRTOS code, to speed up the development and avoid designing what's already available for free. These controllers are mostly used to control stepper motors in a closed loop with rotary encoders. For this reason, the threads used to execute this control are built into the template. This control uses on-line computed ramps to execute a smoother movement of the steppers, reducing the mechanical stress on the rover's joint to be controlled. As is for the main CUs, two threads are used to send and

receive data through the CAN Bus. Lastly, a thread for the watchdog is used as well.

4. ARCHITECTURE IMPLEMENTATION

This section describes the architecture implementation for the Platform of ARDITO, comprising the avionics and power delivery of the Rover, and of the two main subsystems of ARDITO: the mobility system and the robotic arm, called DIANArm.

4.1 Platform

ARDITO's platform is the main system of the rover, it comprises the two central modules:, the avionics and the power module, connected by means of a junction box comprehending various connectors to bring power from a module to the other.

The supply power, whatever the source (wired or battery), is plugged into an industrial connector that is welded on a board designed by our team, called PDU. This board is provided by supply protection controller to protect electronics systems from improperly connected power supplies, to avoid undervoltage, overvoltage and negative input voltages. To avoid to wiring each actuator to the PDU, the mechanism of a distributed power measurement explained in Section 3.1 is used. All the controllers spread throughout the system transmit their consumption via the main CAN Bus showed in Figure 4, to a central controller, called *Power Control Unit* (*PCU*), which collects and computes the power consumption of the Rover. The *PCU* uses the controller based on the STM32H7 described in the previous section.

Another fundamental part of the Platform is the *Mobility System*. It features a rocker-bogie configuration, with six wheels, four of which are capable of steering. In Figure 4, in the central pathway, the Mobility System's electronic architecture can be seen. It is composed of a main CU, called the *Steering and Propulsion Control Unit* (*PSCU*), to which four *Steering Control Unit*s (*SCU*)) are connected, to control the movement of the four steering-capable wheels. These boards, as well as the stepper motor and the encoder, are put into a steering box mounted onto the bracket. Finally, a Time-of-Flight sensor is attached to the *SCU*s, to measure the distance from the nearest obstacle in the rover's path. Moreover, six COTS board, called *VESC*s are used, to control the speed of each BLDC motor present in the wheels. To communicate with the latter, the same CAN Bus as for the SCUs is used, and the messages are encoded as described by the device's manual. The harnessing is designed by us, so as to fit with the mechanical constraints and to avoid all the noise that can be generated from the motor when the systems is turned off or while the rover is moving at high speeds, then the power and logic cables are divided and in addition all the communications harnessing are shielded in order to follow the Electromagnetic Compatibility (EMC), to prevent current spikes that can be dangerous for the MOBC and the PCBs.

To ensure the proper predictability of the firmware, a feasibility analysis has been performed using the Cheddar Framework (Singhoff et al. (2004)). The hardware and firmware descriptions have been put into the framework, which outputs the results seen in Figure 5.

Fig. 5. PSCU Feasibility Analysis

Precise and reliable locomotion control over harsh terrains is achieved by dedicated firmware running on a specific task added to the *PSCU* and interfacing directly with the underlying *SCU*s and *VESC*s. This system is responsible for translating the desired traverse speed and angles received from the user (or from the upper software layer) into commands to be executed by the wheels and steers motors. Our control design choice implements an Ackermann geometry model applied to a six-wheels configuration and differentiates between multiple traverse modes: *Straight* mode locks the steering angles to 0 degrees to avoid possible singularities, *Steering* mode is engaged whenever both the input steering angle and speed are different from zero, *Spinning* mode works similarly to *Straight* mode but by locking the steers on symmetric 45 degrees angles and allowing the rover to turn around its geometric centre. Finally, *Stop* mode allows the rover to maintain a zero velocity on each of its wheel, even in those scenarios in which a non-zero torque is necessary to keep the rover in place. This piece of firmware is also dedicated to ensuring, at each phase of the rover's movement routines, that it never exceeds its intended safety thresholds which might result in damaging the mechanical structure. For this reason, at this level we perform sanity checks to verify that the steering angles are kept within the geometrical requirements, and adopt solutions to ensure that the rover wheels only accelerate or decelerate uniformly and coherently with each other.

