Preliminary Evaluation of Commercial Off-The-Shelf GPUs for Machine Learning Applications in Space

Ioan Octavian Rad*† ioan.octavian.rad@tum.de
Ramon Maria Garcia Alarcia* ramon.garcia-alarcia@tum.de
Sascha Dengler† sascha.dengler@tum.de
Alessandro Golkar* golkar@tum.de
Chiara Manfletti† chiara.manfletti@tum.de

*Chair of Pico and Nano Satellites and Satellite Constellations
Technical University of Munich (TUM)
Ottobrunn, Germany

†Chair of Space Propulsion and Mobility
Technical University of Munich (TUM)
Ottobrunn, Germany

***Abstract*—This work presents the initial assessment of the performance and power needs of two commercially available Systems-on-a-Chip (SoC) featuring both Central Processing Units (CPU) and Graphical Processing Units (GPU): the NVIDIA Jetson Nano Developer Kit [1] and the NVIDIA Jetson AGX Orin Developer Kit [2]. The objective of this evaluation is to offer an early estimation of the GPUs' performance and technical viability for deploying on-board machine learning tasks in an on-board processing subsystem. This evaluation establishes a baseline for future optimization of on-board processing within resource-limited environments, specifically nanosatellite systems. Monitoring processes are configured to obtain a continuous observation of parameters such as execution and training time, CPU and GPU usages, throughput, performance, components power consumption, temperatures and efficiency. Benchmarking different conditions can provide results useful to determine the requirements, criteria and trade-offs to be considered when implementing such devices. Preliminary findings confirm the feasibility of using the NVIDIA Jetson family of devices for space applications involving demanding data processing or Artificial Intelligence (AI) and Machine Learning (ML) models. Additional emphasis has been put in tailoring the Operating System (OS); by eliminating unnecessary processes irrelevant to the desired pipeline or the tuning of resources through the different power modes is possible to alleviate the requirements on the power and thermal subsystems. Finally, an interest in training those AI/ML models in space is taken: the study case of this work, a Water Electrolysis Propulsion (WEP) controller, requires the ability to retrain the neural network in order to autonomously operate the spacecraft. Considering different configurations of the SoCs' resources has lead to the conclusion that training is also viable.**

I. Introduction

In recent years, the improvements in Machine Learning (ML) algorithms plus the advances in computing power have promoted the adoption of Artificial Intelligence (AI) applications in various industries with the trend promising to keep the rise in activity also in the near future [3], [4]. Additionally, an increase in commercialization and decrease in the price of Commercial Off-The-Shelf (COTS) hardware used to run these implementations have been observed.

In the space sector, AI has become of interest as it is possible to bring autonomy to the spacecraft and minimize the required communications to ground. Thus, components capable of running AI/ML models have to be installed on-board spacecraft in order to take advantage of their full potential. For example, solutions for detection of wildfires already consider and implement devices from the NVIDIA Jetson family on-board of micro satellites [5], [6].

This trend has led to the necessity of performance analyses where both the benchmark and the conditions in which it is executed imitate the space environment. A way to perform the analysis is to use the On-Board Processing Benchmarks (OBPMark) and On-Board Processing Benchmarks for Machine Learning (OBPMark-ML) [7], suites of standardized tests developed by the European Space Agency (ESA) to characterize the performance of the systems in data and image processing as well as AI/ML applications, respectively. OBPMark provides algorithms as sequential, parallel (OpenMP), and CUDA implementations, which allows to compare the traditional SoC with only an ARM processor [8] –an architecture highly used in embedded applications in space and other industries– to the new SoCs that include a GPU as seen in Table I.

TABLE I
OBPMark-ML Object Detection Benchmark

Parameter	Device on Jetson AGX Orin	
	Arm Cortex-A78 CPU	**2048-core Ampere GPU**
Total Time	3 min 10 s	50 s
Inference Time	124 s	18.7 s
Peak Power	19.8 W	16.7 W
Total Energy	3.03 kJ	0.97 kJ

Comparison showing the performance advantage of using a GPU over a CPU for a space application: object detection using a convolutional neural network (CNN) infering a total of 1000 satellital images.

