ICT-56-2020 - Next Generation Internet of Things

D 4.4
Final report on cognitive IoT hardware platform and microserver development

Document information	
Contract number	957197
Project website	www.vedliot.eu
Dissemination Level	PU
Nature	R
Contractual Deadline	31.01.2024
Author	Rene Griessl (UNIBI)
Contributors	Kevin Mika (UNIBI), Marco Tassemeier (UOS), Mario Porrmann (UOS), Micha vor dem Berge (CHR), Stefan Krupop (CHR), Gunnar Billung-Meyer (CHR), Erik Funke (CHR), Karol Gugala (ANT), Jens Hagemeyer (UNIBI), Martin Kaiser (UNIBI)
Reviewers	Pedro Petersen Moura Trancoso (CHALMERS), Karol Gugala (ANT)

The VEDLIoT project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957197.

Changelog		
v0.1	2023-11-15	Initial draft
v0.2	2023-12-20	First input included
v0.3	2024-01-03	Added realized microservers
v0.4	2024-01-15	Ready for internal review
v1.0	2024-01-29	Final document

Table of contents

Executive Summary

The final report on cognitive IoT hardware platform and microserver development describes the architecture of the VEDLIoT hardware platform, refining and detailing the overview of the platform, which was reported in D2.2. The document reports the final state of the work on the hardware platform. The VEDLIoT hardware platform covers the entire spectrum of IoT systems, from embedded implementations to cloud-based systems. While this report discusses the improvements made to the cloud platforms RECS|Box and t.RECS, the primary emphasis is placed on the u.RECS platform. This platform, developed entirely within VEDLIoT, serves as the crucial facilitator for deep learning applications at the far edge.. It supports a wide range of low-power microservers based on industry-established COM standards such as NVIDIA Jetson, SMARC, CM4 and AMD Kria. In addition, the architecture offers flexible interfaces to include additional accelerators, e.g. via USB or M.2, as well as a wide range of communication and sensor interfaces. Besides the RECS server systems, microserver developments within VEDLIoT are discussed in Chapter 4, complementing available microservers from the 3rd party providers.

1 Introduction

This deliverable deals with the detailed design of the VEDLIoT hardware platform, particularly the server systems u.RECS, t.RECS, and RECS|Box shown in Figure 1, as well as the different microservers, developed and used in the project. As the general concept of the platform has already been described in deliverable D2.2, this deliverable focuses on the specific architecture details for both, platforms and microservers. Furthermore, as the VEDLIoT hardware platform is an open platform, also relevant third party development are mentioned, which are either already available on the market or will be available soon. The deliverable describes not just the hardware architecture but also presents firmware and mechanical integration aspects.

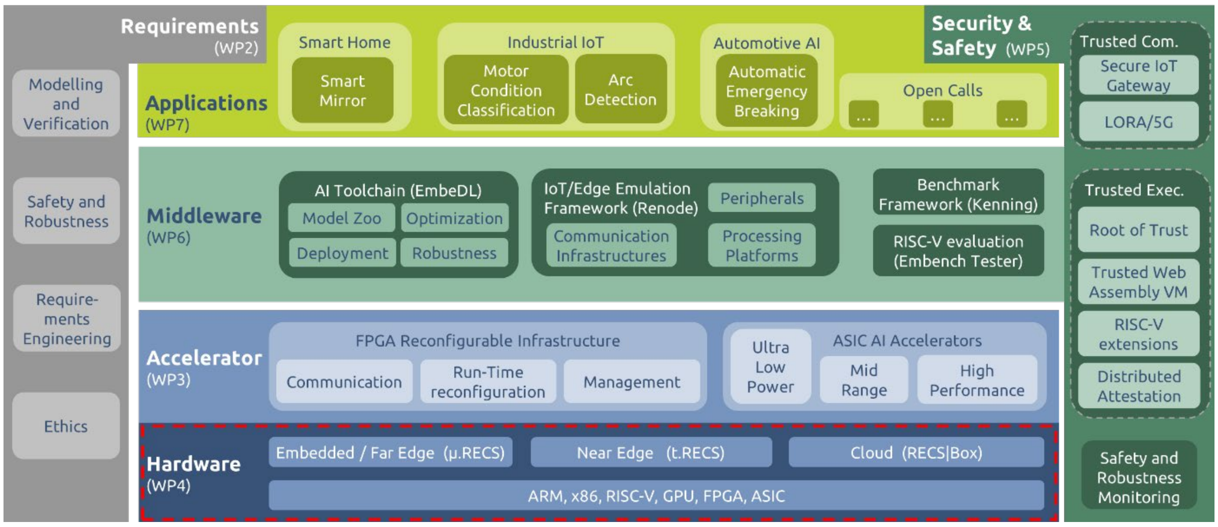
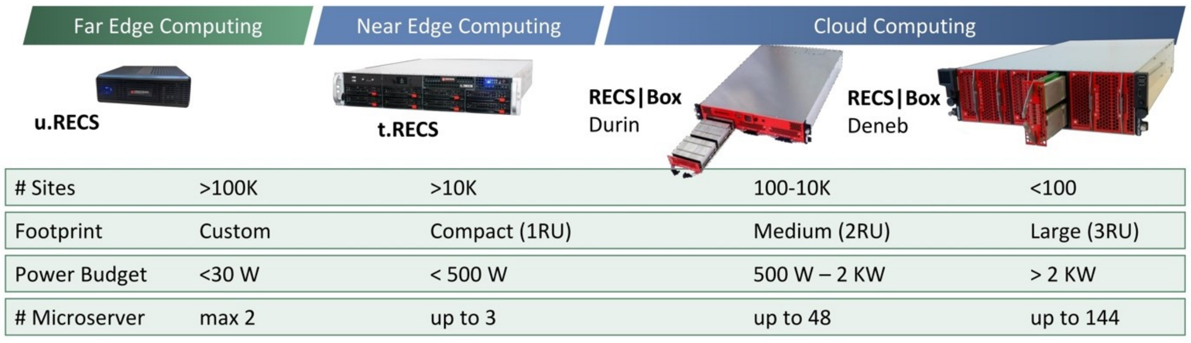


Figure 1: Overview of the VEDLIoT project. This deliverable (D4.4) details platform and microserver developments in WP4.

The rest of this deliverable is structured as follows: Chapter 2 gives an overview of the VEDLIoT hardware platform, in particular the developments regarding the LP Baseboard Orin NX for the RECS|Box, as well as the t.RECS developments. Chapter 3 provides a comprehensive insight into the u.RECS architecture, discussing interfaces, communication, firmware aspects, and notable advances in deep learning acceleration, showcasing the platform's capabilities in pushing the boundaries of DL applications. Chapter 4 delves into the development of new microservers, specifically focusing on ARVSOM and VERSAL Edge.ai COM. This section provides a detailed examination of the intricacies of these microserver technologies, shedding light on their advancements and role in shaping the landscape of microserver development.

2 VEDLIoT Hardware Platform

The VEDLIoT hardware platform is a joint base for the developments within this project. A wide range of AIoT applications can be addressed using a flexible communication infrastructure and exchangeable microservers. Figure 2 shows the RECS platforms covering application domains from embedded/far-edge to cloud computing. All platforms commonly target heterogeneous computing with tightly coupled microservers. The cloud computing platform RECS|Box consists of either 3 rack units (Deneb) or 2 rack units (Durin) and aims for high-density applications using over 100 microservers with high-bandwidth communication requirements. t.RECS houses up to 3 microservers and focuses on edge computing scenarios with low-latency demands like VEDLIoT's SmartMirror [1] use-case or 5G base stations in the automotive [1] use-case. As a new development from scratch, u.RECS rounds off the range of the RECS family towards low-power and compact embedded computing.



	u.RECS	t.RECS	RECS|Box Durin	RECS|Box Deneb
# Sites	>100K	>10K	100-10K	<100
Footprint	Custom	Compact (1RU)	Medium (2RU)	Large (3RU)
Power Budget	<30 W	< 500 W	500 W – 2 KW	> 2 KW
# Microserver	max 2	up to 3	up to 48	up to 144

Figure 2: Overview of RECS platforms, including the new u.RECS developed within VEDLIoT

RECS|Box and t.RECS, the platforms available at the beginning of VEDLIoT, have already been described in detail, in Deliverable D2.2 [2]. This chapter focuses on the development since Deliverable D4.1 [3] and addresses hardware extensions and re-designs, which put RECS|Box (section 2.1), t.RECS (section 2.2) and their common firmware and management software (section 2.3) beyond the state of an early prototyping system. Showcasing these systems in Bielefeld testbed and Christmann testbed as well as using it in the automotive use case raises TRL-level to TRL6.

2.1 RECS|Box

RECS|Box is the cloud computing system of the RECS family. A more in-depth description is available in Deliverable D2.2, Chapter 3.1. RECS|Box provides the highest integration density and compute performance in the RECS family, being the high-end platform when it comes to scaling up applications. A key feature used in the past is the ability to populate the same microservers in the whole range of computing systems, e.g. COM Express modules can be plugged into the t.RECS as well as the RECS|Box Durin and Deneb. This accelerates porting of applications from the edge to the cloud. In VEDLIoT, a big focus is put on the new far-edge platform u.RECS, which comes with smaller microserver form factors like SMARC [4] and NVIDIA Jetson NX [5]. To deliver fast scaling capabilities, as done in the past, the existing LP Baseboard that hosts 16 NVIDIA Jetson TX2 is modified to host 16 NVIDIA Jetson Orin NX. This is also of interest for the SMARC form factor, but unfortunately, the mechanical characteristics of SMARC require a basic re-design of the LP Baseboard, which isn't the case for the Orin NX.

2.1.1 LP Baseboard Orin NX

In Deliverable D4.1 this Baseboard was named LP Baseboard Xavier NX. But since the NVIDIA Orin NX supports the native RECS|Box supply voltage of 12 V we decided to simplify the design and skip the 5 V regulators that would have been needed for the NVIDIA Xavier NX. Apart from that the LP Baseboard Orin NX as described in D4.1, it is derived from the existing LP Baseboard. A significant improvement of the new Baseboard is that the Orin NX modules are capable of PCIe Gen.3, which doubles the PCIe communication bandwidth from 16 Gbit/s to 32 Gbit/s. The mechanical dimensions of Orin NX modules are slightly smaller than that that of the TX2, which ensures that the integration density of 16 modules can be kept on the new PCB. Figure 3 shows the LP Baseboard that can carry up to 16 NVIDIA Jetson TX2.

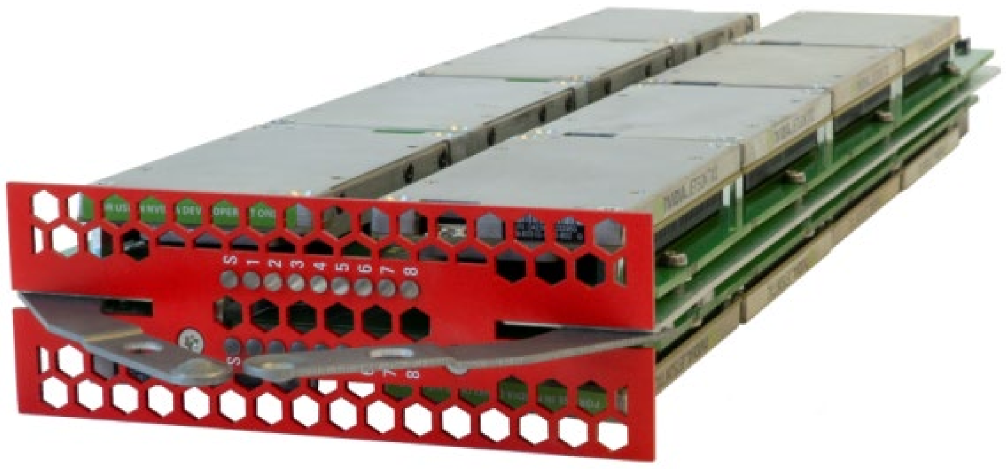


Figure 3: RECS|Box LP Baseboard assembled on mechanical carrier

The assembled LP Baseboard consist of two PCBs (Master, Slave) and the mechanical carrier. The LP Baseboard Master is longer and carries the backplane connectors on that extra space. An additional connector in between the two PCBs connects the LP Baseboard Slave.

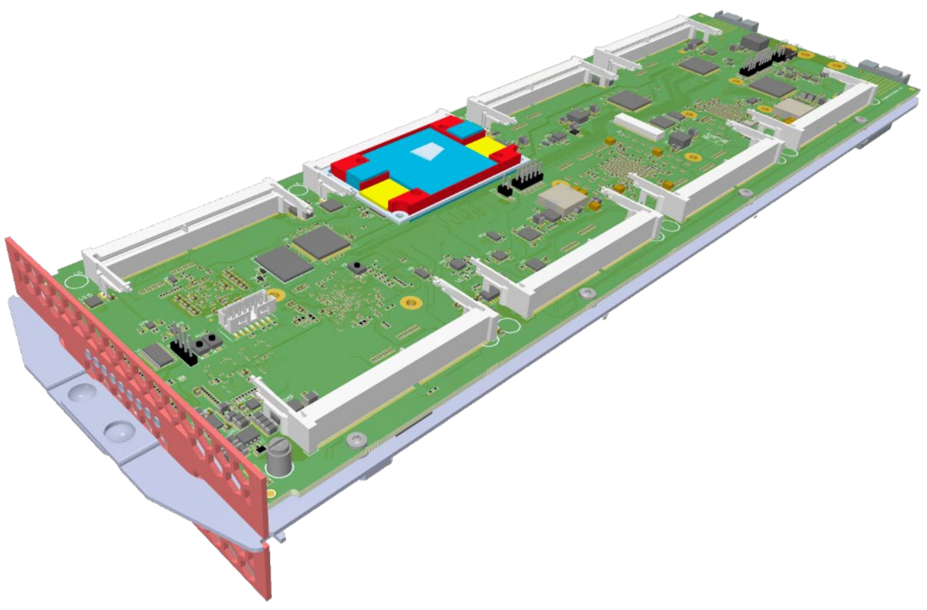


Figure 4: Rendering of LP Baseboard Orin NX

The rendering of the new LP Baseboard Orin NX (see Figure 4) shows the Baseboard with the connectors for the Jetson NX series as well as on Jetson Orin NX inserted to one slot of the Baseboard.

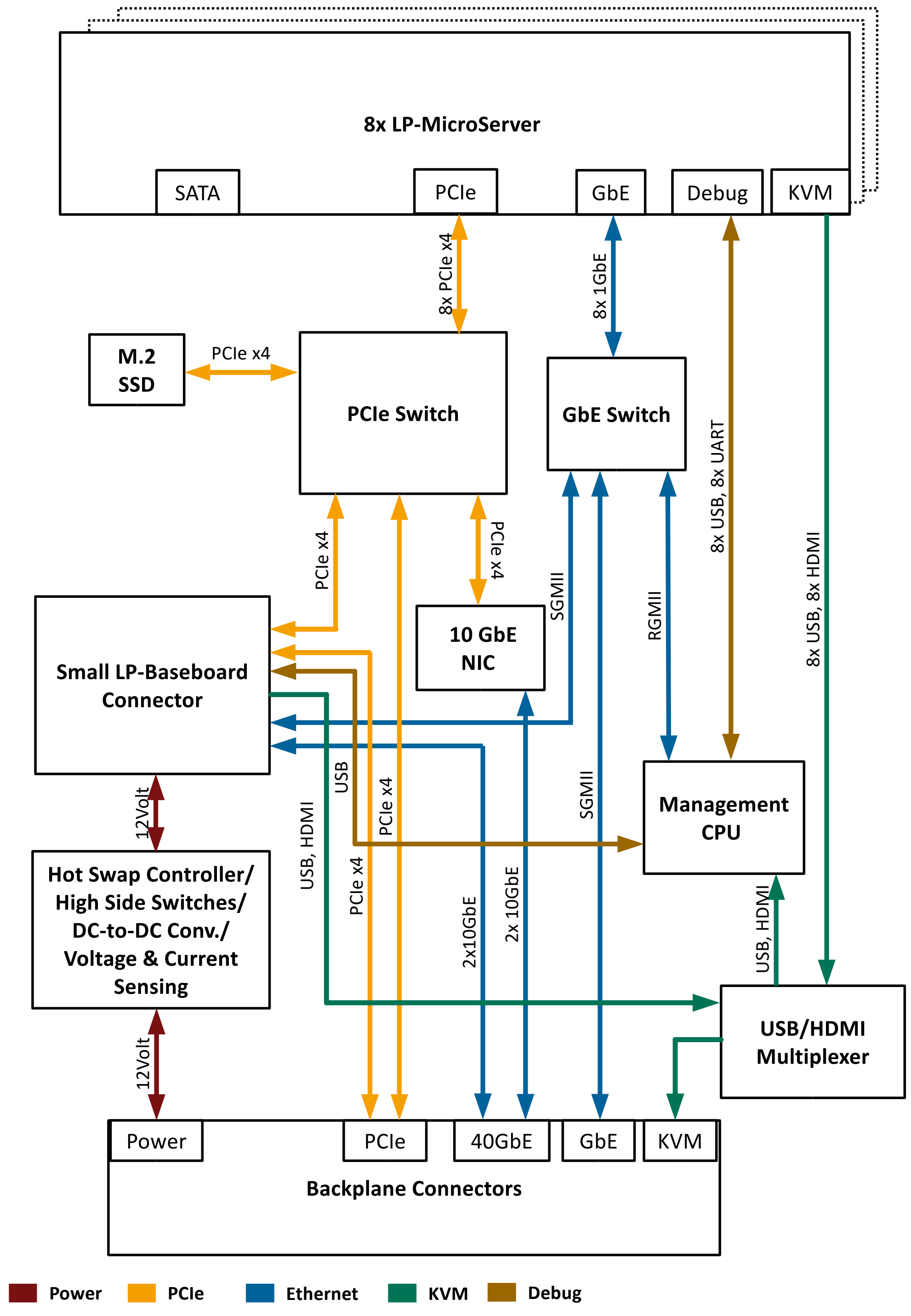


Figure 5: Block diagram of LP Baseboard Master PCB

The block diagram in Figure 5 shows all communication paths and management signals on the Master PCB. The Slave isn't shown here because it looks very similar except not including the management CPU and backplane connectors. The architecture of the LP Baseboard is kept reusing as much as possible from the existing design. As seen in Figure 6, all interfaces connected to the NVIDIA Jetson TX2 on the LP Baseboard can be equally connected to the NVIDIA Orin NX modules on the LP Baseboard Orin NX.