The described system constitutes the lower abstraction layer of ARDITO's navigation software stack. Built on top of it, the MOBC ultimately runs the software dedicated to controlling the rover's movement, either in semi-autonomous or in fully autonomous mode (Mustich et al. (2023)).

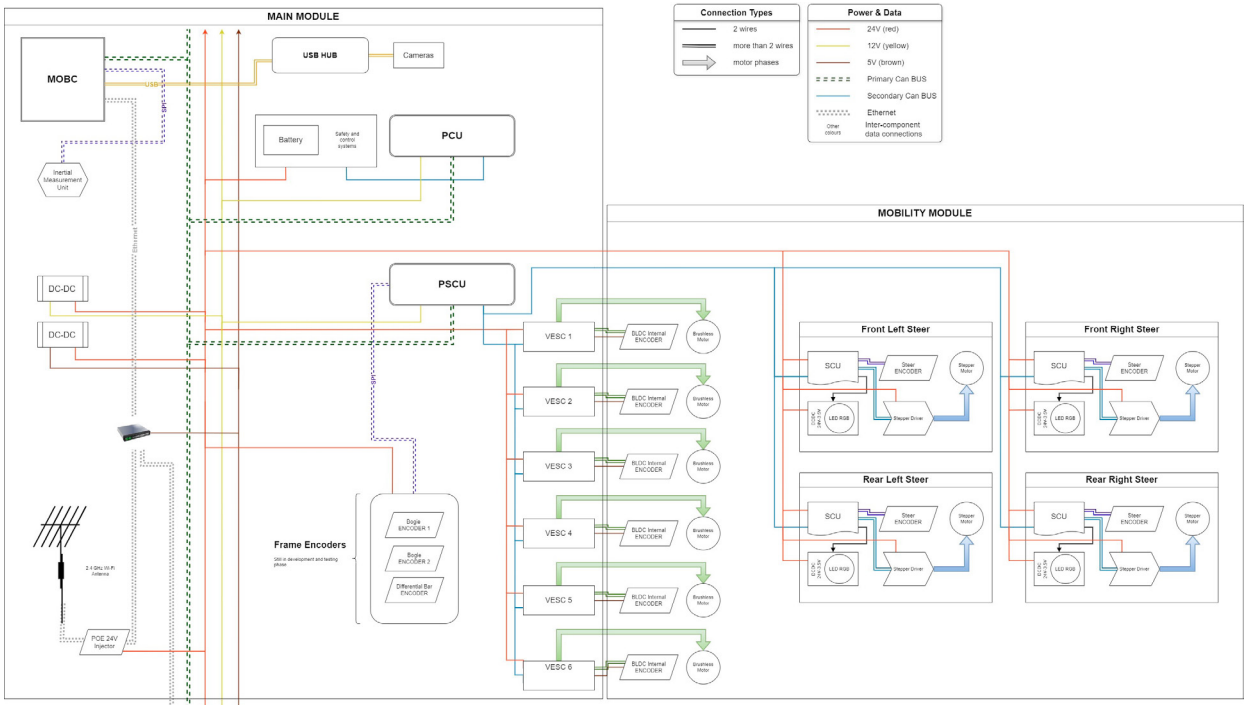

Fig. 4. Harnessing of platform and mobility systems.