However, many other design choices still free for the user

to control, such as the device's physical environment, the OS, and the monitoring process. Furthermore, as the development of the desired solution advances, it is also necessary to benchmark it in a reproducible manner to predict its execution conditions once in space. Therefore, establishing a methodology to follow is needed as much as the tests to execute.

In this work, a prototype controller (based on the reinforcement learning algorithm DQN) for the control of a Water Electrolysis Propulsion (WEP) system developed by Dengler et al. [9] used to study the feasibility of in-space autonomous decision-making within an optimized Concept of Operations (CONOPS) is deployed and characterized on the COTS GPUs.

II. Methodology

A. Monitoring Workflow

Using the internal sensors available in all the Jetson devices, it is possible to set up a pipeline for measuring and reporting values of interest. The `jetson_stats` library [10] (based on the lower level `tegrastats` command) can monitor the status of the device periodically –every $5\,\mathrm{s}$ for the AGX Orin and every $10\,\mathrm{s}$ for the Jetson Nano, values chosen based on sensors' sampling rate and CPU performance– and obtain readings for the following parameters:

- CPU usage, as a percentage.
- GPU usage, as a percentage.
- Power consumption breakdown by component, in Watts.
- Fan usage, as a percentage.
- Temperatures at different points of the SoC, in Celsius.

A simple server is set up on a RaspberryPi 4B [11] running a monitoring database, Prometheus [12], which fetches –via Ethernet– and stores all the parameters. Furthermore, Grafana [13] is used to filter, navigate, visualize, and oversee the information in real time. Grafana also allows to export the selected data in other formats, such as CSV, for further postprocessing.

This workflow has been designed to shift the workload of storing, visualizing and processing the data away from the devices into a dedicated server, ensuring the readings are less influenced by the monitoring processes. Furthermore, it is feasible to implement in production, which would only require to transmit the recorded data from `jtop`/`tegrastats` as telemetry through a satellite link to a receiving server.

B. Characterization of the Monitoring Workflow

As with any other running process, monitoring comes at the expense of CPU usage and power consumption. In order to characterize the monitoring system's cost an external Rigol DP932A [14] power supply is used. The power supply is configured to provide $5.1\,\mathrm{V}$ and $20\,\mathrm{V}$ for the Jetson Nano and the Jetson Orin AGX, respectively, and to record voltage, current and power values at a rate of $10\,\mathrm{samples/s}$, later filtered to match the sampling rate of the internal monitoring. Three power readings are obtained for each device:

1) External, from the DP932A, with the device on idle without the monitoring process.
2) External, from the DP932A, with the device on idle with the monitoring process running.
3) Internal, from the device, with the device on idle with the monitoring process running.

Subtracting reading 1) from 2), we can obtain the increase in power consumption due to the monitoring process and comparing 3) to 2), we can validate if the values reported by the devices match the demanded power of the board.

TABLE II
Characterization of the Monitoring System

Parameter	Device	
	Jetson Nano	**Jetson AGX Orin**
Cost of monitoring	$0.6\,\mathrm{W}$	$1.0\,\mathrm{W}$
Power supplied minus reported	$0.0\,\mathrm{W}$	$+2.5\,\mathrm{W}$

By running `htop`, we identify that the process takes $<1\,\%$ CPU usage, which is insignificant in terms of power consumption. Therefore, the obtained values in Table II can be attributed to the usage of the Ethernet peripheral and internet connectivity, which was unused during measurements in case 1). Comparing the measured values on the same state, we observe that the Jetson Nano accurately reports the consumed power of the board, while the Jetson AGX Orin reports $2.5\,\mathrm{W}$ less than the actual input power. These values refer to the devices mounted on their corresponding carrier boards and can vary when using custom boards with different peripherals. Moving forward, all power values are obtained from the internal monitoring system unless specified otherwise.