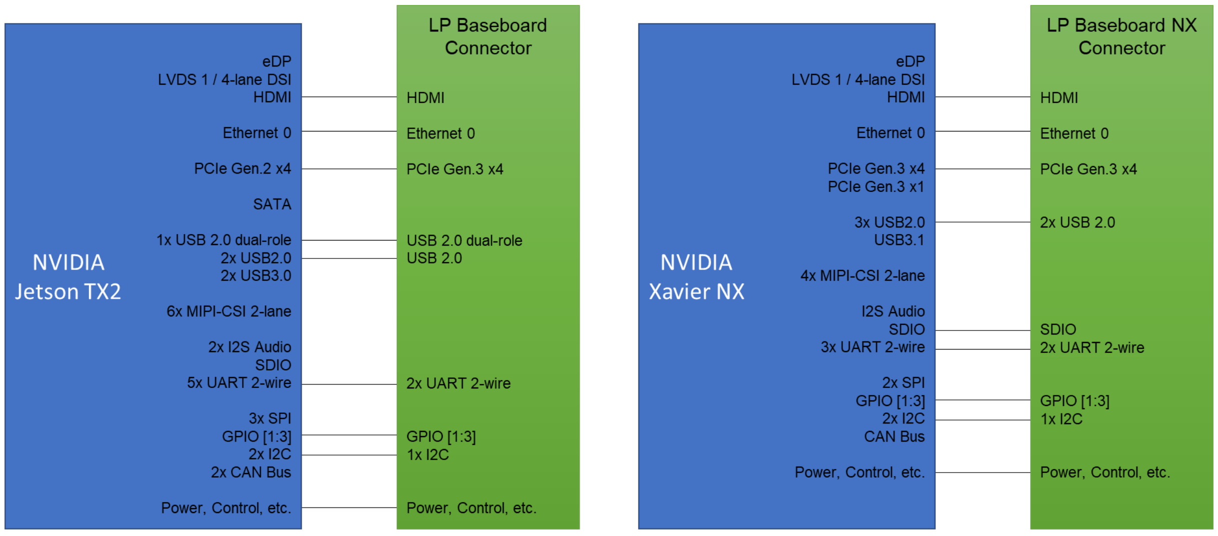


Figure 6: Comparison of interfaces provided by Jetson TX2 and Xavier NX/Orin NX

The following list shows the key features of the LP Baseboard Orin NX:

- Up to 16 NVIDIA Jetson Orin NX modules
  - Each connected to PCIe-Switch via PCIe x4 Gen.3
  - Direct PCIe Host-2-Host communication (32 Gbit/s)
  - Virtualized 10 GbE interface via PCIe-Switch
- Separate GbE management network
- iKVM and KVM from each module
- M.2 Storage attached via PCIe-Switch
- Power monitoring of each module

2.1.2 Challenges

The LP Baseboard design within VEDLIoT encountered difficulties due to the unavailability of Nvidia Orin NX modules until March 2023, resulting in modification of the initially planned design schedule. This was compounded by the need for NVMe storage for each module, adding to the existing challenges. We have been seeking multiple solutions for the NVMe issue. One approach involves utilizing SR-IOV to create a virtual environment for sharing NVMe resources efficiently among the Nvidia Orin NX modules. Additionally, we have considered a design change, which involves removing the LP Baseboard Small that accommodates 8 modules. This alteration aims to integrate 8 physical NVMe slots, ensuring adequate external storage for the remaining 8 modules. Despite these challenges, we've completed the design phase for the LP Baseboard Orin NX.

2.2 t.RECS

Deliverable 2.2 focused on the overall system architecture, including audio/video, peripheral USB support and the board management. This section discusses more details on the communication infrastructure and the different possible heterogeneous combinations. Recently, a new revision of t.RECS was manufactured, addressing several issues with the power supply and the system's overall stability (see Figure 7).

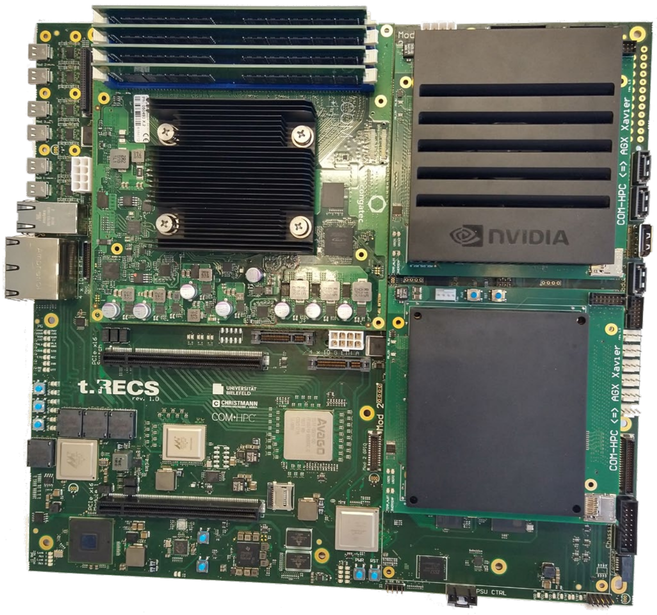


Figure 7: Current revision of the t.RECS PCB with three microservers attached

2.2.1 PCIe communication infrastructure

Figure 8 shows the t. RECS's high-speed, low-latency communication infrastructure between the three different microservers. t.RECS supports one COM-HPC [6] microserver in the type D format (160x160mm) and two microservers following the COM-HPC client specification. A dedicated PCIe Switch interconnects all three microservers with 8 PCIe Gen.3 lanes each and allows for a highly flexible interconnection of the microservers, which can be reconfigured at run-time.

Figure 9 a) and b) show two possible configuration options that leverage the switched PCIe interconnects. A classical CPU-based clustering is depicted in a), where each microserver is operated as a PCIe root complex (RC). As a unique feature of the PCIe switch, all three microservers can exchange data by a host-2-host communication. A CPU-centric approach is drawn in b) including two accelerators connected via PCIe as endpoints (EP).

The configurations c) and d), depicted on the right side of Figure 9, leverage the direct high-speed interconnections between neighbouring microservers resulting in a ring-topology. In c) these interconnections are used as PCIe interconnects, where in d) the protocol can be set to use more low-level protocols like the AMD Aurora [7] protocol. The Aurora protocol interconnects FPGAs with less overhead, resulting in higher bandwidth and lower latency than the packet-based PCIe.

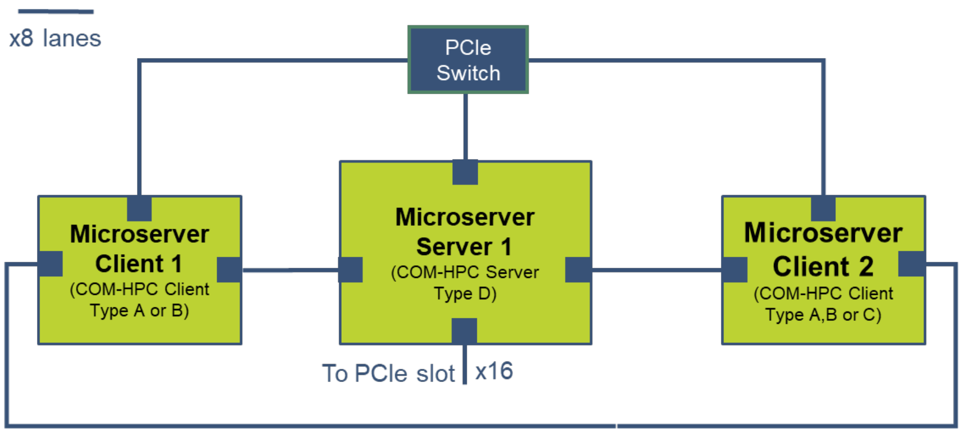


Figure 8: t.RECS PCIe communication architecture, based on two independent infrastructures: Switched PCIe and direct interconnection to neighbouring modules

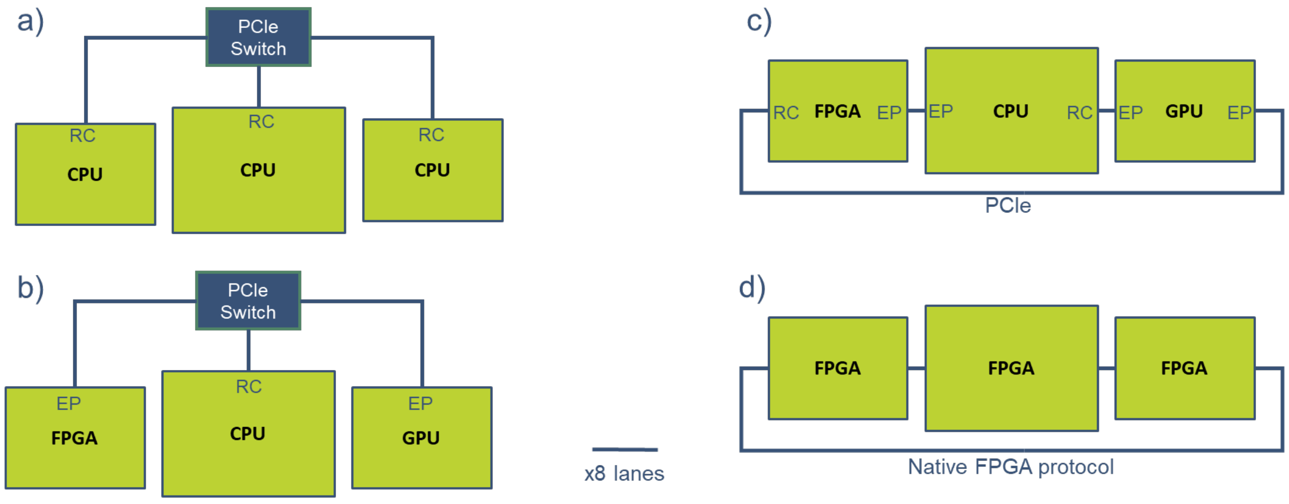


Figure 9: Selected t.RECS configurations of different and partially heterogeneous target architectures. All depicted connections are based on 8 bi-directional lanes

2.2.2 Supported Microservers

t.RECS uses the COM-HPC [6] standard, officially released in January 2021. Since this standard is still emerging, only a few microservers are currently available off-the-shelf. Figure 10 lists the microservers that are available for the t.RECS at the end of the VEDLIoT project. As a member of the standardization committee, Bielefeld University works closely with various manufacturers. Consequently, it was provided with engineering samples of COM-HPC modules with x86 CPUs from Intel. FPGA modules were developed together with Trenz Electronic [8], another Bielefeld University partner. Among the FPGA modeules is a Client module Size B with AMD UltraScale+ FPGA. Another Trenz module with Intel FPGA (Server Size D) is expected in 2024.

To compensate for the availability problems, Bielefeld University developed adapter PCBs that map COM Express to COM-HPC and thus ensure the broad availability of modules. No GPU-based modules have been planned so far. A COM-HPC to Xavier AGX adapter was already built since the NVIDIA Xavier AGX modules are highly relevant for VEDLIoT, especially for the Smart Mirror use-case.

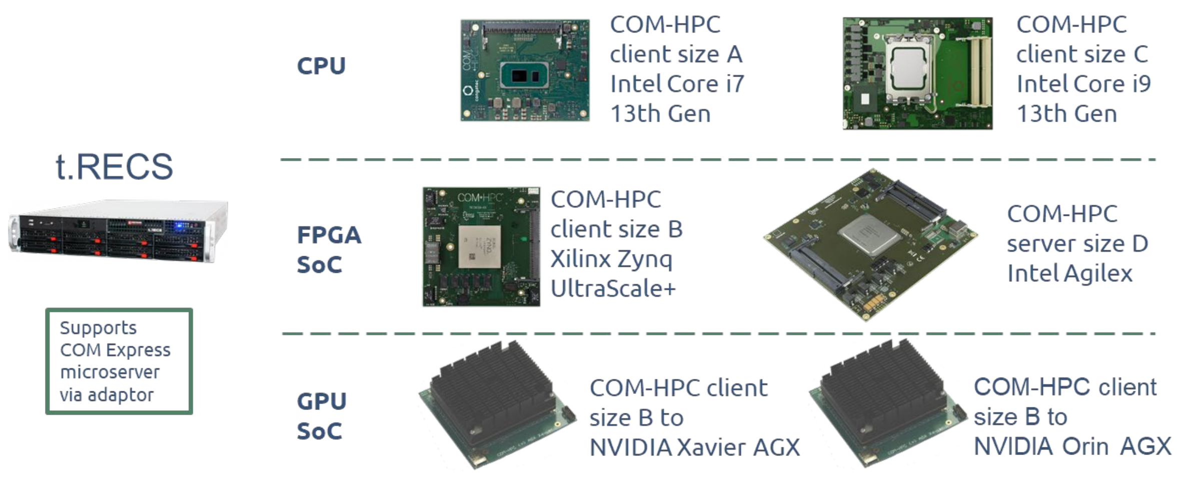

t.RECS
Supports COM Express microserver via adaptor
CPU
COM-HPC client size A Intel Core i7 13th Gen
COM-HPC client size C Intel Core i9 13th Gen
FPGA SoC
COM-HPC client size B Xilinx Zynq UltraScale+
COM-HPC server size D Intel Agilex
GPU SoC
COM-HPC client size B to NVIDIA Xavier AGX
COM-HPC client size B to NVIDIA Orin AGX

Figure 10: Supported Microserver and target architectures of t.RECS

Within VEDLIoT the NVIDIA Orin AGX modules became available. The COM-HPC to Xavier AGX adapter could be reused for integration of the Orin AGX into the t.RECS system. Table 1 shows a brief comparison of both NVIDIA Jetson AGX modules. NVIDIA's benchmarks show a performance increase of 6x for AI applications, when upgrading from the Xavier to the Orin, which makes the microserver highly interesting for VEDLIoT. The Xavier AGX adapter board needed some changes to support the Orin modules. One of the significant differences regarding interfaces are the doubled PCIe lanes, allowing the Orin to simultaneously communicate in parallel via the PCIe switch and the direct connectors.

Table 1: Brief comparison of Jetson AGX Xavier [9] and its successor Jetson AGX Orin [10]

	Jetson AGX Xavier	**Jetson AGX Orin**
CUDA/ tensor cores (architecure)	512/64 (Volta)	2048 / 64 (Ampere)
CPU	8x NVIDIA Carmel ARMv8.2	12x Nvidia ARM Cortex A78A v8.2@2GHz
PCIe	8x PCIe Gen4	2x8 (or 1x8 + 2x4) Gen4
Memory	64 GByte, 256 Bit LPDDR4 136 GByte/s	32GByte, 256 Bit LPDDR5, 204 GByte/s
Deep Learning Accelerators	2x NVDLA	2x NVDLA v2.0
Technology/ Transistors	20nm/ ~7 billion	8nm /~17 billion
Storage	32 GByte eMMC	64 GByte eMMC

2.2.3 Microserver configurations for VEDLIoT

The interchangeability of the t.RECS's microserver offers excellent advantages for VEDLIoT, especially to adapt to different use-cases. The Smart Mirror use-case [1] benefits considerably from this feature. Figure 11 shows how the Smart Mirror hardware configuration can be extended by newer and additional components in several iterative development steps. The current base configuration with two PCIe-coupled Xavier AGX as shown in Figure 11 a. One module takes care of the sensor capturing and video input, control logic and the face recognition neural network. Additional neural networks for object- and gesture detection are off-loaded to the second Xavier AGX. In the current work, these neural networks are being mapped to an AMD UltraScale+ FPGA with the expectation of being significantly faster and at the same time more energy efficient (configuration in Figure 11 b). A second FPGA can also be added using the switched PCIe interconnects. In addition, the NVIDIA Jetson Orin AGX provides an efficiency increase, as well as an performance increase (configuration in Figure 11 c). Since the Orin has additional PCIe lanes [10], these can also be interconnected directly, without the interconnected PCIe switch, which results in a reduction of the power consumption by about 10 Watts (Figure 11 d-f).

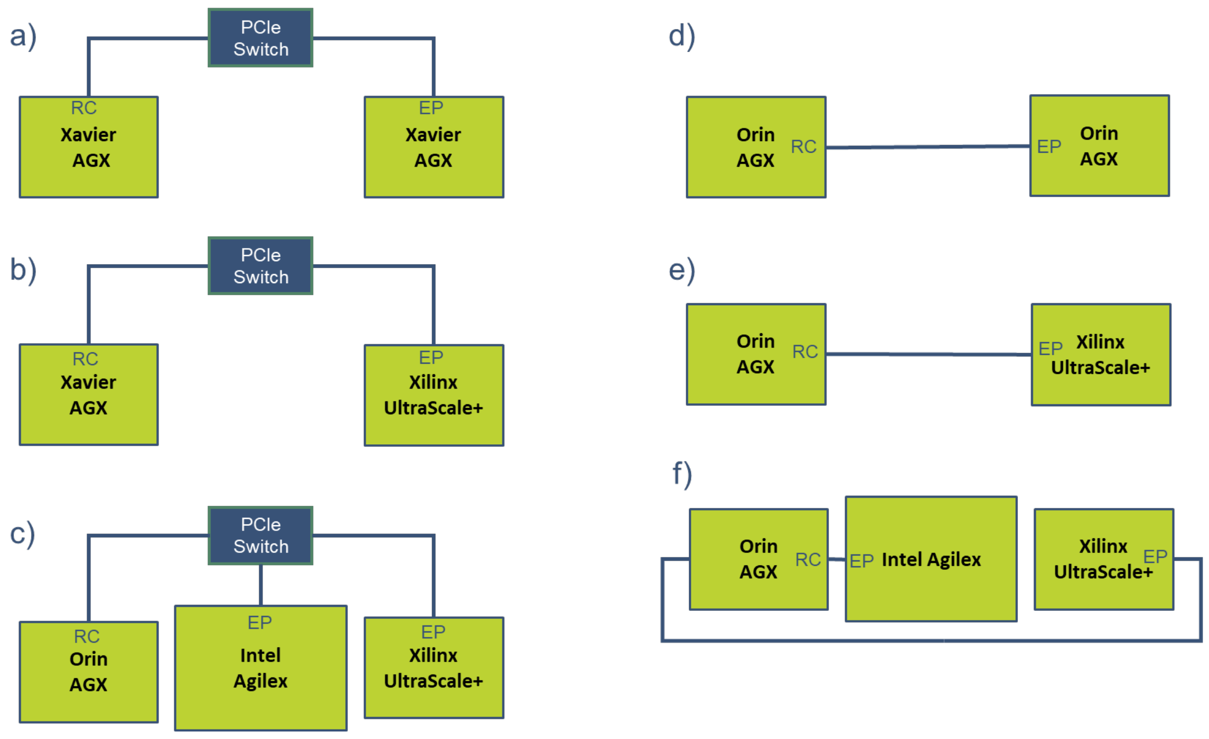


Figure 11: Selected implementation variants of the Smart Mirror using the t.RECS communication infrastructure. a) - c) include the central PCIe switch, d) - f) are with direct PCIe interconnects between microservers

2.3 Management and Firmware

This section briefly describes the current development status of the RECS|Box and t.RECS server systems and then focuses on the achievements reached within VEDLIoT.