4.2 DIANARM

The DIANARM is a six degrees-of-freedom anthropomorphic robotic arm and can be divided in two main parts, the structural Arm and the Wrist. Its main controller features the same hardware as the PSCU, which will be referenced as *Arm Control Unit* (*ACU*). It is directly connected to the three lower level controllers called *Joint Control Units* (*JCUs*), one for each joint they are to control.
The ACU is also connected to the wrist. The three DoFs of the wrist are moved through COTS motors, called Dynamixels (Robotis Inc. (2024)), which are controlled by the *ACU* via half-duplex serial communication. It also communicates with a board connected to the end effector, which is similar to the JCU. For space reasons, the carrier board had to be modified, while keeping the same sockets for the microcontroller, but removing the CAN transceiver. This board is controlled by the *ACU* via a custom protocol built on top of the serial communication. The end effector is capable of opening and closing a custom designed gripper, while reading the force applied. It can also measure and return to the operator the distance from a target object by making use of a Time-of-Flight sensor.

On the ACU, an *RTEMS*-based application, obtained from the aforementioned template, runs three additional tasks: one to control the the JCUs, one for the wrist, while a third one is added for activating the end-effector.

The DIANARM controller firmware is developed to have four different modes of operation: it can be controlled in position or incremental velocity, either in joint space or Cartesian space. In position mode the operator is able to control DIANARM by feeding it a target joint position or alternatively an absolute pose in the 3D Cartesian space (with the origin posed on the robot's root joint) to be reached by the end effector, if needed by computing the necessary inverse kinematics transformations. In incremental velocity mode instead the user can apply a more fine grained control by acting directly on the joint velocities or the Cartesian velocities of the end effector. This mode allows for precise adjustments in real-time, making it particularly useful for tasks requiring delicate manipulation capabilities. As per the Mobility firmware, DIANARM control firmware also performs safety checks, avoiding the arm to collide with the rover itself or to encounter control singularities.

As for the Mobility System, a feasibility analysis has been performed through the Cheddar Framework.

5. EDUCATIONAL ASPECTS

Due to its educational purpose, DIANA enables students to gain experience in robotic systems from the very first layer of design. The team has independently developed both the ARDITO control stack and the electronic architecture.

While developing all software layers from low-level protocols may be less competitive, it gives students a deep understanding of how entire systems are built and function. This contrasts with merely using existing packages within established frameworks.

This approach cultivates skills highly valued in the industry, as evidenced by the success of former team members in both academia and aerospace industry. Furthermore, a fruitful exchange of ideas with these companies allows us to adapt to evolving industry needs in terms of the skills we develop.