C. Benchmarking Idle State and Execution of WEP Agent

For the Jetson Nano, three OS have been explored:

- RedHawk [15], a Real-Time OS (RTOS) built upon Ubuntu 18.04 with JetPack 4.3.
- NVIDIA's latest OS release for the Jetson Nano, running Ubuntu 18.04 with JetPack 4.6.
- A community release of Ubuntu 20.04 with JetPack 4.6 [16].

And for the Jetson AGX Orin, only official OS releases were mature enough at the time of our testing, two versions of JetPack have been studied:

- Ubuntu 20.04 with JetPack 5.0.2.
- Ubuntu 20.04 with JetPack 5.1.1.

The necessary packages for the WEP controller, introduced in Section I, are installed and configured in order to execute a pretrained version of the agent inside a simulation.

First, each OS and device combination is left unused, on idle, over a period of $6\,\mathrm{h}$ in order to obtain baseline readings. Afterwards, the WEP Agent is executed in a sped up simulation –which includes a model of the WEP and an orbit propagator– where it solves an orbit change problem. The cost of running the simulation is not accounted for, as in a real environment it is not needed.

D. Benchmarking the Training of WEP Agent

The WEP Agent trains by solving tasks in the simulation. Thus, in this case the simulation must be accounted for: a low amount of data (1.3 GB) is loaded into memory and a CPU core must constantly run it. The whole process starts with a fast multicore initialization and setup of the neural network, and the rest of the training requires the single core with the simulation to feed the inputs to the Agent and react to its decisions. Meanwhile, the neural network resides in the GPU, where we observe a utilization between 20 % to 70 %, which can be considered low load and, therefore, low power.

The duration of the training of the DQN is in the order of hours and in this case different behaviours can be observed by modifying the resources available to the process. A simple and safe way to do this is by using Jetson's Power Modes [17], [18], which are predefined configurations with underclocked and disabled resources which considerably reduce power consumption at the cost of performance.

Considering this conditions, we perform trainings and measure performance metrics for the different devices and power mode settings.

III. Results

A. Idle State and Execution of WEP Agent

The execution of the Agent is very fast with minimal resources utilization, resulting in negligible differences from the idle state. For this reason the focus has been shifted to the optimisation and tuning of the OS in order to minimise the baseline operation of the devices.

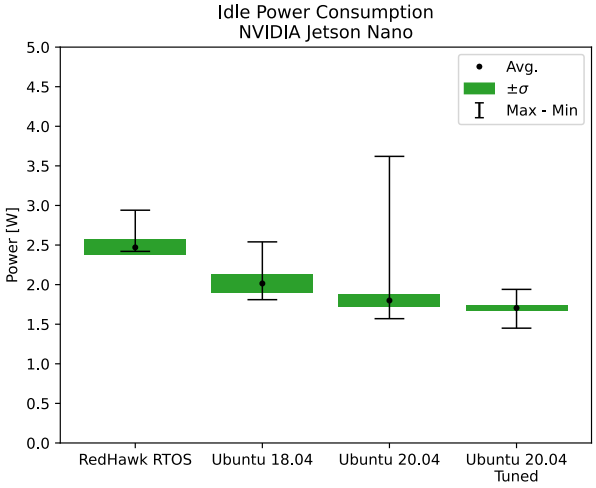

Fig. 1. Idle power consumption comparison of NVIDIA Jetson Nano

Fig. 1 shows power measurements for the three selected OS over a period of 6 h. An additional custom tuned version of Ubuntu 20.04 with disabled graphical user interface and other unnecessary processes –such as `apt-daily` and `fwupd`, which only check for available updates on installed packages– that triggered periodically has been also considered. This tuned version runs with the lowest power consumption, does not provoke unexpected power peaks, and it is still easily maintainable: Ubuntu as the base OS allows for easy installations of NVIDIA's official releases of drivers and packages directly from their websites, compared to any other Linux distribution which would require custom builds and compilations.

As for thermal properties, the tuned version can stay idle without the need of the fan in ambient conditions.