The RECS|Box and t.RECS server systems both include an integrated management system that allows the user to control the installed microservers and monitors them. Both systems share most of the management functionality. The firmware binary is the same for both systems. The firmware dynamically detects the hardware during boot and adapts accordingly.

The main functionality of the management system is as follows:

- Allow control of compute nodes:
    - On/Off, Reset
    - Basic configuration
    - Configuration of internal switching (VLANs)
    - Configuration of PCIe infrastructure (sharing PCIe functions)
    - Fan management
- Monitor sensors and other data:
    - Temperatures
    - Power consumption, Voltages

  - Power supplies
  - Fan speeds
  - Data provided by OS running on compute nodes (RECSDaemon)
- Allow access to nodes via Keyboard, Video, Mouse:
  - Via Front Panel (HDMI, USB) or via network (iKVM)
- Provide interfaces for monitoring and control:
  - WebGUI
  - REST-API
  - Redfish
  - IPMI
  - Grafana/Prometheus

To achieve these goals, each of the Baseboards and Backplanes (in the case of the RECS|Box) and the t.RECS include an embedded microprocessor that runs Linux and manages all the components on the respective board. One of these processors additionally runs the main management software, called RECS_Master. While in the RECS|Box this is runnning on the first backplane of the system, in the edge server it is running on the baseboard itself. The RECS_Master is communicating with the rest of the Firmware programs and is providing the management interfaces for the user. For regular manual administrative usage, a WebGUI [11] is provided that allows access to all user-changeable settings and all monitoring and control functionality. In addition to that, an IPMI interface is offered by the chassis controller. Furthermore, there is a self-defined RESTful API [12], which allows retrieval of all measured sensor values and basic control of the nodes. Additionally, a Redfish API [13] is available, including all monitoring and control features of the aforementioned RESTful API and also allows for controlling the PCIe infrastructure. The software architecture is further detailed in Chapter 2.3.2 when describing the changes necessary for the t.RECS.

Within VEDLIoT, a Prometheus [14] exporter, together with a Grafana dashboard, were added to the capabilities of the RECS_Master [15] to allow advanced monitoring with these popular tools. The big advantage of the Prometheus exporter compared to other APIs is that it dynamically exports its own metrics and thus, additional metrics can be added or removed during run-time after changing or hotplugging hardware. This allows to export only metrics of those microservers that are plugged in. As the RECS|Box has a modular approach and every RECS|Box can be equipped with different base boards and microserver configurations, this approach is of high relevance. Using traditional monitoring tools, that don't support the export of dynamic metrics, needs regular manual changes of the configuration, resulting in unnecessary additional work.

The Prometheus export API can be enabled and disabled in the Web Interface. Once it is enabled, all metrics relevant are available via https://<host>/metrics/ and can be added as an endpoint in Prometheus.

A (truncated) Prometheus export looks like this:

```
# HELP recsbox_fan_rpm Fan RPM
# TYPE recsbox_fan_rpm gauge
recsbox_fan_rpm{FanName="Back 1",} 4432.0
recsbox_fan_rpm{FanName="PSU 1",} 5376.0
recsbox_fan_rpm{FanName="Back 3",} 4411.0
recsbox_fan_rpm{FanName="Back 2",} 4496.0
# HELP recsbox_recsdaemon_sensors Diverse OS-level sensors from the RECS Daemon
# TYPE recsbox_recsdaemon_sensors gauge
# HELP recsbox_psu_status PSU Status
# TYPE recsbox_psu_status gauge
recsbox_psu_status{PSUName="PSU 1",} 1.0
# HELP recsbox_node_power_usage_watts Node Power Usage
# TYPE recsbox_node_power_usage_watts gauge
recsbox_node_power_usage_watts{NodeName="Node 2-3",PowerType="Total",} 18.79768073
recsbox_node_power_usage_watts{NodeName="Node 2-3",PowerType="PEG",} 0.0
recsbox_node_power_usage_watts{NodeName="Node 2-2",PowerType="PEG",} 0.0
recsbox_node_power_usage_watts{NodeName="Node 2-1",PowerType="PEG",} 0.0
recsbox_node_power_usage_watts{NodeName="Node 2-3",PowerType="Node",} 18.79768073
recsbox_node_power_usage_watts{NodeName="Node 2-1",PowerType="Total",} 16.2074491
recsbox_node_power_usage_watts{NodeName="Node 2-2",PowerType="Node",} 20.0557932
recsbox_node_power_usage_watts{NodeName="Node 2-1",PowerType="Node",} 16.2074491
recsbox_node_power_usage_watts{NodeName="Node 2-2",PowerType="Total",} 20.0557932
…
```

Prometheus needs very little configuration to automatically parse all information and write it into a database. This makes all metrics easily accessible. The 'scrape config' for the metrics looks like this:

```
  - job_name: 'RECS_Master'
    scrape_interval: 1s
    scrape_timeout: 1s
    static_configs:
     - targets: ['<host>']
    basic_auth:
      username: 'user'
      password: 'password'
```

It is recommended to use Grafana as a graphical dashboard to read out these captured metrics from a Prometheus server. A pre-build Grafana dashboard is publicly available at the Grafana website [16]. It can be integrated in Grafana using the "Import" function. The dashboard automatically reads the available metrics from Prometheus and dynamically adapts to the number of available microservers. It is divided in the two parts "Nodes" and "Chassis" to structure the monitoring data, see Figure 12.

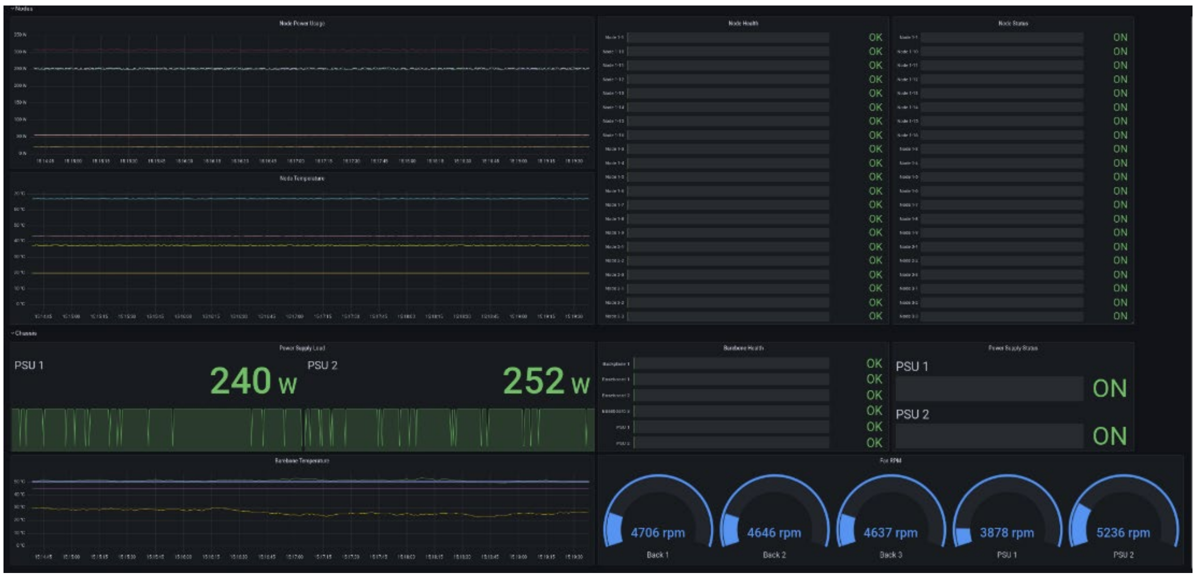
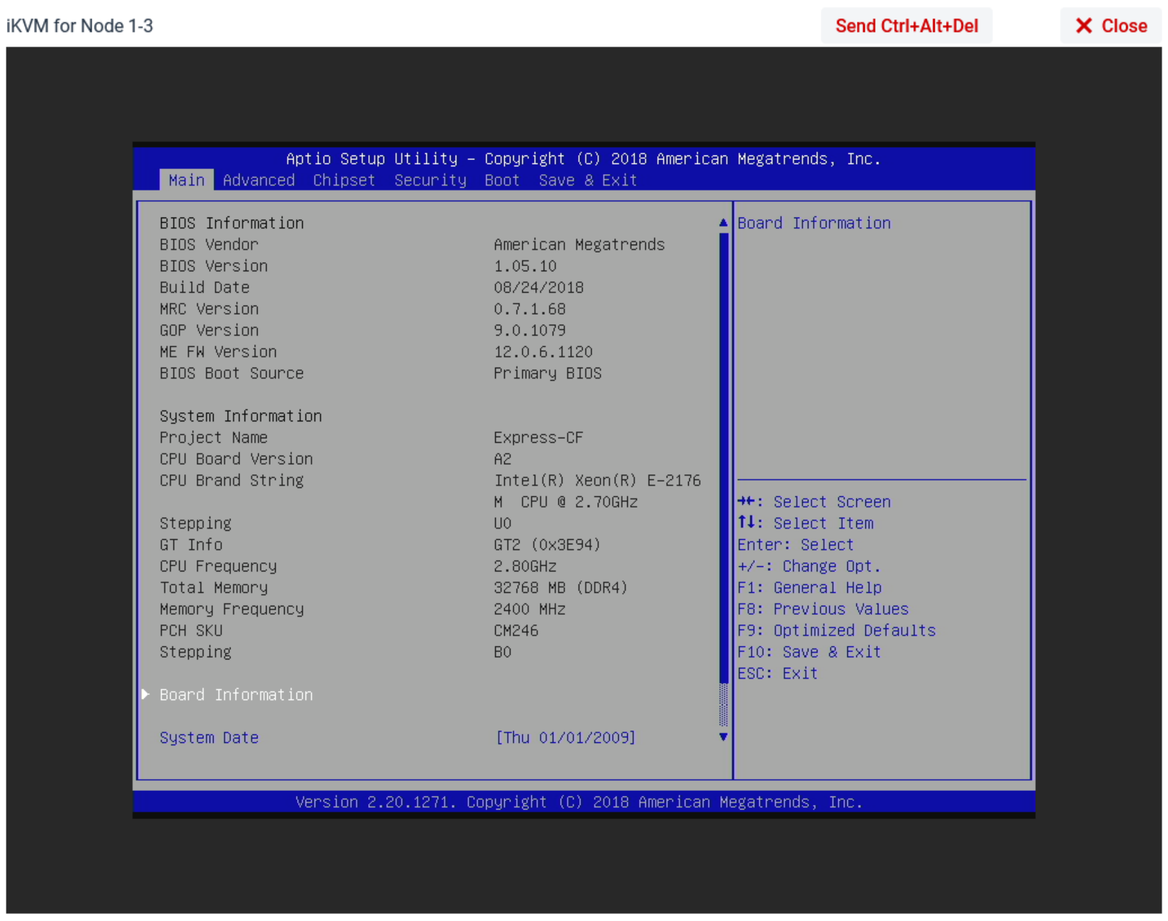


Figure 12 - Grafana Dashboard for RECS|Box and t.RECS

The REST API, which was originally developed in previous projects [17] [18], was enhanced within VEDLIoT. It now contains even more detailed monitoring data. A comprehensive documentation can be found at [12].

The RECS|Box and t.RECS systems include a switched KVM[1] infrastructure to allow direct access to each of the installed microservers. It is either possible to connect physical monitor, mouse and keyboard or to access the Microservers via iKVM (KVM over Ethernet). The iKVM solution is based on the VNC protocol, as this is a very popular protocol for remote access and many clients are available.

Within VEDLIoT, an HTML 5 noVNC-Client was integrated in the WebGUI to further improve the remote management capabilities of the microservers within the RECS|Box and t.RECS platforms via iKVM. With this client, it is possible to access one microserver of a baseboard (or the t.RECS, respectively) at a time. The microserver to be accessed via iKVM can be selected in the WebGUI or by using the REST or Redfish APIs. The noVNC client in the WebGUI can then be opened and shows the current video output stream of the selected microserver and passes the keyboard and mouse control signals back to it. A detailed description of the iKVM subsystem can be found in [18].



```
Aptio Setup Utility – Copyright (C) 2018 American Megatrends, Inc.

BIOS Information                                    Board Information
BIOS Vendor              American Megatrends
BIOS Version             1.05.10
Build Date               08/24/2018
MRC Version              0.7.1.68
GOP Version              9.0.1079
ME FW Version            12.0.6.1120
BIOS Boot Source         Primary BIOS

System Information
Project Name             Express-CF
CPU Board Version        A2
CPU Brand String         Intel(R) Xeon(R) E-2176
                         M  CPU @ 2.70GHz             →←: Select Screen
Stepping                 U0                           ↑↓: Select Item
GT Info                  GT2 (0x3E94)                 Enter: Select
CPU Frequency            2.80GHz                      +/-: Change Opt.
Total Memory             32768 MB (DDR4)              F1: General Help
Memory Frequency         2400 MHz                     F8: Previous Values
PCH SKU                  CM246                        F9: Optimized Defaults
Stepping                 B0                           F10: Save & Exit
                                                      ESC: Exit
▶ Board Information

System Date              [Thu 01/01/2009]

Version 2.20.1271. Copyright (C) 2018 American Megatrends, Inc.
```

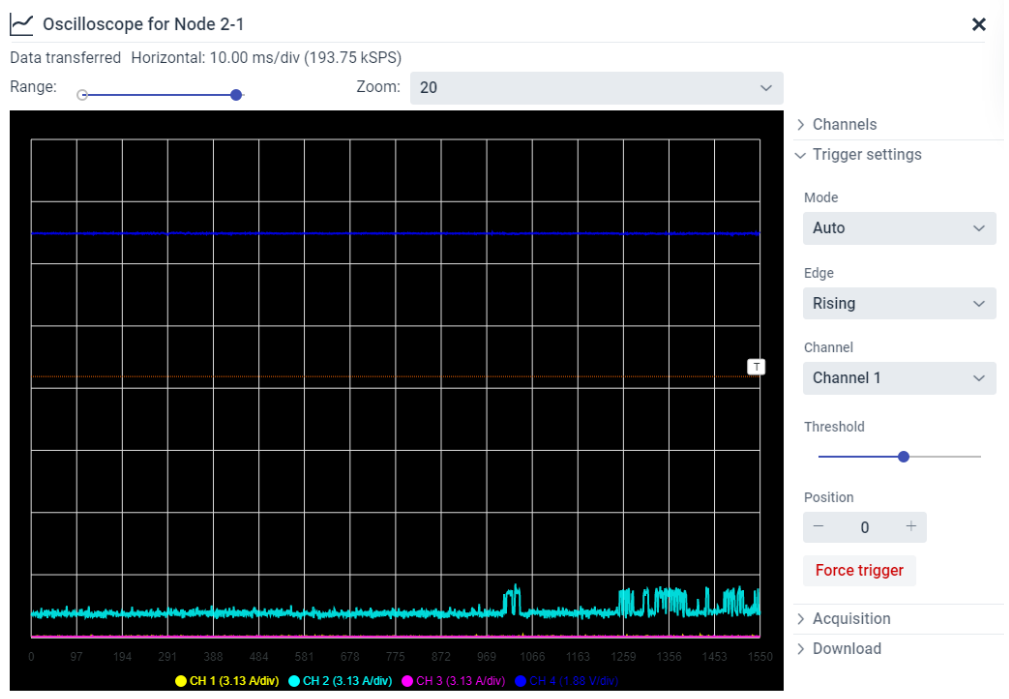
Figure 13 - NoVNC-Client integration in the WebGUI

In addition to the KVM and iKVM capabilities, the RECS_Master features access to the serial console of each microserver. An embedded serial console client is integrated in WebGUI. Besides the standard input and support for copy and paste actions, it allows resizing the console window and selection of different code pages. Alternatively, it is also possible to start an embedded telnet server and connect to the selected microserver via an external client like e.g. PuTTY [19].

[1] Keyboard, Video and Mouse

The monitoring system of the RECS|Box and t.RECS servers usually works with a sensor update rate in the order of 1 Hz. While this is certainly fast enough for monitoring the health status of the system, it does not allow characterizing the energy consumption of computational kernels running on the microservers. To be able to do this a much higher sampling frequency for certain signals in a given timeframe is realized via a 'Oscilloscope mode'. When this mode is enabled by the user, the baseboard hosting of the selected microserver samples the respective ADC channels with the maximum available sample rate. Instead of just computing average, minimum and maximum values of the data, it is completely stored in a ring buffer where it can later be retrieved for analysis. The sample rate available to the user is hereby increased to up to 1 Mega Samples per Second (MSPS). Within VEDLIoT, a comprehensive control panel for this 'Oscilloscope mode' was integrated to the WebGUI to ease the fine-grained monitoring/characterization of the RECS|Box and t.RECS servers, e.g., for application profiling.

The control panel shown in Figure 14 can be opened by clicking a button on the management page of a node in the system. The dialogue then allows setting up the trigger mode to either automatically sample one buffer after the other, trigger each time a certain condition is met or trigger once that condition is met. A channel to be checked for the trigger condition can be selected. This channel is then checked for either rising above a given threshold, falling below it or both. The trigger level and position relative to the sample buffer can also be set. As the hardware has no special circuit for triggering, this is a software trigger only. But for the intended purpose of analyzing computational kernels, the trigger delay is small enough to show the desired results. Currently, the sampling rate is fixed, but a zoom- and a range control allow displaying parts of the current sample buffer in more detail. Finally, the raw sample buffer data can be downloaded as a simple CSV file for further analysis in other programs. Future planned improvements are selection of channels to be sampled, thus increasing the sampling rate (as the maximum sampling rate is shared between all active channels). Also the size of the sample buffer should be adjustable, and the scaling of the Y-axis could be improved, as the size of each division is currently based on the raw ADC value, leading to odd values like "3.13 A/div" instead of the usual 1/2/5 scaling known from regular oscilloscopes.