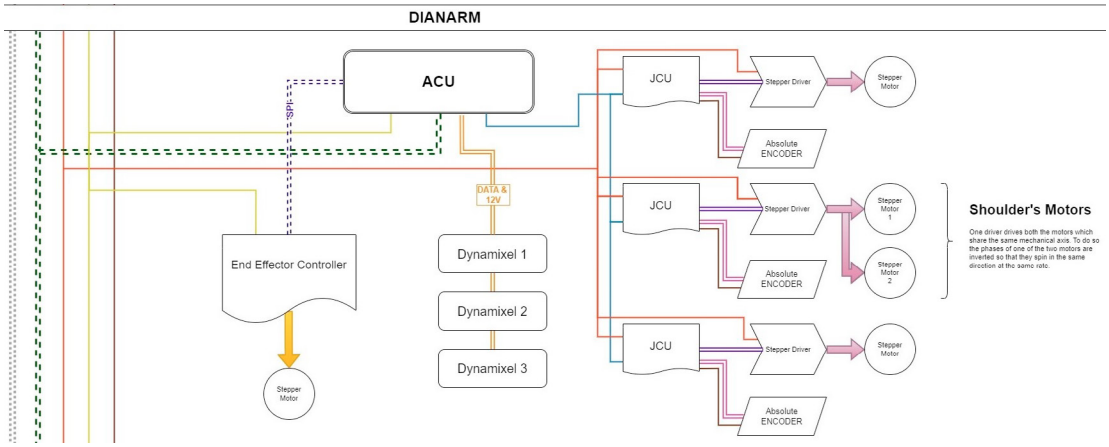

Fig. 6. DIANARM Harnessing

5.1 Lessons Learnt

Being the proposed work entirely developed within the frame of a university student team, several problems arose in the design and integration phase of either a subsystem or an entire system. Due to lectures and exam periods, the organization of the tasks is strictly related to the availability of the members, which often require several months of training before being able to meaningfully contribute to the project. The continuous turnover, unavoidable in a student team, also requires for a dedicated focus on the transfer of the acquired know-how from the senior members to the new ones. On a technical side, many challenges encountered were related to procurement and production issues. The presented electronic architecture is therefore also a consequence of the efforts that were put into designing an easy to maintain and flexible platform, optimizing the dual objective of having a hardware and software system reasonably easy to understand for a beginner and sufficiently expandable and configurable to allow professional-level control.

6. CONCLUSION

The proposed modular architecture offers enhanced flexibility and adaptability compared to traditional monolithic designs, allowing easier upgrades and modifications, which are crucial for the evolving needs of space missions. Moreover, by standardizing the rover's components and employing a modular approach, the overall development and production costs can be significantly reduced. The architecture also supports scalability, enabling the creation of rovers of various sizes and capabilities. It ensures that the rovers can be customized for different tasks, from scientific research to construction and maintenance on planetary surfaces.

REFERENCES

Amazon Web Services (2024). FreeRTOS. `https://freertos.org`.

Bickler, D.B. (1989). Articulated suspension system.

Bosch (1991). Controller Area Network (CAN) Specification. *Robert Bosch GmbH, Postfach*, 50, 15.

Brukardt, R. (2022). How will the space economy change the world? *McKinsey Quarterly*.

ESA (2028). ESA's ExoMars Mission. `https://www.esa.int/Science_Exploration/Human_and_Robotic_Exploration/Exploration/ExoMars`.

European Space Foundation (2024). European Rover Challenge. `https://roverchallenge.eu/`.

Kassel, S. (1971). *Lunokhod-1 Soviet Lunar Surface Vehicle*. RAND CORP SANTA MONICA CA. URL `https://apps.dtic.mil/sti/citations/AD0733960`.

Maxime Ransan, E.M.A. (2006). A collaborative model for astronaut-rover exploration teams. AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before., 52–58.

Mustich, F., Festa, L.M., Stesina, F., Camoriano, R., and Tiboni, G. (2023). Point Cloud-Based Reinforcement Learning for Autonomous Navigation of a Robotic Rover on Planetary Surfaces. In *International Astronautical Congress (IAC) 2023*.

NASA (2011). NASA's Mars Science Laboratory Mission. `https://science.nasa.gov/mission/msl-curiosity/`.

NASA (2020). NASA's Mars 2020 Mission. `https://science.nasa.gov/mission/mars-2020-perseverance/`.

Post, M.A., Yan, X.T., and Letier, P. (2021). Modularity for the future in space robotics: A review. *Acta Astronautica*, 189, 530–547. doi:ISSN:0094-5765.

Robotis Inc. (2024). Dynbamixels. `https://www.dynamixel.com`.

Singhoff, F., Legrand, J., Nana, L., and Marcé, L. (2004). Cheddar: a flexible real time scheduling framework. In *Proceedings of the 2004 annual ACM SIGAda international conference on Ada: The engineering of correct and reliable software for real-time & distributed systems using Ada and related technologies*, 1–8.

Stesina, F. and Corpino, S. (2021). STAR laboratory at Politecnico di Torino: a facility to develop educational space projects. *IFAC-PapersOnLine*, 54(12), 80–87.

STMicroelectronics (2023a). STM32F446RE. `https://www.st.com/en/microcontrollers-microprocessors/stm32f446re.html`.

STMicroelectronics (2023b). STM32H7 Nucleo-144 boards. `https://shorturl.at/j2H8J`.

The RTEMS Project (2024). Real-Time Executive for Multiprocessor Systems. `https://www.rtems.org`.