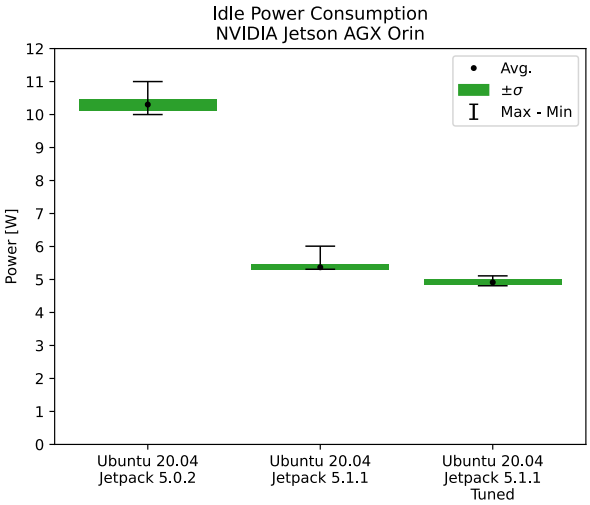

Fig. 2. Idle power consumption comparison of NVIDIA Jetson AGX Orin

From Fig. 2 the improvements in resource management during idle state performed by NVIDIA in successive JetPack releases, which halve the power consumption, are clearly identified. As for the Nano, a tuned version with similar changes is configured and tested. However, even in the optimised OS, the device needs to trigger its fan periodically to cool down in ambient conditions, which would later mean more complex heat dissipation needs in space.

B. Training of WEP Agent

TABLE III
Performance of Training the WEP Agent on Jetson Devices

Device	**Jetson Nano**		**Jetson AGX Orin**	
Power Mode	**MAXN**	**5W**	**MAXN**	**15W**
Duration	3 h 50 min	5 h 0 min	1 h 24 min	1 h 28 min
Avg. Power	3.40 W	2.54 W	10.0 W	9.6 W
Energy	46.8 kJ	46.0 kJ	63.9 kJ	62.6 kJ

*Benchmarks ran on the corresponding tuned OS of each device.

In the Jetson Nano section in Table III, it can be seen how the underclocking of the CPU and GPU has drastically increased the required time to train the model. However, the reduction in power consumed leads to an equal amount of expended energy per training. On the other hand, for the Jetson AGX Orin, the underclocking does not affect the total duration, and the observed power savings are only achieved by disabling 8 of the 12 CPU cores, which would have remained idle during the majority of the training. This shows that keeping unused

resources available when they are not required is a mistake that results in an unnecessary energy consumption.

The ranges of measured power values appear in the following Fig. 3 and Fig. 4. Another benefit of using lower power modes is reducing variance and, most importantly, instantaneous power peaks, conditions which simplify the design of the device's power supply as a lower maximum current can be considered.