Oscilloscope for Node 2-1

Data transferred Horizontal: 10.00 ms/div (193.75 kSPS)

Range: Zoom: 20

Trigger settings

Mode: Auto

Edge: Rising

Channel: Channel 1

Threshold

Position: 0

Figure 14 - Oscilloscope control panel integration in the WebGUI

In addition to the main features described above, the RECS_Master was further enhanced within VEDLIoT. A fan control mechanism was implemented to manage the fan speed automatically based on the system temperature. For a better user experience, a persistent node configuration was implemented to store data per microserver like its boot mode, power state, KVM mode and serial console code page. Also, the power supplies now support an auto-on configuration to boot the system automatically. Furthermore, the WebGUI now features a notification system to keep the user updated on health events, errors or other information.

The improvement of robustness and reliability of the RECS|Box and t.RECS software systems was also a major focus of the software development. For this, a code-review of the whole RECS_Master management software was done. Several bugs as well as other programming flaws and potential security risks were eliminated. Simultaneously, the code-base was refactored, now utilizing the latest Java and third-party library features with the focus to enhance maintainability as well as performance.

An additional security control layer was implemented in the RECS_Master to enhance security and unify permission control for all access methods to the management software (WebGUI, REST API, Redfish API, IPMI). Also, by redesigning the communication between the RECS_Master and the Ethernet switch management software, the robustness of the system was enhanced.

Furthermore, the automated testing was heavily improved by implementing numerous additional tests. Advanced integration tests were developed to cover the whole RECS_Master software stack from the low level firmware interfaces up to the user APIs and the client-side web browser Javascript logic in the continuous integration testing pipeline.

2.3.1 RECS |Box and LP Baseboard NX

Within the VEDLIoT project, the firmware and management software was prepared to support baseboards with the NVIDIA Jetson Xavier NX microservers. Similar to already existing baseboard types (COM-Express, NVIDIA Jetson Tegra X2), monitoring and control of the attached microservers is possible. This includes power control, configuring the Ethernet and PCIe communication infrastructure as well as advanced monitoring of power and temperature sensors. The goal of this effort was the seamless integration of the LP Baseboard NX with its microservers into the RECS|Box management software, also supporting all available interfaces.

From a firmware standpoint, the LP Baseboard NX and the older LP Baseboard are extremely similar, allowing reuse of most of the existing code. As the board has a different combination set at the ID GPIO pins, the common bootloader can detect the new type of board and load an appropriate Device Tree file, which is mainly a copy of the LP baseboards' file. It also sets the "board_type" parameter in the Linux kernel command line arguments appropriately, allowing software starting later in the boot process to detect that it's running on the new LP Baseboard NX. Most of the components (e.g. the 10 G Switch manager and the PCIe switching manager) simply use the same configuration as the LP Baseboard. The main firmware program distinguishes between the two baseboard types to be able to report the correct node types to the RECS_Master, but as this software is written in C++, this was easily possible to inherit from the existing LP baseboard class. Node control (e.g. Power sequencing) is the same for both the Jetson TX2 and the Xavier NX. From the software perspective, the main difference is the 5V power supply of the Xavier NX microservers that lead to a small change in the calculation of the module power usage.

2.3.2 t.RECS

The t.RECS uses the same components for the management CPU, Ethernet switch and PCIe switch as the RECS|Box does. Thus, it makes sense to re-use the same software components as well. However, as now all functions are integrated on one board instead of split between Baseboards and Backplanes as in the RECS|Box, some changes where necessary. From a software standpoint, the t.RECS is implemented as a RECS|Box baseboard. Changes include added support for controlling and monitoring fans (which in the RECS|Box are connected to the backplane), a different power management microcontroller (which in the RECS|Box is located on a separate Power Distribution Board) and some new driver classes for hardware that have changed for the newer hardware (e.g. the ADC for measuring currents). As the firmware program is a modular C++ software, existing classes (e.g. for fan control) could simply be re-used to add the new features. Another difference of the t.RECS compared with the RECS|Box is that the RECS_Master is running on a Baseboard instead of a Backplane (as there is none). To achieve this, the t.RECS includes an M.2 slot that hosts a SATA SSD. The RECS_Master and its Linux Filesystem are stored on this drive.

Figure 15 shows the components of the monitoring system and their interaction. The RECS_Master and the firmware program, which is a regular Linux executable, share the monitoring and control functionality between them. The firmware program is handling low-level hardware access like setting pins, controlling the compute modules, reading sensors, etc. The RECS_Master is talking to the firmware program via Apache Thrift, querying its data and giving instructions for changing the system state. The RECS_Master then aggregates data and provides all the user-facing interfaces like the WebGUI and APIs. The RECS_Master also directly monitors the power supplies and handles the user interface on an optional small LCD screen for direct control on the chassis.

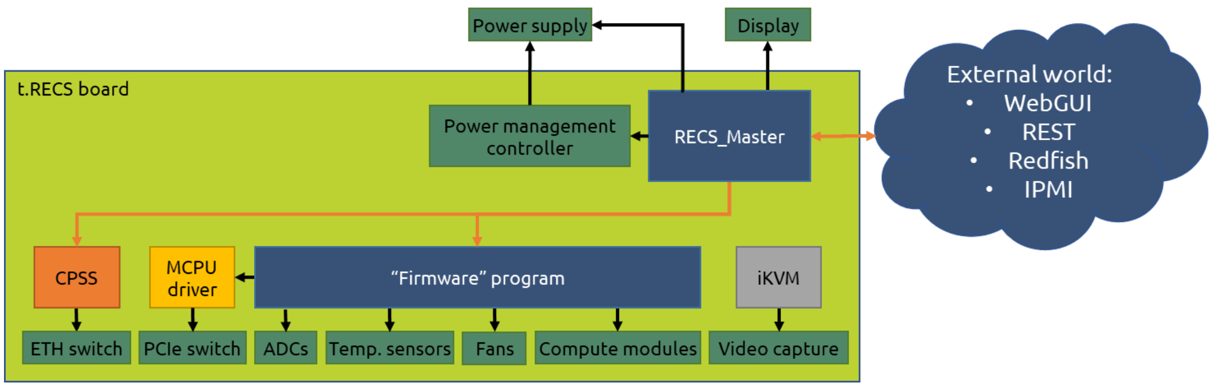


Figure 15 - Architecture of the management system

"CPSS" is the vendor-specific management software for the Ethernet switch used in the t.RECS and RECS|Box. It configures and manages the switches via PCIe and contains an external interface that is used by the RECS_Master to monitor and control the switches, e.g. setting up VLANs for the different ports. CPSS was also changed to initialize the PHYs on the t.RECS as necessary.

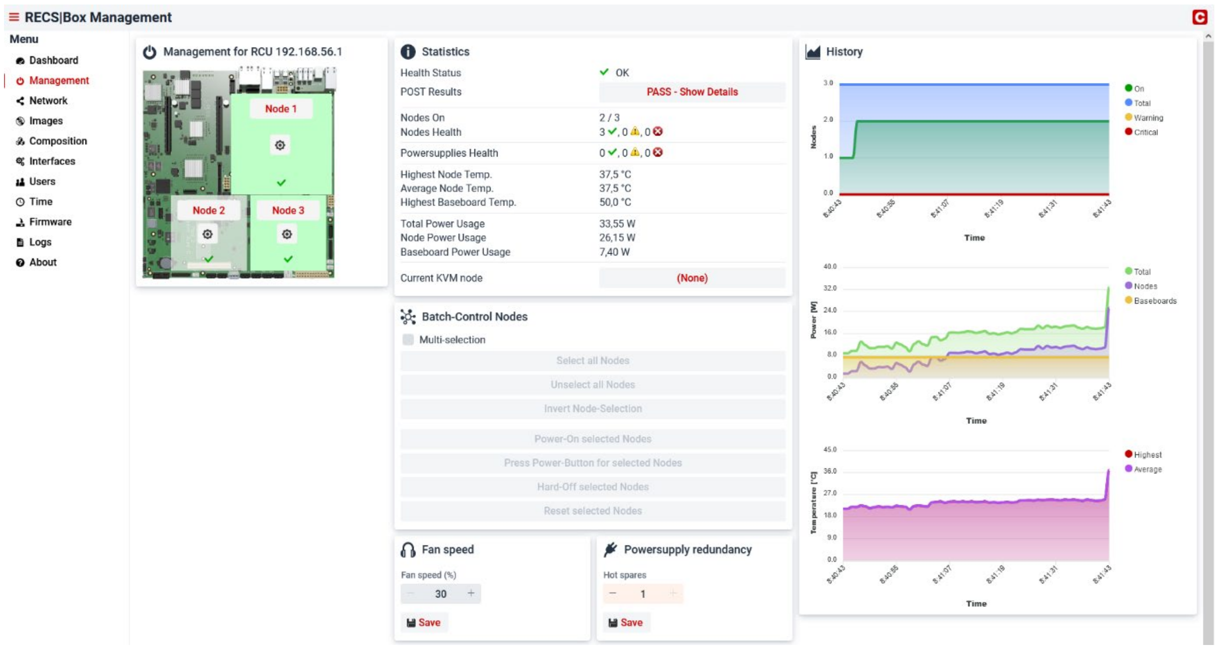
The "MCPU driver" is necessary to configure the advanced functionality of the PCIe switch. The firmware program can interact with this driver via SysFS files, e.g. for connecting Single Root – I/O Virtualization (SR-IOV) functions to the compute modules as instructed by the RECS_Master.

The t.RECS also includes iKVM functionality, allowing access to the Keyboard, Video and Mouse of each compute module via Ethernet. The protocol used for this in the RECS|Box and t.RECS is known as "VNC" (technically RFB). As the open-source implementation of this protocol is licensed under GPL, the iKVM software was written as a separate program to avoid licensing issues. It captures video images from the selected compute module, processes them and makes them available via the RFB protocol. In the opposite direction it accepts mouse and keyboard input via RFB and forwards it to a virtual USB mouse and USB keyboard, that are provided by the management CPU's USB gadget functionality.

During the project, the support of the t.RECS server platform within the RECS_Master management software was also further improved. As the t.RECS has no integrated power supplies supporting Power Management Bus (PMBus), an enhanced configuration mechanism was implemented to also support ATX power supplies.

On the frontend side, the development goal was that the t.RECS can be controlled and monitored in the same manner like the RECS|Box (see screenshot in Figure 16). This includes all features listed at the beginning of this section. This has been achieved by adapting the RECS_Master backend and the respective views in the WebGUI to match the differing requirements.

As an additional feature of the t.RECS WebGUI, the node composition wizard was enhanced to feature templates for handling common PCIe communication scenarios (e.g. connecting a PCIe device to a node exclusively).



Statistics

Health Status	✓ OK
POST Results	PASS - Show Details
Nodes On	2 / 3
Nodes Health	3 ✓, 0 ⚠, 0 ✖
Powersupplies Health	0 ✓, 0 ⚠, 0 ✖
Highest Node Temp.	37,5 °C
Average Node Temp.	37,5 °C
Highest Baseboard Temp.	50,0 °C
Total Power Usage	33,55 W
Node Power Usage	26,15 W
Baseboard Power Usage	7,40 W
Current KVM node	(None)

Fan speed

Fan speed (%): 30

Powersupply redundancy

Hot spares: 1

Figure 16 – Screenshot of t.RECS control view in the RECS_Master Web UI

All interface capabilities of the RECS_Master management software have been adapted to support the t.RECS, allowing for easy integration into external software environments and applications.

When more different COM-HPC modules became available in the project, some had issues turning on in the t.RECS. Debugging this led to hardware- and configuration fixes as well as improved power sequencing in the firmware so that all tested modules now start in the t.RECS. Other configuration changes fixed problems with the USB-C ports and improved link up reliability between the modules and the PCIe switch.

The ADC originally used in the t.RECS was not fast enough for the “Oscilloscope mode” to be really useful and thus was replaced with a different one using a PCB that can be soldered to existing connections on the t.RECS PCB. The software was updated to now use this new ADC, which is the same as previously used in the RECS|Box. The accuracy of the power measurements was increased by implementing calibration functions that allow correcting for offsets and linearity of the sense circuitry. These settings are then stored on the nonvolatile memory and loaded on booting the system.

3 Detailed u.RECS architecture

After the VEDLIoT use-cases were analyzed in WP2, the requirements for the new hardware platform were defined [2]. It was determined that the current RECS family is not suitable for the VEDLIoT use-case requirements. The smallest form factor available, the t.RECS, is not only mechanically too large but also consumes too much power for far-edge use cases. Therefore, the family needed to be extended. This has been achieved through the development of the u.RECS-baseboard. Figure 2 shows the current RECS family, including the newly developed u.RECS.

It was necessary that the u.RECS retained some of the RECS family features. Furthermore, it needed to be flexible, modular, adaptable and easily expandable. Like the RECS family so far, the u.RECS uses a module-based approach. This makes it possible to directly support various ML accelerators and Computer-on-Module (COM). Additionally, it is possible to combine these modules and extend them with PCIe-based accelerators.

The overview of the u.RECS is presented in Figure 17. The two integrated module slots support the SMARC 2.1 standard and the NVIDIA Jetson NX form factor. An overview of the ML accelerators and COMs that can be installed in these slots is given in the following Sections. In addition to the two module slots, an M.2 slot and an mPCIe slot are integrated, which can be used to add further accelerators or communication methods, such as 5G, to the u.RECS. Furthermore, communication options, such as Ethernet or PCIe, and energy measurement methods are integrated on the board, to make the u.RECS a perfect fit for the VEDLIoT use-cases. In addition to the overview, Figure 18 shows a detailed block diagram of the u.RECS architecture.

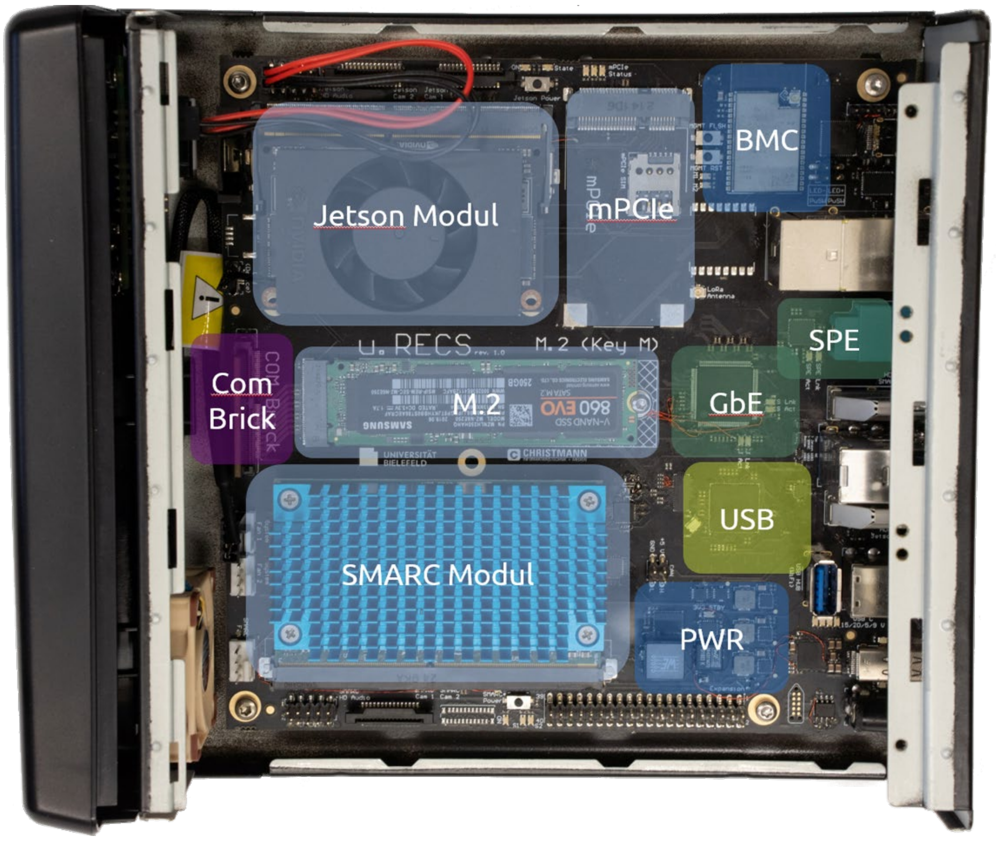


Figure 17 Overview of u.RECS system

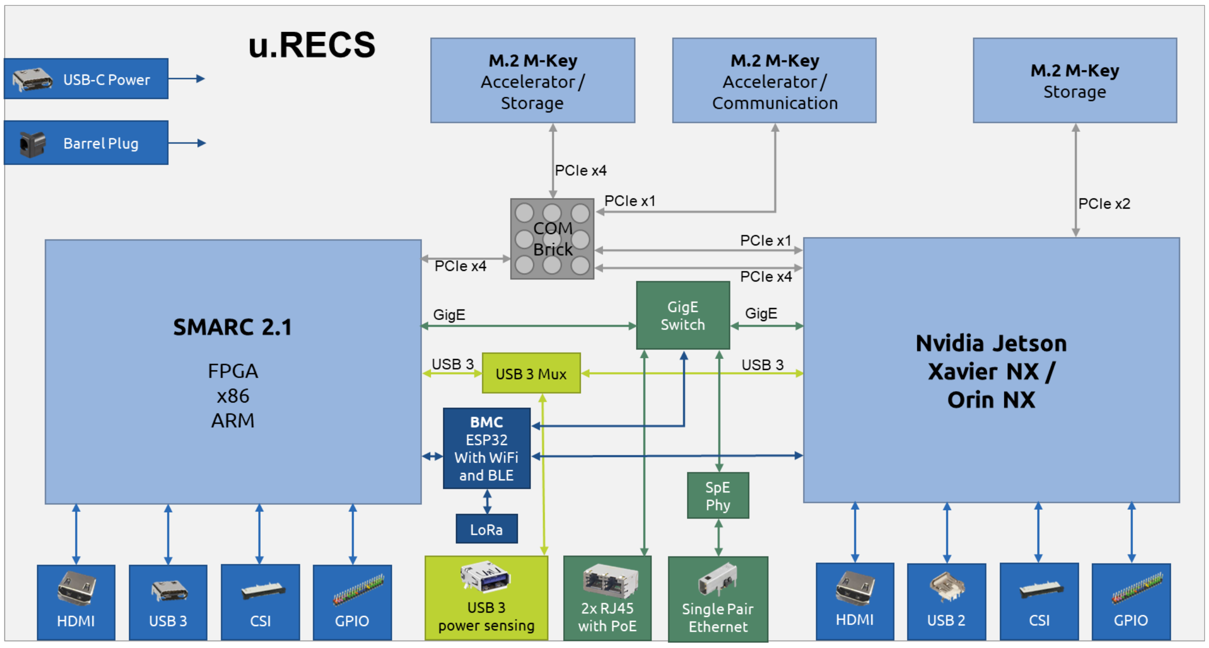


Figure 18: u.RECS Architecture

To manage the different modules and functions on the u.RECS, a board management controller (BMC) is used. An ESP32 was chosen, which is characterized by its low cost, low power consumption and good software support. It also provides its own WLAN and Bluetooth interface, in addition a LoRaWan radio module is connected to the ESP. It is, therefore, possible to access the management interface of the u.RECS via a variety of interfaces. The BMC is able to change the power states of the different modules and parts of the communication infrastructure on the u.RECS.