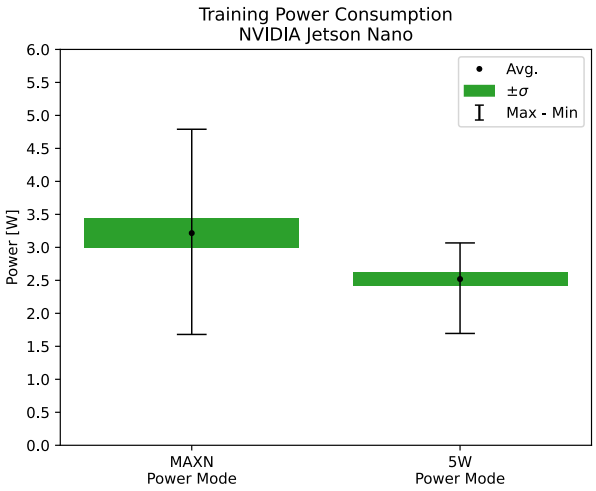

Fig. 3. Training power consumption comparison of NVIDIA Jetson Nano Power Modes

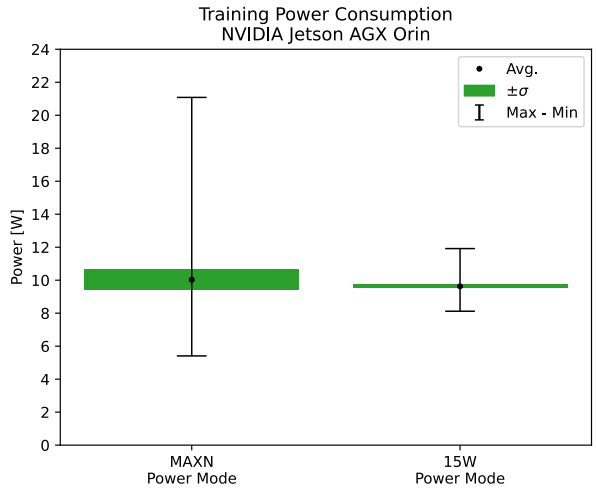

Fig. 4. Training power consumption comparison of NVIDIA Jetson AGX Orin Power Modes

As for thermal results in ambient conditions, we observe that the Jetson Nano in the 5W mode does not require the actuation of the fan at any time, and the device's temperature remained under $47.5\,^{\circ}\mathrm{C}$. On MAXN mode, it did turn on for an interval of $15\,\mathrm{min}$ as the temperature reached $55.0\,^{\circ}\mathrm{C}$ (triggering temperature for the standard configuration of the fan). However, on the AGX Orin, the fan was primarily turned on throughout each training, but always at low speeds ($20\,\%$ to $50\,\%$ of maximum RPM).

C. Power Budgets

By also measuring power consumption during SC7 sleep with an external device, the DP932A, in this case, we can build comprehensive power budgets.

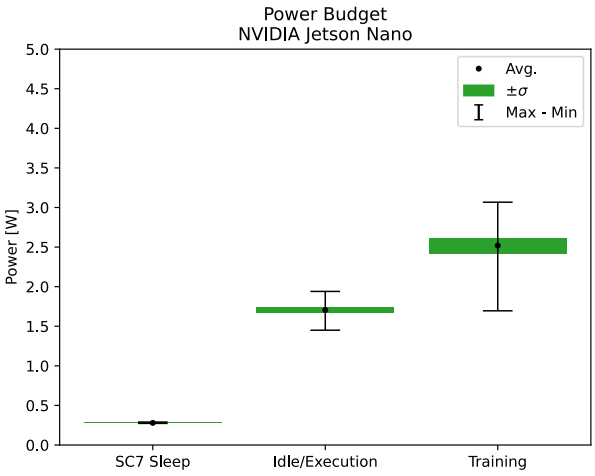

Fig. 5. Power budget of NVIDIA Jetson Nano

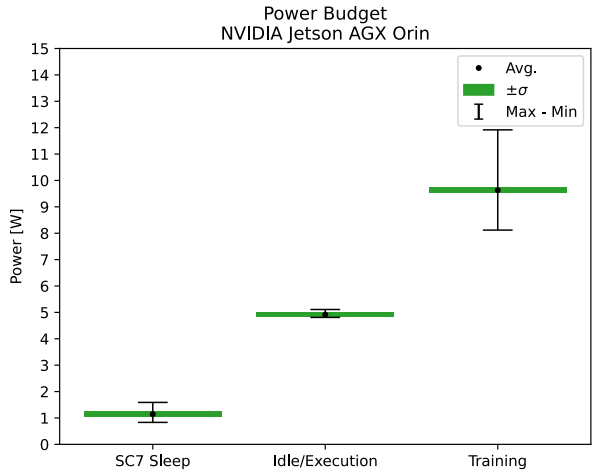

Fig. 6. Power budget of NVIDIA Jetson Nano

Figs. 5 and 6 show the three expected states for each device when deployed.

- Deep sleep for very low power consumption.
- The "Idle/Execution" state as nominal consumption.
- Training requires the most power but is also only a punctual event that happens from time to time.