It is still possible to access the management interface through its own WLAN interface or a connected LoRaWan radio module even if the Ethernet switch is powered off. Another key feature of the BMC is to measure the energy consumption of the modules, this is made possible by an ADC connected via SPI and corresponding current measurement modules.

In the following Sections, the specific communication possibilities are described and the management and monitoring functionalities are introduced. Furthermore, an overview of the power supply and the power budget is given, and finally, an outlook on the mechanical integration and pictures of the system are presented.

3.1 Supported Microservers

Support for the SMARC 2.1 standard gives the u.RECS access to a wide range of supported COMs and ML accelerators, as SMARC-Modules are available on the market through different module manufacturers, such as Congatec or AD-Link. The SMARC slot can be equipped with, among others, the following types of system (cf. Figure 19):

- ARM CPU (e.g., i.MX 8)
- x86 CPU (e.g., Atom CPU)
- FPGA (e.g., Xilinx Zynq UltraScale+)
- ML accelerator (e.g., Vedliot Versal SMARC module)

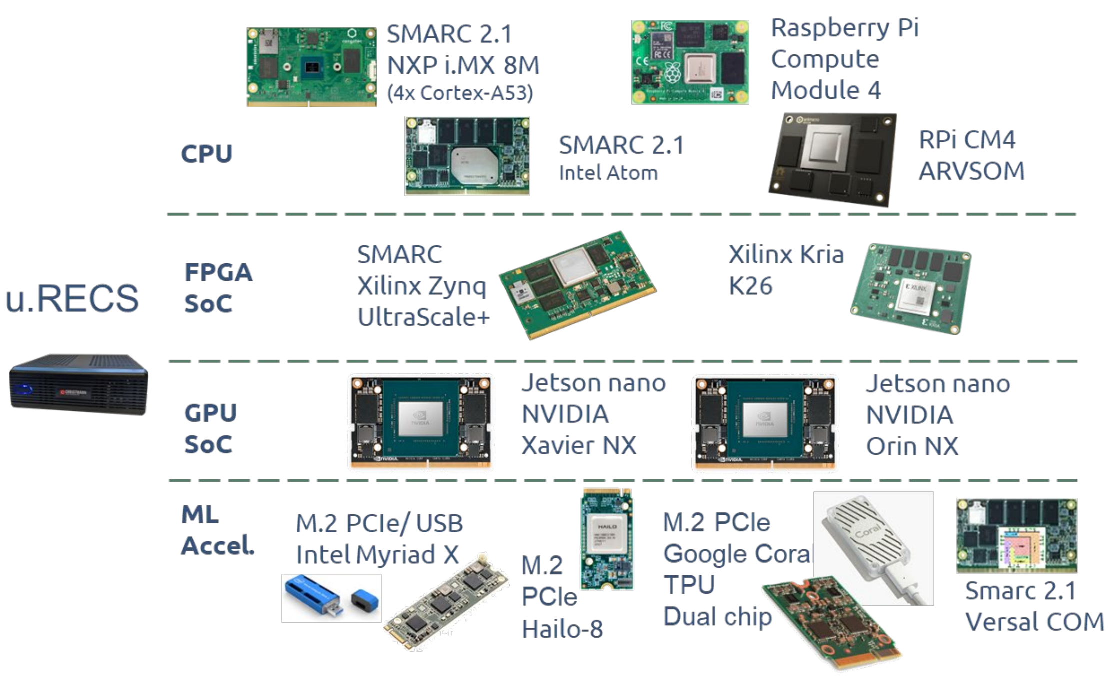


Figure 19 Overview of the supported Microservers for the u.RECS

Out of the box, the Jetson NX slot only supports NVIDIA Jetson NX and Nano modules. The Orin NX has the same form factor as the Jetson NX but additionally needs higher supply voltage and dedicated NVMe storage as also discussed in section 2.1. u.RECS was extended with variable supply voltage and separate M.2 NVMe connector enabling Orin NX support. The NVIDIA modules are all based on ARM systems and integrate a tightly coupled NVIDIA graphics accelerator. The Xavier NX combines its Volta-based GPU with a six-core Carmel ARM®v8.2 CPU. In the new Orin NX generation, this is upgraded to an Ampere GPU in combination with an eight core Arm® Cortex®-A78AE.

To increase the usability of the Jetson NX slot, an adapter to the Raspberry PI CM4 standard is foreseen. This would allow the installation of, e.g., a low-cost RPi CM4 module or the ARVSOM board from Antmicro. Furthermore, considering the popularity of the Raspberry PI, it can be assumed that this standard will become increasingly popular and more modules supporting it will be introduced to the market.

3.2 Expansion Interfaces

There are a number of additional ML accelerators that can be equipped in or connected to an M.2 or mPCIE slot (cf. Figure 19). Additionally, it is possible to connect accelerators via USB 3.0 and access them from one of the compute modules. Furthermore, with the u.RECS it is possible to measure the energy of an accelerator connected via USB. Accelerators supported this way include:

- Intel Myriad X
- Hailo-8
- Google Coral Edge TPU

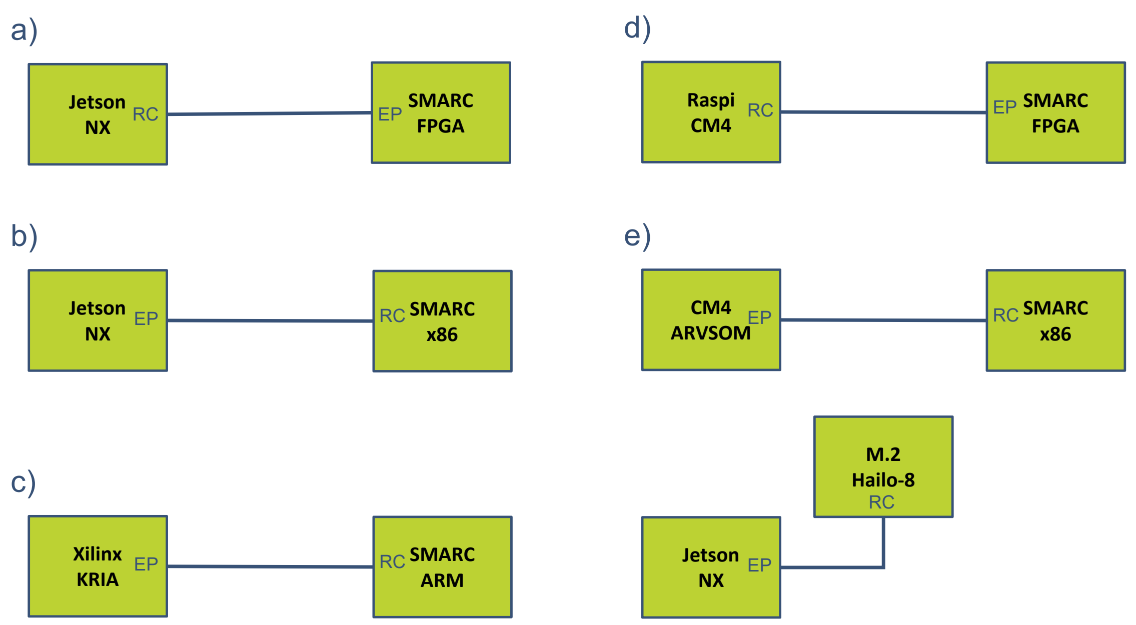


Figure 20 Possible PCIe Connections between Microservers

As shown in Figure 18, the SMARC and Jetson modules can communicate among each other and with the outside via Ethernet. They also feature a switchable USB port with energy measurement capability, to which, e.g., an external accelerator can be connected. PCIe is another option for communication on the u.RECS, it connects the two modules and the M.2 slot. To allow different configurations without increasing the power consumption, a passive PCIe connector is used instead of the previously presented implementation in Deliverable D2.2 of a PCIe mux. This design is described further in the following subchapter. Depending on the configuration, the module on the Jetson, SMARC or M.2 is, therefore, a PCIe root complex (PCIe host) or PCIe endpoint (PCIe device). An overview of the different possible configurations is given in Figure 20.

3.3 Communication

As mentioned in previous chapters, one form of communication between modules is Gigabit Ethernet. In contrast to the larger members of the RECS family, GbE is the fastest ethernet-based interface for communication on the u.RECS, due to low cost and high efficiency.

The Ethernet switch has seven ports, it connects the modules and the RJ45 ports via copper standard 1G_BaseT and the BMC (ESP32) is interfaced via single-ended ethernet standard RGMII. Additionally, a single pair Ethernet (SPE) connection is available for the automotive use case, as well as the ability to connect two Gigabit Ethernet cameras with Power over Ethernet (PoE). To achieve this, an additional SPE-Phy is connected to the SGMII port of the switch, and the power supply for PoE on the RJ45 ports was implemented.

Furthermore, the IEEE1588 [20] standard should be supported. This enabled the use of the Precision Time Protocol (PTP) and, thereby, the realization of time-synchronous Ethernet. As a result of these requirements, the choice of chips was very limited, and in addition, due to the chip crisis during Covid-19 pandemic, chips of this type have been rarely available on the market. This significantly delayed the manufacturing of the u.RECS within VEDLIoT.

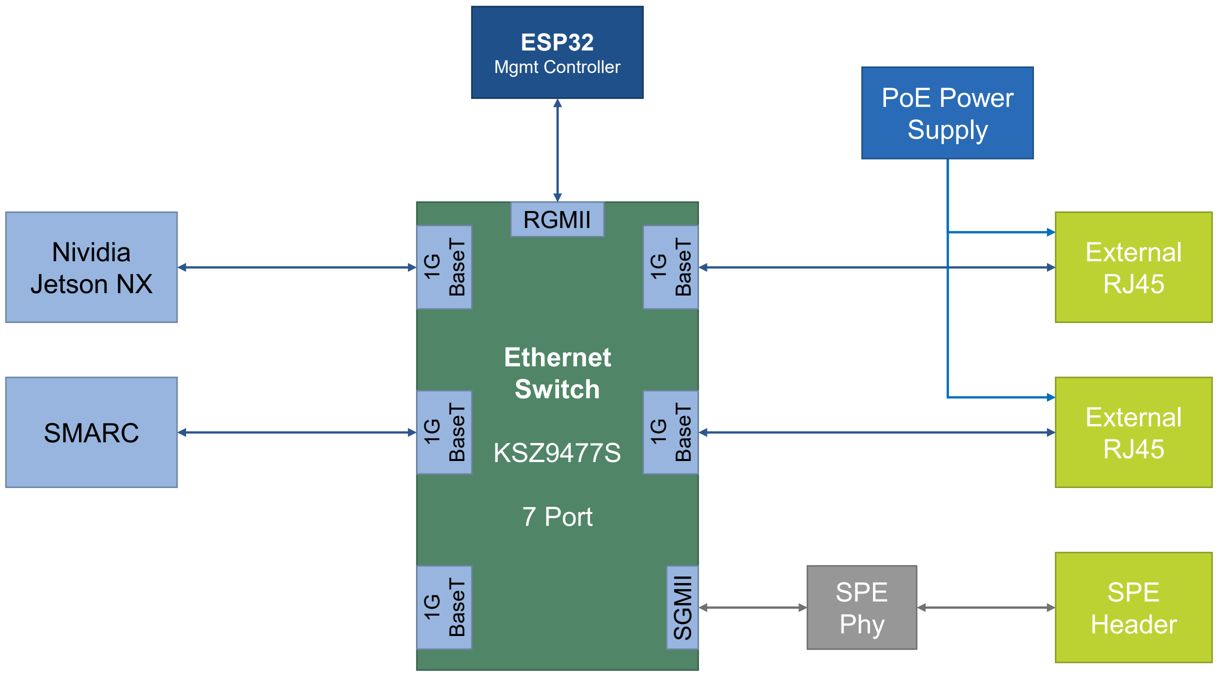


Figure 21 Ethernet Connections on the u.RECS

In the larger version of the RECS platform, PCIe-based connection of the modules was realized using a multiplexer. However, this requires a number of additional ICs, which increases the power consumption, and is a problem for far-edge platforms like the u.RECS. Therefore, the usefulness of the mux-based design was re-evaluated. One of the main advantages of the mux-based solution is that the PCIe topology can be changed electrically at run-time without physical user intervention. However, this is not an advantage for the u.RECS, since changing the configuration of the PCIe connections would only be done in combination with a module swap. Therefore, a passive approach is taken for switching the PCIe connections. To facilitate the manual reconfiguration of the PCIe topology, the PCIe lanes of the different modules are consolidated on a connector. With the so-called "PCIe Communication Brick (COM-Brick)" it is then possible to connect the modules as required (see Figure 22). It is possible to cover all necessary connections with three different COM-Bricks (COM1, COM2 and COM3), illustrated in the figure similar to click bricks. The user, therefore, not only has to change the module but also select the appropriate COM-Brick. However, no additional energy is consumed, as this solution is passively implemented.

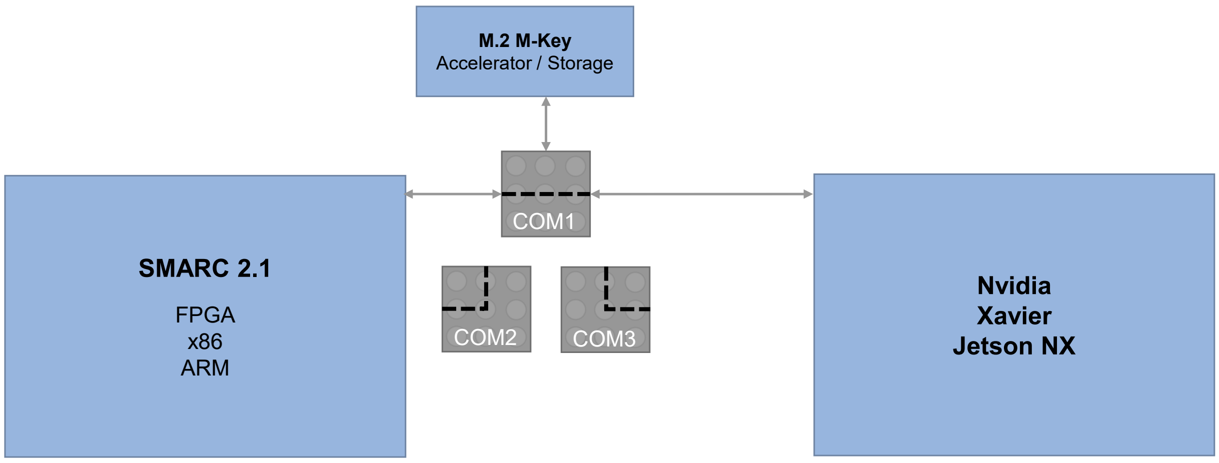


Figure 22 Layout of the PCIe Communication Brick

The SMARC and Jetson module provide additional interfaces (cf. Figure 18). Both modules are equipped with a dedicated USB and HDMI port to simplify initial system setup and increase overall ease of use. In addition, serial camera interfaces (CSI) are provided for each module, allowing one or more cameras to be connected directly to the modules. Beyond that, there is a GPIO connector that shares the GPIO pinout of the Raspberry Pi Modules. This connector has SPI, I2C, CAN and UART interfaces, among others. There are also plans to implement a CAN header for the Jetson NX that adapts to a sub-D9 connector which is used in automotive evaluation systems. This and 5G communication are requirements arising from the automotive use case.

In the u.RECS, 5G communication is provided by a mPCIe expansion card. Such cards are available, e.g., from the manufacturer QUECTEL. This allows for the NVIDIA Jetson NX to serve as a gateway for the other devices connected via Ethernet.

3.4 Management & Monitoring

The u.RECS can be powered in two different ways: via a barrel connector or a USB Type-C connector. One of the most common ways in the industry to power a Mini-ITX board is the barrel connector. Therefore, this is also supported in the u.RECS. In addition, the widely used method of powering laptops and cell phones via USB-C is also supported. This allows for the u.RECS to be powered with any standard USB-C laptop power adapter, or using an USB-C power bank with sufficient power. The USB-C port of the u.RECS supports the USB power delivery (USB PD) standard [21].

Table 2 u.RECS power budget overview

Power Budget		
54 V	PoE Power (560 mA 30 W)	
20 V – 9V	Jetson Orin NX (4A 30 W)	
5 V	Jetson (4 A 20 W)	SMARC (5 A 25 W)
3.3 V	M.2 (3 A 10 W)	mPCIe (3 A 10 W)
2.5V	KSZ9477S (0.3 A)	
1.5V	mPCIe (0.5 A)	
1.2V	KSZ9477S (1.3 A)	
1.0V	Dp83tg720s (0.2 A)	

The USB PD standard specifies different power levels that can be supplied. When analyzing the power budget of the u.RECS, it was determined that the 60 W mode satisfies the requirements. The USB-C PD controller negotiates a voltage of 20 V with the power supply, from which a maximum current of 3 A can be drawn. From this supplied 20 V, other voltages are generated, including 5 V and 3.3 V for the modules as well as other voltages for the communication infrastructure. As discussed in section 3.1 a variable power supply for the Jetson NX slot was implemented, providing higher voltages to the Orin NX module enabling high performance modes of the module. Table 2 provides detailed information on the individual voltages and the required power. The voltages that are required for the Switch and Phy are sequenced in hardware, this is done to use less pins on the BMC.

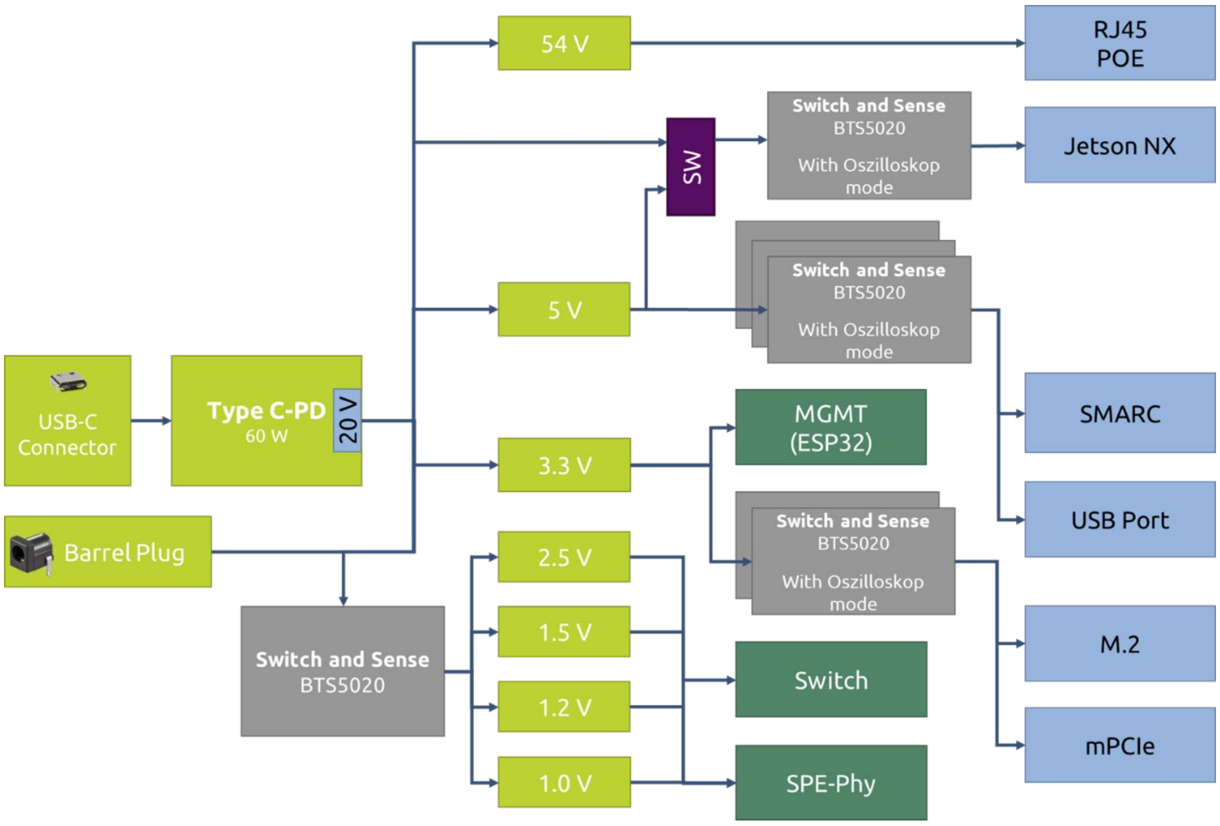


Figure 23 u.RECS power supply overview

To evaluate the energy efficiency in the use cases, the various components on the u.RECS are measured in terms of energy. Due to hardware limitations, these measurements are only performed once every second. The different consumption measurements can be accessed via the BMC using the web GUI. Figure 23 gives an overview of the voltage levels. The gray boxes are points where the voltage can be measured and turned on or off if required. The energy for the following components can be measured:

- t.RECS total power
- Jetson
- SMARC
- One USB port
- M.2 Slot
- mPCIe Slot
- Switch and SPE-Phy

Additionally, an oscilloscope mode for the modules is foreseen. This mode allows for the energy measurement to be performed at a higher rate than the usual one second cycle. Although this can only be done for one module at a time, it is very useful to analyze the energy efficiency of an algorithm more precisely, by evaluating the different parts of an algorithm individually. To implement this, the firmware will use the DMA controller of the ESP32 to directly write ADC samples to a memory buffer.

Considerably effort is being made to measure the energy consumption of the modules and the communication infrastructure. The idea is that these measurements could provide a more holistic view of a distributed ML problem in terms of energy consumption. Figure 24 gives an example of how such an analysis might look like. The diagram on the left side shows the distribution of energy consumption of an example system with different topologies. In the diagram next to it the performance in terms of frames per second (FPS) is plotted. Using such an analysis of a distributed ML system it is possible to quickly identify the most efficient topology.

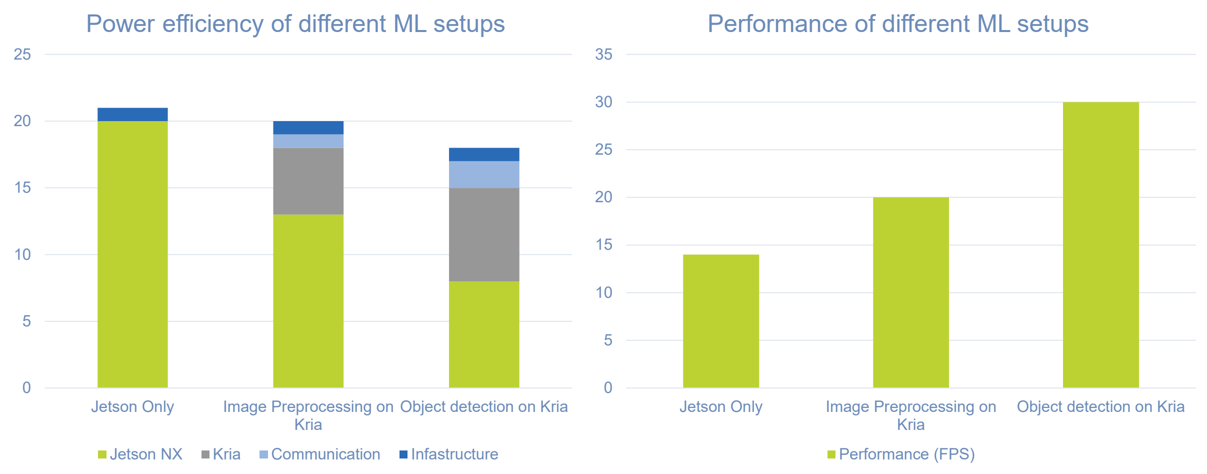


Figure 24 Envisaged diagrams for comparing the energy efficiency of modules

The firmware can connect to the publicly available "The Things Network" (TTN) via the LoRaWAN module on the u.RECS. When configured it will send out a status message containing e.g. Jetson/SMARC module status, power usage, temperatures and fan speeds in regular intervals. The status message is in binary form using as little bytes as possible. Therefore, a so called "Payload formatter" for TTN was written that converts the message back into separate fields for further processing.

3.5 Mechanical Integration

The modules that the u.RECS accommodates have a direct influence on the form factor. A test arrangement of the modules showed that the Mini-ITX [22] standard would be a suitable candidate for the u.RECS. This is a widely used form factor with numerous different cases available on the market. This allows for different chassis to be purchased, at a reasonable price without the need for expensive custom designs.

Therefore, the Mini-ITX standard was chosen as the form factor for the u.RECS. Additionally, with its 170 by 170 mm footprint, the Mini-ITX standard is just large enough to accommodate all the components of the u.RECS, without wasting any space while remaining small enough for edge application. Figure 25 shows the DIN-rail mountable Mini-ITX case, that accommodates the u.RECS.

Figure 25 Mini-ITX case that can be mounted on a DIN-rail [23]

The u.RECS board was, therefore, developed according to the Mini-ITX standard. Figure 26 shows the board outline with the matching mounting holes. On one edge of the board is the area for the slot bracket, here a number of connectors are placed to be able to use the crucial interfaces when the chassis is closed. It should also be noted that there is only 3 mm of space under the PCB towards the chassis. Therefore, no components should be placed on the bottom side that violates this restriction.

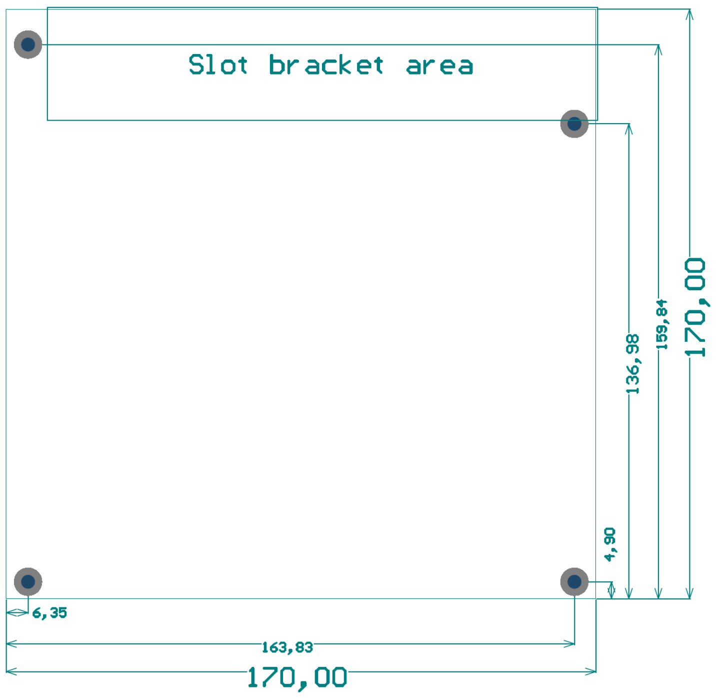


Figure 26 Board outline of the u.RECS

Figure 27 shows front and back view of the actual u.RECS inside the Mini-ITX case. It is visible that the connectors for both modules have been placed on the board and the required interfaces have been positioned in the area of the slot brackets. Some interfaces are only accessible when the u.RECS is not used in a Mini-ITX case (see also Figure 17). This is a deliberate design decision, as it is likely that in situations where these interfaces are needed, a user-specific chassis is required.

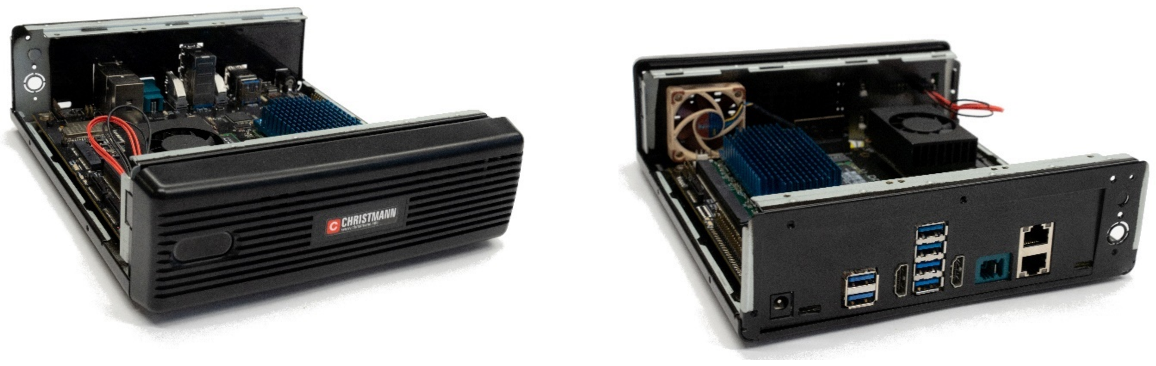


Figure 27 Front and back view of the u.RECS inside Mini-ITX case

3.6 Firmware architecture

As described in the previous subchapters, the management microcontroller is an ESP32. It is well-known in the maker community due to its low price point, its integrated WiFi, Bluetooth Low Energy and Ethernet support [24]. Therefore, it has a broad software support with lots of different libraries that help us in the software development.

The main tasks and implemented features of the management microcontroller are:

- Power management for the SoC modules
- Ethernet Switch configuration and monitoring
- WiFi configuration
- Power, voltage, temperature and health monitoring for all relevant devices
- PoE status monitoring
- Chassis fan management
- Provide Web GUI for management and monitoring
- Provide REST API for management and monitoring
- Provide LoRa interface and easy sign-on for The Things Network [25] for management and monitoring
- Tunneling of LoRa functionality to HTTP REST interface
- Easy update of the u.RECS firmware
- Basic user management & authentication

The basic firmware of the ESP32 is based on the Espressif ESP-IDF development framework [26] and Platform.IO [27] The code-structure is object-oriented and as much as possible similar to the RECS_Master code structure of the RECS|Box and t.RECS firmware. But as we need to develop the firmware for the EPS32 in C++ (not it Java like the RECS_Master), the whole firmware needs to be developed from scratch. Still, the look and feel of the Web GUI as well as the REST-API is as close as possible to the RECS_Master, but reduced to the minimal function set to keep it small and reactive.

The power management and power sequencing of the SoC modules are being implemented as documented in the SoC datasheets of the Nvidia Jetson Xavier NX [5] and of the SMARC standard [28]. Both standards are publicly available after registration. For the Nvidia Jetson Xavier NX module, there is a small 8-Bit microcontroller on the eval board that does the power sequencing, but we wanted it to be directly managed by the ESP32 to have it under full control and get rid of the extra microcontroller. The SMARC standard allows users to select various boot sources which have been implemented. Still, as there are different vendors, some microservers behave differently and we needed to adapt the firmware from time to time to get all microserves to boot up correctly.

The Gigabit Ethernet switch needs a correct power sequencing and a basic configuration to function as desired. This is all done via an SPI connection between the ESP32 management microcontroller and the Ethernet chip. It is planned to also store user-specific settings and restore them after power-cycle. The major configuration that can be changed are the VLAN settings of the Ethernet switch. Our default is that all Ethernet ports share the VLAN 1.

The Web GUI has been developed from scratch and features a very simple look and feel, similar to the RECS|Box and t.RECS websites. It has some nice graphs to show the last 60 seconds of sensor readings like power consumption of the SoC modules and temperatures. The management interface is either available via WiFi or Ethernet, the IP address can be either assigned via DHCP or static. Included in the webserver is the REST API which is publicly defined at the Christmann recswiki website [29] As the REST API behaves like the well-known RECS|Box and t.RECS REST APIs, Kenning can directly be used for all types of RECS hardware.

A major security issue of nearly all IoT devices is the old and not maintained firmware. Christmann plans to support the u.RECS firmware for the next years and implemented an automatic 1-Button Updater into the firmware. It retrieves the latest firmware from the Christmann webserver, checks integrity and flashes it onto the u.RECS in a second partition. If the firmware update should break anything, a rollback mechanism restores the last state.

4 VEDLIoT Microserver

4.1 COTS Microservers

Within VEDLIoT, we use a mix of commercially available microservers from the market and self-developed microservers to ideally fit the needs of the individual applications and to reach highest energy efficiency. Due to the modular design of the RECS systems, this is easily possible. In early stages, we have been able to simply use evaluation systems from different vendors, build up a first mockup and later just swap the microservers to our own platforms to enable the full advantages of tight integration, communication, power measurements, ML accelerators, etc.

A comprehensive description of the used microserver form factors like COM Express, COM HPC, SMARC 2.1, Xilinx Kria K26, Nvidia Xavier AGX, Nvidia Jetson NX and Raspberry Pi Compute Module has been provided in chapter 4 of D2.2 (Hardware platform architecture). In addition, a list of available microservers is presented in D7.4 (Final report on testbed deployment and maintenance) [30], also several accelerators have been described in detail in D3.3 (Evaluation of the DL accelerator designs) [31]. Since then, the following microservers and accelerators were purchased and are available for testing, development and optimization inside the VEDLIoT consortium. Some of them have already been worked on very intensively, like the FPGA-based SMARC module Seco SM-B7 which has been chosen as the base for the DL accelerator design as described in D3.2 (Initial report on the DL accelerator design) [32].

Name	**Company**	**Description**
GPU-based		
Xavier NX 16GB	NVIDIA	Computer on Module + Carrier
Orin NX 16GB	NVIDIA	Computer on Module + Carrier
ASIC-based		
Hailo-8 M.2	Hailo	M.2 2280 AI accelerator module
AI Core XM 2280	AAEON	M.2 2280 AI Edge Computing Module with 2x Intel Movidius Myriad X VPU
ARM		
ROM-5722	Advantech	SMARC 2.1 Module with NXP i.MX 8M Plus with ARM 4-Core Cortex-A53 / M7 + NPU
conga-SMX8-Plus	Congatec	SMARC 2.1 Module with NXP i.MX 8M Plus with ARM 4-Core Cortex-A53 / M7 + NPU
x86		
SOM-2532	Advantech	SMARC Module with Intel Atom x6425E 16GLPDR4 32GeMMC
SMARC-sXAL4 E2	Kontron	SMARC Module with Intel Atom x7 E3950, 8GB LPDDR4, 32 GB eMMC pSLC
COM HPC Server D	Congatec	COM HPC Server Size D Microserver with Intel D-1746TER (Icelake D LCC)
FPGA-based		
SM-B71	SECO	SMARC Module with Ultrascale+ ZU4EG Quad Core+GPU @1,30GHz, PS DRAM 2 GB, PL DRAM 512M, eMMC 4 GB

4.2 Tentative Microservers

In this section, different possible microserver designs are presented. During the runtime of the project the availability of electronic components was a pressing worldwide issue which we tried to deal with as best as possible. Therefore, we didn't only propose one or two microservers that we planed to build, but multiple promising candidates (cf. D4.1). Finally, only two candidates could be selected due to real available chips within the timeframe of the project. The bases for this selection are the benchmarking activities performed within WP3 and displayed in deliverable D3.3. In this section we are showing the microserver candidates that were envisioned but couldn't be realized within the project.