IV. Discussion

From the obtained results, we can envision how the device will generally operate on-board of a satellite. While in eclipse,

a complete shutdown may be possible, but entering deep sleep may be preferable as a full power off leads to a slower start-up and possible data loss. During the sun phase, we can consider the "Execution" state as nominal consumption and account for the training phase only if it is required or we have excess power.

Examining the final power budgets (Figs. 5 and 6) and analysing the numbers we can contemplate the usage of the Jetson Nano as a Data Handling (DH) computer on-board of small factor satellites such as 12U CubeSats, as it could perfectly fit the typical budget of these. Furthermore, the reduction in power peak loads will directly translate in a simpler design of the power supply –as it has to support lower current– and an improvement in battery life or the possibility to choose a smaller one.

However, the Jetson AGX Orin would require a bigger platform to operate, not only because of the numbers obtained in this work but also because it has been tested at its lowest power mode. Lower specifications devices of the same family that come at a lower price tag would be better suited for micro satellites and, taking it to the extreme, it could be argued that CPUs are enough for certain applications if their execution load is very minimal, for example, when only executing a pretrained and optimised neural network.

Moreover, if we consider the typical sun phase durations in Low Earth Orbit (LEO) we come across the fact that we cannot perform a full training from start to finish without interruption. It is necessary to consider intermediate states of the model and store them before entering deep sleep. These intermediate models can be reloaded when energy production ramps up again –or in case of unexpected errors– and training can continue. Only when convergence is reached, the final model can be considered for execution and the training process completed.

Finally, the thermal analysis and radiation testing have yet to be conducted and extended. In terms of thermal conditions, it shows promising signs, especially for the Jetson Nano, as little to no active cooling has been required in ambient conditions. Also with good results, radiation tests such as the one by Rodriguez-Ferrandez et al. on the NVIDIA Jetson Xavier [19] can already give an idea on how these type of devices will act and what errors to look for when operating in the harsh space environment.

V. Conclusions

The presented work shows the preliminary approach to benchmark these types of devices and tailor them for space environment specific needs with some changes to their software. Furthermore, the obtained results can help determine the desired requirements for our subsystem and process of implementation.

Section II explains the idea behind a monitoring pipeline useful in both testing and production environments. Accounting for the effects of the monitoring processes running on the device is also a crucial step to obtain accurate results. Understanding the environment and hardware in which our application runs is necessary to know which parameters can be modified from benchmark to benchmark and obtain a variety of results.

Subsection III-A shows how newer software comes with more optimized power consumption, although other characteristics such as package support and maturity can cause a trade-off. It also states the importance of monitoring, understanding, and tuning the different processes running on the OS to avoid unnecessary workload.

Subsection III-B shows the comparison of different performance values measured that can help understand the trade-offs that come into play when comparing devices/configurations. Other applications or models may have different relevant parameters, such as throughput, and choosing which ones to measure is also a decision. Moreover, power measurements in this preliminary phase are helpful to build up the power budgets for the subsystems.

VI. Future Work

As for future work, the hardware side of the devices must be analysed. A custom carrier board that can fit into a micro sat –or even following the CubeSat standard– must be designed with only the necessary peripherals to run the device in space, reducing, even more, the maximum required power. This will also lead to a more in-depth tailoring of the OS and probably a custom compilation.

The custom board will also require a heat spreader that can extract the heat generated, make it testable in a thermal vacuum chamber, and give more insights into the operational conditions of the device. Radiation testing could also be considered, however it is difficult to justify the installation of a viable test bench for a couple devices due to difficulty and cost.

Moreover, as this is a constantly improving field, newer software and devices will have to be benchmarked and compared, bringing more performance at a cheaper cost. An honorable mention is the Jetson Orin Nano released during the writing of this work. It comes with newer software and promises more power efficient resources than the Jetson Nano.

References

[1] S. Cass, "Nvidia makes it easy to embed ai: The jetson nano packs a lot of machine-learning power into diy projects - [hands on]," *IEEE Spectrum*, vol. 57, no. 7, pp. 14–16, 2020.