4.2.1 HX40416 (SMARC)

The Coherent Logix HX40416 is an accelerator based on a communication matrix that consists of data memory routers (DMR) and processing elements (PE) which is called HyperX architecture by the manufacturer. A detailed insight of the accelerator is given in Deliverable D3.1 Section 7.5.4. A training was performed that provided hands-on experience using the smaller already available HX40408 to assess its performance further. Unfortunately, Coherent Logix wasn't able to provide samples of the HX40416 in time for the development of a microserver.

4.2.1.1 Module Overview

The HX40416 was foreseen as SMARC module to be compatible with the u.RECS system. The chip is optimized for high energy efficiency by using only the minimum necessary peripherals needed for operation. As shown in Figure 28, the only peripherals will be 4x LPDDR4, real-time clock, GbE Phy and USB to SATA converter. Not shown is the power management of the system, which is obligatory.

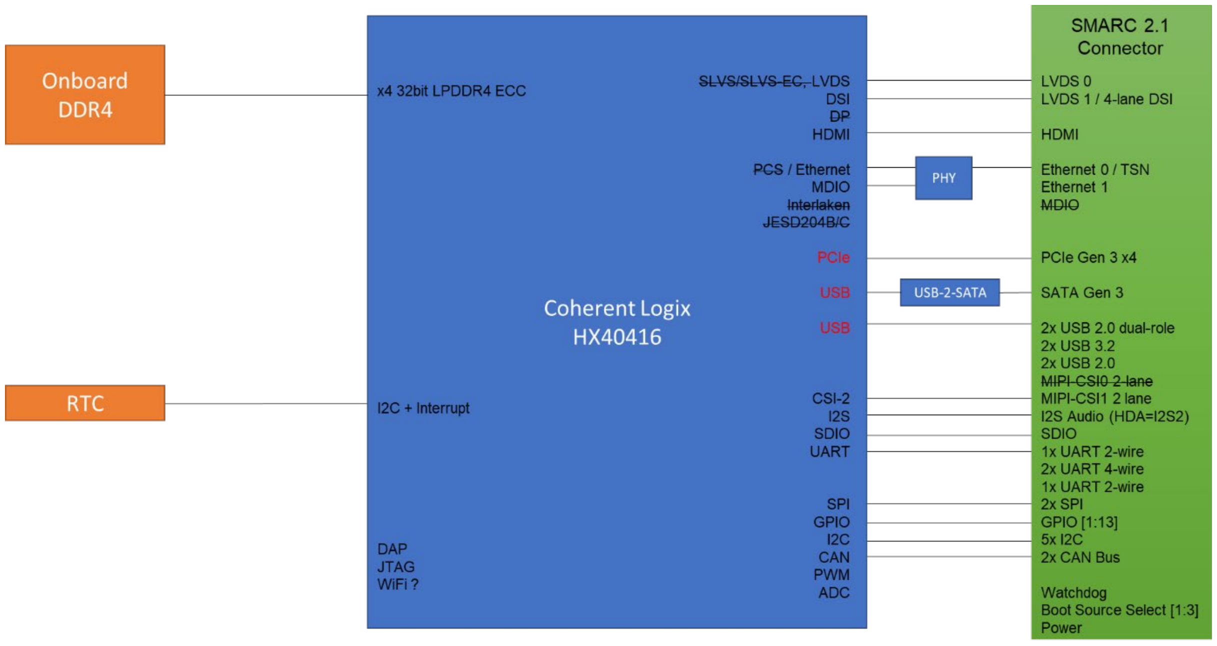


Figure 28: Blockdiagram of the forseen SMARC module showing the main components

Due to the military background of the Coherent Logix accelerator, the communication interfaces are different from commercial embedded products. E.g. USB is not implemented in the RISC-V processing system of the chip. This makes it difficult to implement all interfaces of the SMARC2.1 standard. The module would lack USB2.0 and the second Ethernet port, also some UART, SPI and I2C interfaces would not be connected. But still, it could provide adequate interfaces for the success of the project, like CSI for the connection of cameras and GbE for external communication.

The following list shows the key features of the HX40416 SMARC module:

- Coherent Logix HX40416
  - 4x RISC-V SiFive 54 @ 1500 MHz
  - 416 Processing Elements (2,496 GMACS INT16 @ 1500 MHz)
  - 20 Mbyte SRAM
- 4x LPDDR4-3200 32-bit (4x 12.8 GByte/s)
- Battery-powered RTC
- Communication: PCIe x4 Gen.3, GbE, USB3.0
- Video: CSI, LVDS, HDMI

4.2.1.2 Device Configuration

Similar to an FPGA, not all interfaces are connected to the processing system of the chip. In this case the RISC-V has quite limited communication capabilities as shown in Figure 29. All high-speed interfaces are directly attached to the communication matrix of the ML accelerator. An advantage can be a CSI camera stream directly feeding the processing elements, while a disadvantage is the higher complexity utilizing a PCIe or USB3.0 interface from the RISC-V.

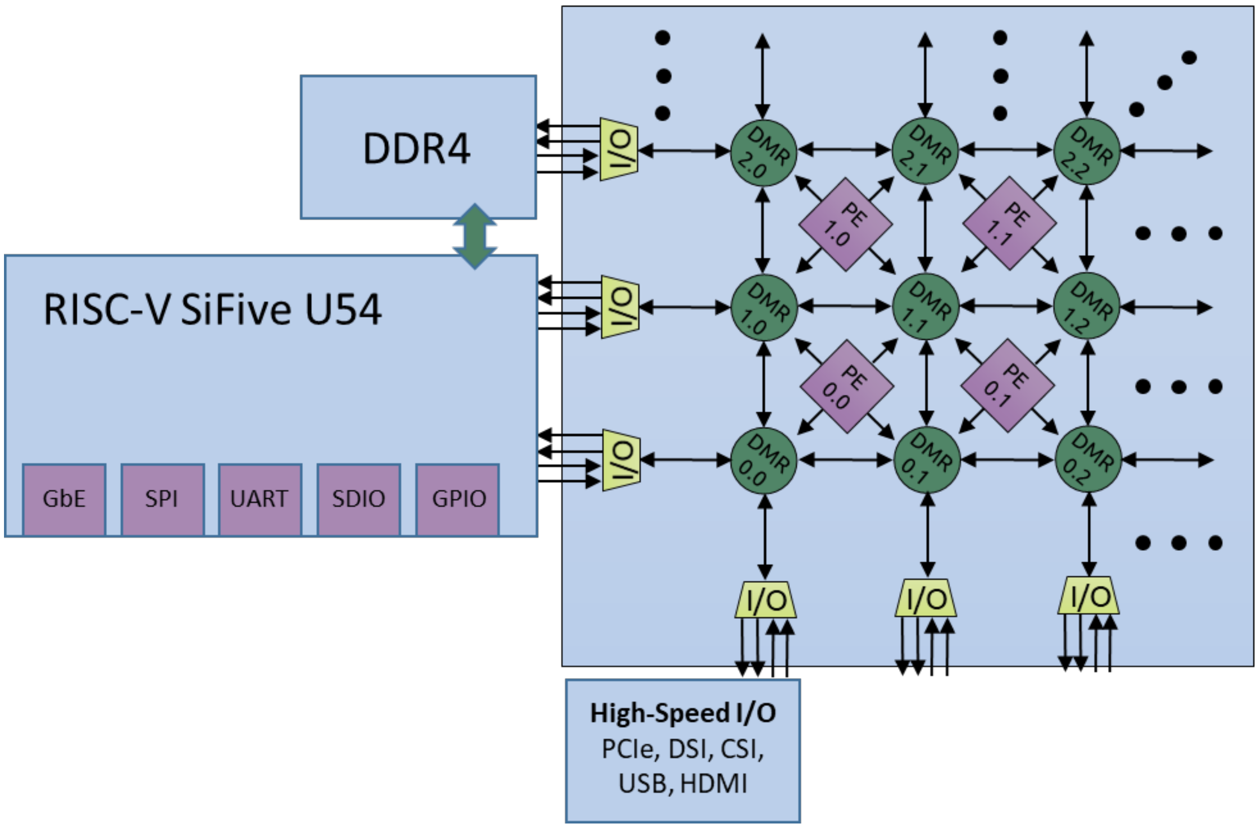


Figure 29: Blockdiagram of the Coherent Logix HX40416 showing I/O connections [33]

Already at design time of a module it must be taken into account were to physically attach the high-speed I/O interfaces to save valuable resources in the ML accelerator. E.g. PCIe and USB3.0 should be placed close to the RISC-V processing system in order to keep the latency inside the DMR matrix as low as possible.

4.2.1.3 Software

The HX40416 needs separate software for the processing system and the ML accelerator. The RISC-V based processing system is capable of running a small Linux operating system. This can be booted from an SD-Card via the SDIO interface inside the processing system. Once the chip is powered, it boots automatically and becomes available via the Ethernet interface. The RISC-V can then program the ML accelerator and attach the high-speed I/O interfaces. Coherent Logix provides a proprietary toolchain to compile neural nets from TensorFlow models.

4.2.2 NXP i.MX 95 + Hailo-8 accelerator (SMARC)

In addition to the other envisioned microservers, a SMARC based microserver with a Hailo-8 accelerator seems to be a very good choice for several use cases, based on our recent benchmarks of the Hailo-8 ML accelerator. The benchmarks confirm that the Hailo-8 chip is one of the most attractive ML accelerators at the market at the moment, but only available as separate M.2 cards – not on a module together with a CPU/SoC. Therefore, we are in close contact with the company Hailo, negotiating to potentially get the chips and hardware development support to build an ML accelerated microserver.

4.2.2.1 Module Overview

Figure 30 shows the block diagram of an envisioned SMARC microserver with a NXP i.MX 95 SoC as the host and a Hailo-8 ML accelerator that is coupled via PCI Express. Both chips will reside on the same SMARC module. The NXP i.MX 95 is planned to become available beginning of 2023, the Hailo-8 accelerator is already available.

To verify the performance and energy efficiency expectations as well as the general architectural design, we plan to build up this scenario with the purchased M.2 Hailo-8 accelerator and a similar SoC chip on a SMARC module that will become available soon and compare it against the other alternatives.

This design has two disadvantages, compared to the other designs: The NXP i.MX 95 has only one PCIe lane which is used to connect the Hailo-8, but the Hailo-8 could be attached with four lanes, so there might lead to a small performance drop in data-intensive applications. The other disadvantage is, that the only available PCIe lane is used to connect the Hailo-8 chip and thus there is no free PCIe lane to satisfy the SMARC external PCIe interface. Therefore, we might change the NXP i.MX 95 later to a comparable module that has more PCIe lanes.

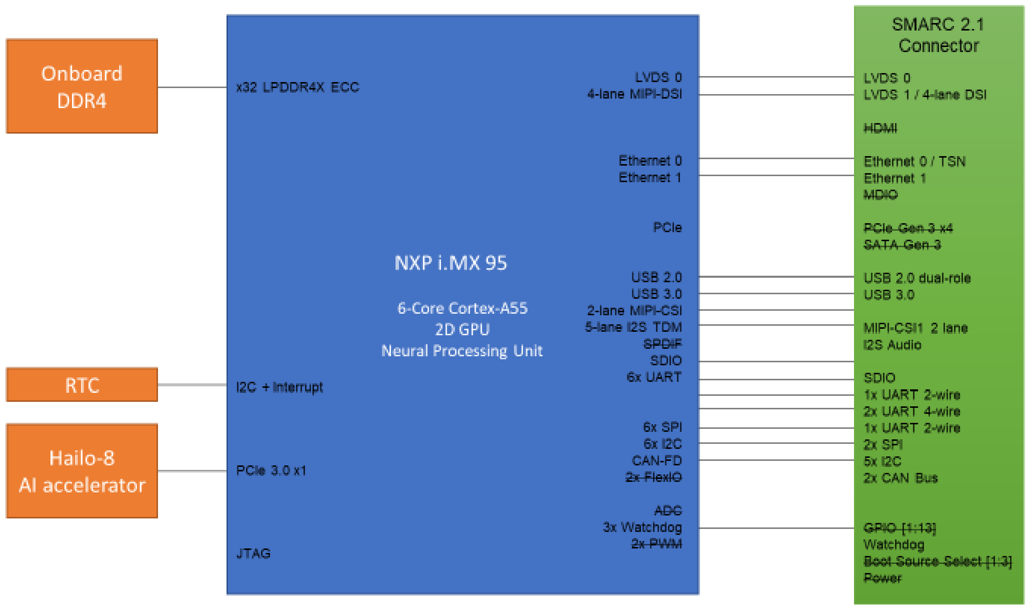


Figure 30: i.MX95 + Hailo-8 SMARC microserver

4.2.2.2 Software

From a software point of view, both the NXP i.MX 95 and the Hailo-8 need to be supported. We expect to get a standard Linux on the NXP SoC up and running quite easily, potentially with the support from NXP. Once the NXP SoC is up and running, the Hailo-8 accelerator should appear as PCIe device and can be used the same way as the M.2 attached accelerators can be used. The Hailo-8 software has already been used by UNIBI in a first evaluation test.

The core of the Hailo-8 software environment is the Hailo Runtime (HailoRT). It not only contains the firmware and driver for the module, but also includes a virtual python environment containing all the packages, tools and fully functional examples required to perform inference on the Hailo-8 accelerator. In addition to the python environment, the run-time also provides a C/C++ interface and corresponding examples. The Hailo model optimizer provided by the HailoSDK is required to prepare a new network for use with the Hailo-8. This python-based tool enables users to optimize, quantize and finally compile, e.g., Tensorflow or ONNX models for the Hailo-8.

In addition to the run-time and SDK, an extensive model zoo is also available [34]. It contains a large number of different models that are compatible with the Hailo-8. The models contain among others, Resnet50, MobileNet v3 and Yolo v4. With the included Python script, these models can on the one hand be easily optimized and compiled for use on the Hailo-8, and on the other hand, their respective performance can be measured when executed on the module.

4.3 Realized Mircoservers

4.3.1 AVRSOM

In this section we introduce ARVSOM, Antmicro's fully open-source system-on-module powered by the RISC-V architecture and featuring the StarFive JH7100 SoC. As a Founding Member of RISC-V International, Antmicro has played a pivotal role in shaping the RISC-V ecosystem from its early days. The ARVSOM open-source hardware shown in Figure 31 delivers unparalleled openness, reusability, and functionality across diverse verticals.

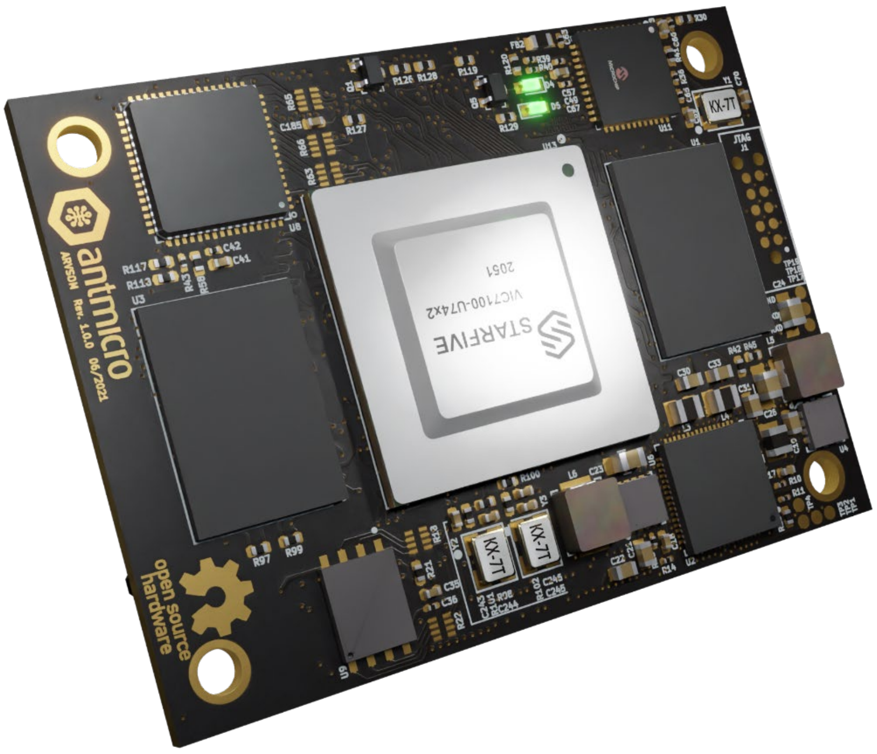


Figure 31 Front side of the ANTMIRCO AVRSOM module

It is compatible with Scalenode [35] – Antmicros's open hardware baseboard targeting the server room and other applications. Together they form an easily scalable compute platform for building robust runners for CI and similar workloads, as well as creating custom, open source hardware and software based systems for edge AI applications, which Antmicro has been building for its customers for more than a decade.

Designed to cater to a wide range of use cases and providing unmatched flexibility, ARVSOM stands as the latest testament to Antmicro's proficiency in developing practical and easily modifiable technologies within the realm of open source.

4.3.1.1 Module Overview

Sitting at the heart of ARVSOM is the StarFive JH7100 system-on-chip - the first Linux-capable RISC-V SoC that is intended for mainstream, general-purpose edge applications and is expected to be used in millions of devices to be built on the open source RISC-V ISA. The module is offering the following spec:

- RISC-V U74 dual-core with 2MB L2 cache @ 1.5 GHz
- Vision DSP Tensilica-VP6 for computing vision
- NVDLA Engine 1 core (configuration 2048 MACs @ 800MHz – 3.5 TOPS)
- Neural Network Engine (1024MACs @ 500MHz – 1 TOPS)
- VPU – H.264/H.265 decoder up to 4Kp60, dual-stream decoding up to 2Kp30
- JPEG encoder/decoder
- Audio Processing DSP and sub-system

Module's design is available in GitHub repository [36] and is licensed on the permissive Apache-2.0 license. Among other platforms such as the u.RECS, the ARVSOM module is compatible with Antmicro's Scalenode - rack-optimized baseboard for computing modules. Scalenode baseboard with ARVSOM module eases building RISC-V computing clusters.

The main part of the module is the StarFive JH7100 SoC, a block diagram showing the internal connections of the module can be seen in Figure 32. The module is pin compatible with Raspberry Pi Compute Module 4 (CM4). It provides the following communication interfaces exposed on dual DF40 board-to-board connectors:

- HDMI
- Gigabit Ethernet
- USB 3.0
- 2x MIPI CSI-2 (2 lanes each)
- I2C
- MIPI DSI (4 lanes)
- SD/SDIO/eMMC
- 26 GPIOs

The module provides 8 GByte of LPDDR4 RAM memory (2x4 GByte) and 32 MByte QSPI NOR Flash memory.