[2] M. Barnell, C. Raymond, S. Smiley, D. Isereau, and D. Brown, "Ultra low-power deep learning applications at the edge with jetson orin agx hardware," in *2022 IEEE High Performance Extreme Computing Conference (HPEC)*, 2022, pp. 1–4.

[3] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[4] R. Pugliese, S. Regondi, and R. Marini, "Machine learning-based approach: Global trends, research directions, and regulatory standpoints," *Data Science and Management*, vol. 4, pp. 19–29, 2021.

[5] K. Thangavel, D. Spiller, R. Sabatini, S. Amici, S. T. Sasidharan, H. Fayek, and P. Marzocca, "Autonomous satellite wildfire detection using hyperspectral imagery and neural networks: A case study on australian wildfire," *Remote Sensing*, vol. 15, no. 3, 2023. [Online]. Available: https://www.mdpi.com/2072-4292/15/3/720

[6] S. Shah, T. Grübler, L. Krempel, S. Ernst, F. Mauracher, and S. Contractor, "Real-time wildfire detection from space – a trade-off between sensor quality, physical limitations and payload size," vol. XLII-2/W16, 09 2019, pp. 209–213.

[7] D. Steenari, L. Kosmidis, I. Rodriguez-Ferrandez, A. Jover-Alvarez, and K. Förster, "Obpmark (on-board processing benchmarks) – open source computational performance benchmarks for space applications," 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:250526971

[8] S. B. Furber, *ARM system-on-chip architecture*. pearson Education, 2000.

[9] S. Dengler, A. W. Y. Otero, and C. Manfletti, "Optimising conops of water electrolysis propulsion systems using deep q-learning," in *Aerospace Europe Conference 2023 Joint 10th EUCASS – 9th CEAS Conference*.

[10] R. Bonghi, "Jetson stats," https://github.com/rbonghi/jetson_stats, 2022.

[11] H. D. Ghael, L. Solanki, and G. Sahu, "A review paper on raspberry pi and its applications," *International Journal of Advances in Engineering and Management (IJAEM)*, vol. 2, no. 12, p. 4, 2020.

[12] J. Turnbull, *Monitoring with Prometheus*. Turnbull Press, 2018.

[13] M. Chakraborty and A. P. Kundan, "Grafana," in *Monitoring Cloud-Native Applications: Lead Agile Operations Confidently Using Open Source Software*. Springer, 2021, pp. 187–240.

[14] RIGOL, *DP900 User Guide*, https://beyondmeasure.rigoltech.com/acton/attachment/1579/f-8bcd4294-be35-42c8-beda-dc121d720dca/1/-/-/-/-/DP900_UserGuide_en.pdf, 03 2022.

[15] e. a. Baietto, Jason, "Real-time linux: The redhawk approach," in *Concurrent Computer Corporation White Paper*, 09 2008.

[16] QEngineering, "Jetson nano with ubuntu 20.04 os image," https://github.com/Qengineering/Jetson-Nano-Ubuntu-20-image, 2022.

[17] NVIDIA, *Jetson Nano Supported Modes and Power Efficiency*, https://docs.nvidia.com/jetson/archives/l4t-archived/l4t-3273/index.html#page/Tegra%20Linux%20Driver%20Package%20Development%20Guide/power_management_nano.html#wwpID0E0FL0HA, 2022.

[18] ——, *Jetson AGX Orin Supported Modes and Power Efficiency*, https://docs.nvidia.com/jetson/archives/r35.1/DeveloperGuide/text/SD/PlatformPowerAndPerformance/JetsonOrinNxSeriesAndJetsonAgxOrinSeries.html?highlight=nvpmodel#supported-modes-and-power-efficiency, 2022.

[19] I. Rodriguez-Ferrandez, M. Tali, L. Kosmidis, M. Rovituso, and D. Steenari, "Sources of single event effects in the nvidia xavier soc family under proton irradiation," in *2022 IEEE 28th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2022, pp. 1–7.