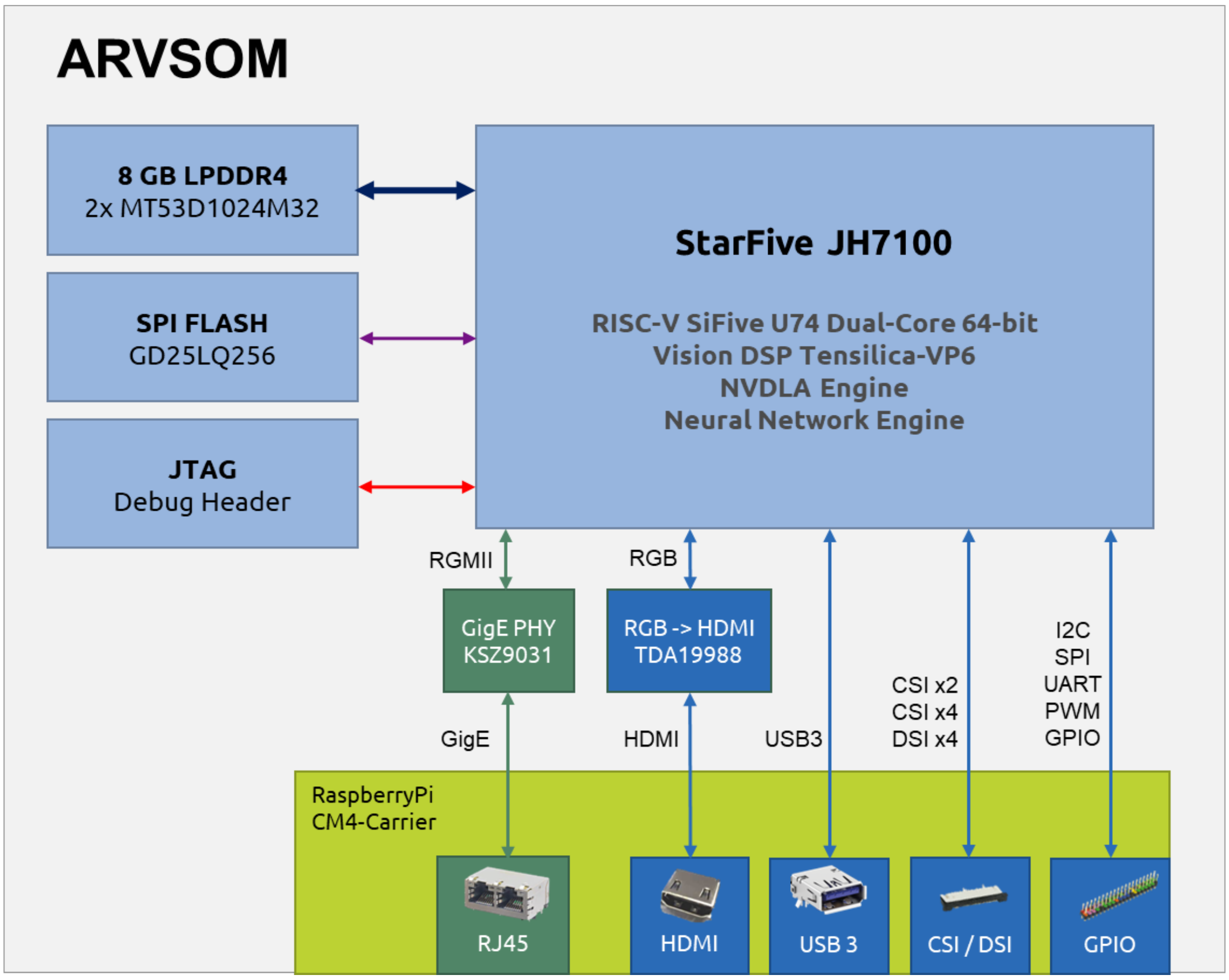


Figure 32 Block diagram of the ARVSOM

4.3.2 VERSAL Edge.AI COM (SMARC)

The AMD Versal series is the latest generation of reconfigurable architectures, introduced by AMD, called Adaptive Compute Acceleration Platforms (ACAP). The Versal ACAPs combine an ARM processing system with a programmable logic fabric and a variety of I/O interfaces. Additionally, new DSP engines and optional AI engines are integrated, together with a dedicated network on chip infrastructure that ties all components together. The combination of flexibility of the programmable logic and compute power in the DSP or AI engines makes the Versal architecture a perfect choice for VEDLIoT as a complement or successor of the current UltraScale+-based FPGA modules. This section describes intrinsics of the new VERSAL Edge.AI COM shown in Figure 33.

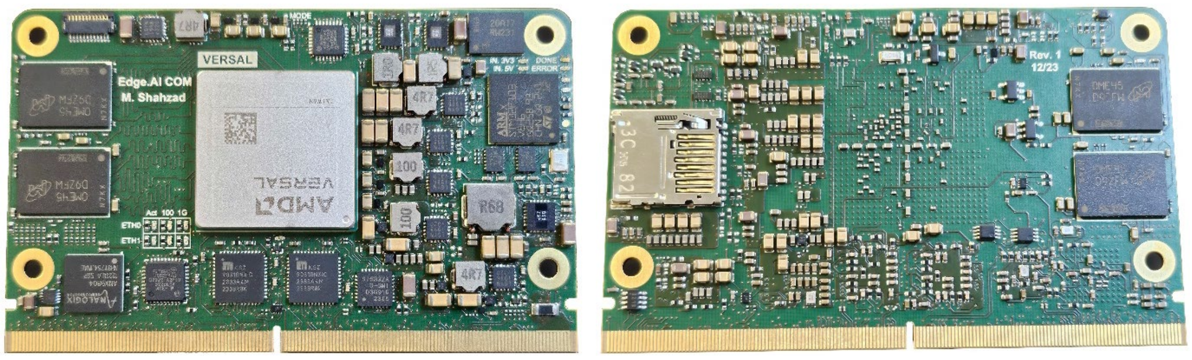


Figure 33: Front- and Backside of the VERSAL Edge.AI COM

4.3.2.1 Module Overview

The small to mid-range Xilinx Versal [37] devices can be integrated into the SMARC module, which is compatible to the u.RECS system. The wide variety of I/O interfaces, the Versal offers, allows to directly connect nearly all interfaces the SMARC standard specifies, as shown in Figure 34. Main peripherals on the module are the RTC and 8 GB of LPDDR4 memory. Within VEDLIoT the AMD Versal VE2302 was chosen from requirements of the use-cases. The VE2302 is the smallest Versal that is large enough to support the DPU IP-core described in deliverable D3.3. It enables the execution of large AI-models like the YOLO model [38] used in the Smart-Mirror use-case. The module comes with the following specification:

- AMD Versal VE2302
  - Dual-core Arm® Cortex®-A72, 48 KB/32 KB L1 Cache w/ parity & ECC; 1 MB L2 Cache w/ ECC
  - Dual-core Arm Cortex-R5F, 32 KB/32 KB L1 Cache, and 256 KB TCM w/ECC
  - 256 KB On-Chip Memory w/ECC
  - 34 AI Engine-ML
  - 464 DSP Engines
  - 329.000 System Logic Cells
- 8 GB LPDDR4 memory
- 64 MB QSPI Flash
- uSD-Cardholder
- Real-Time-Clock
- Board Management Controller STM32F103
- JTAG Debug Header
- SMARC interfaces
  - 2x Gigabit-Ethernet via KSZ9031 PHY
  - HDMI via ANX9804 RGB->HDMI Inverter
  - PCIe x4 Gen.4
  - 4x USB 2.0 via USB Hub USB2514B
  - CSI x2/x4
  - 2x I2C
  - CAN
  - SDIO

- UART
- CAN

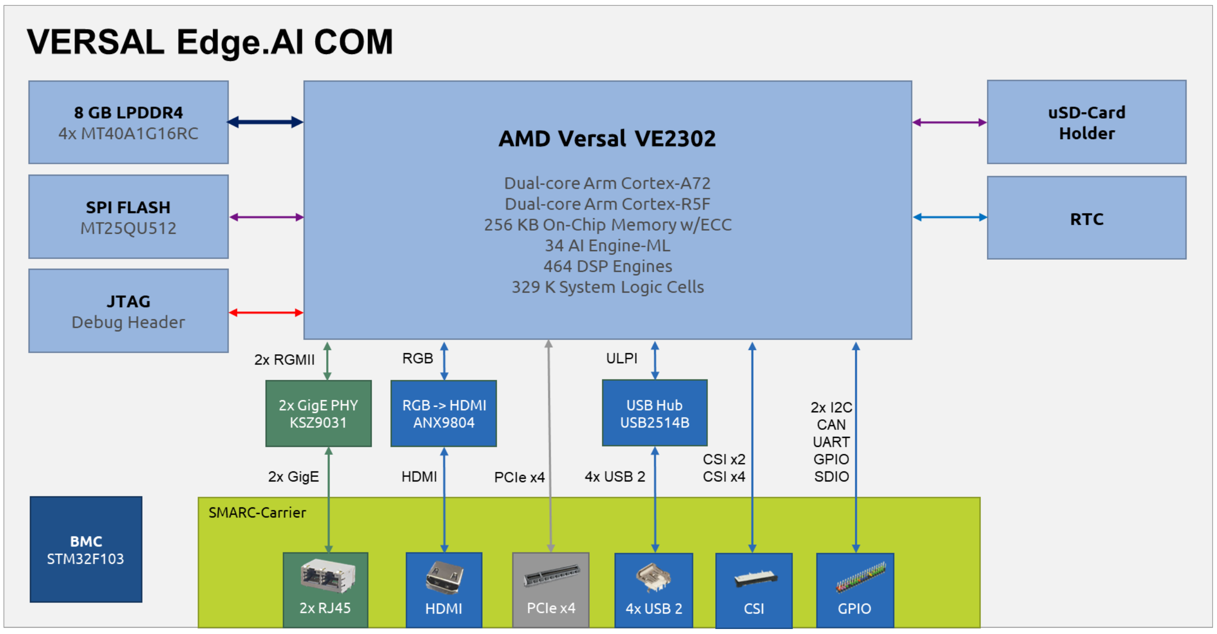


Figure 34: Block-diagram of the VERSAL Edge.AI SMARC module showing the main components

4.3.2.2 Architecture Overview

The Xilinx Versal architecture consists of five main components: a processing system, a programmable logic, DSP and AI engines, a network on chip and the I/O interfaces as shown in Figure 35. Like the Zynq Ultrascale+ architecture the processing system is based on ARM cores. In the Versal ACAPs, these are a dual-core Cortex-A72 application processing unit and a dual-core Cortex-R5 real-time processing unit. The programmable logic provides the typical FPGA resources like LUTs, Flip-Flops, Block-RAM and UltraRAM blocks. The DSP engines offer various operations like a multiplier and an accumulator; new with the Versal architecture are three-element vector fixed-point dot-product, complex multiply-accumulate and single-precision floating-point multiplier and adder. The AI engines are organized in a 2D-array. Each AI engine consists of a SIMD and a VLIW processor, a 32KB memory that is shared between neighboring engines and a stream interface. The I/O part offers interfaces like memory controller, Ethernet, UART, MIPI and others. Finally, the network on-chip offers a high bandwidth connection between all these components.

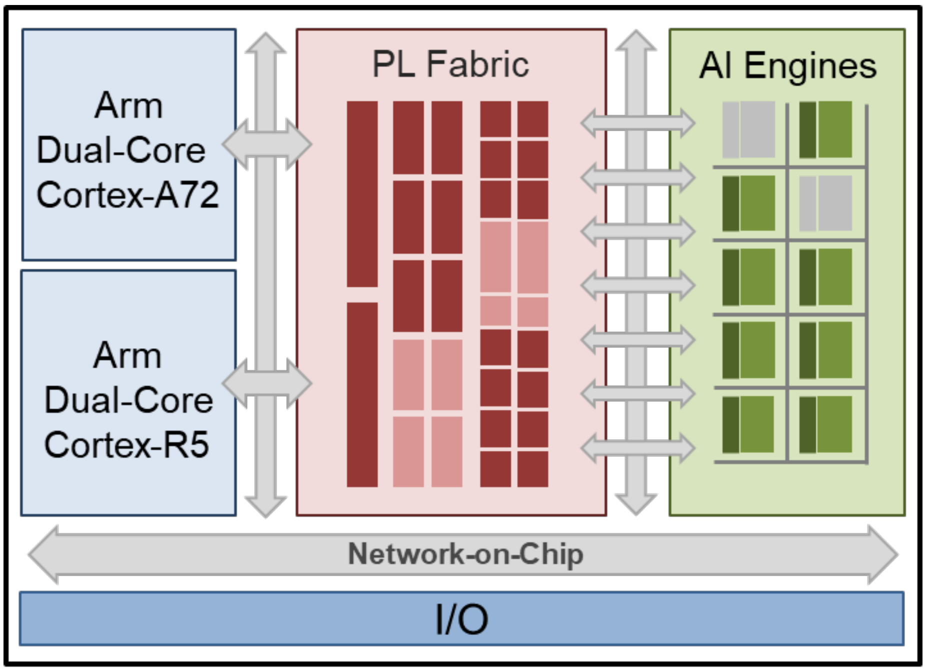


Figure 35: High-level architecture of the AMD Versal devices [37]

4.3.2.3 Tool-flow

The development of applications for the Versal chips is integrated into the AMD tool-flow for FPGAs and AMD SoCs using Vivado [39] and Vitis [40], which is already used for the Zynq Ultrascale+ SMARC module. The ML flow is integrated into the Vitis-AI flow used for the AMD DPUs. Matching to the new Versal architecture, AMD also provides a new DPU version that utilizes the AI engines to achieve a better performance in comparison to the DPU utilizing only PL resources.

The Linux system running on the ARM processing system can be booted from an SD card over SDIO and accessed via Ethernet. As with other AMD SoCs, the processing system can then configure the programmable logic as well as the DSP or AI engines. This makes the system highly flexible since new configurations can be loaded at run-time from the SD card or over the Ethernet interface.

4.3.2.4 Firmware

The Versal Edge.AI COM features a “board management controller" (BMC) that, coordinates power sequencing, provides additional interfaces and surveils the power budget of the system. It is the low-level interface, that is available on the SMARC connector, while the Versal is powered-off. It handles all side-band signals, e.g. Power-On request and reset. In addition, it monitors different supply rails for, e.g. CPU and RAM The captured data is available either from the OS running on the Versal or via I2C from an external management system like the one built into the u.RECS. Other monitored parameters include temperatures of different critical components on the module and the tacho signal of a connected fan.

A Board Support Package (BSP) for the Versal on the microserver modules is already available. It is based on a customized Linux kernel accompanied by an appropriate Device Tree, including the microserver module's additional hardware, e.g. the GbE PHYs. As such, the end-user of the module can work with a familiar Linux operating system as a starting point.

5 Conclusion

This document describes the architecture of the VEDLIoT hardware platform, refining and detailing the overview of the platform that was reported in Deliverable D2.2. The VEDLIoT hardware platform covers the full range of IoT systems, from embedded or far-edge deployments, all the way up to cloud-based systems. This allows to use the platform for hybrid use-cases such as the VEDLIoT automotive use-case, requiring components in the cloud/near-edge, as well as in the embedded/far-edge domain.

For the cloud system, the RECS|Box, an extension of the low-power microserver platform to support newer microservers, such as the Jetson Orin NX, was designed. This allows the deployment of very dense scale-out systems with up to 144 microservers. In addition to this, the t.RECS system is available, as outlined in Section 2.2, providing a flexible platform for small cloud or near-edge use-cases, integrating a powerful communication infrastructure with capabilities similar to those of the RECS|Box, however, in a smaller form factor.

The novelty in VEDLIoT is the introduction of the u.RECS platform to the RECS family. This is a flexible modular platform that supports multiple, heterogeneous microservers in a compact form factor for energy efficient solutions. It supports a wide range of low-power microservers based on industry-establish COM standards such as NVIDIA Jetson, SMARC, CM4 or AMD Kria. In addition, the architecture offers flexible interfaces to include additional accelerators, e.g. via USB or M.2, as well as a wide range of communication and sensor interfaces. The u.RECS platform enabled the support for complex Deep Learning tasks with high energy-efficiency implementations that can make it feasible to deliver the required features at the far-edge.

Apart from the platforms, microserver developments within VEDLIoT came up with two new modules, the ARVSOM and the AMD Versal Edge.AI COM. They provide a significant boost in energy efficiency compared to existing solutions is archived by developing microservers with a special focus on ML or DL.

6 References

[1] VEDLIoT, *Deliverable D2.3: Specification for selected pilots / use cases,* 2021.

[2] VEDLIoT, *Deliverable D2.2: Definition of hardware platform architecture,* 2021.

[3] VEDLIoT, *Deliverable D4.1: First report on cognitive IoT hardware platform and microserver development,* 2022.

[4] SGeT, „Smart Mobility ARChitecture (SMARC),“ [Online]. Available: https://sget.org/standards/smarc/. [Zugriff am 14 02 2022].

[5] Nvidia, „Datasheet of Nvidia Jetson Xavier NX,“ [Online]. Available: https://developer.download.nvidia.com/assets/embedded/secure/jetson/Xavier%20 NX/DG-09693-001_v1.5.pdf. [Zugriff am 19 01 2022].

[6] PICMG®, „COM-HPC® Overview,“ [Online]. Available: https://www.picmg.org/openstandards/com-hpc/. [Zugriff am 14 02 2022].

[7] Xilinx, „Aurora 64B/66B v11.2,“ [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/aurora_64b66b/ v11_2/pg074-aurora-64b66b.pdf.

[8] Trenz Electronic, „Homepage,“ [Online]. Available: https://www.trenz-electronic.de/.

[9] NVIDIA, „NVIDIA Jetson Xavier AGX,“ [Online]. Available: https://www.nvidia.com/de-de/autonomous-machines/embedded-systems/jetson-agx-xavier/. [Zugriff am 14 02 2022].

[10] NVIDIA, „NVIDIA Jetson Orin AGX,“ [Online]. Available: https://www.nvidia.com/de-de/autonomous-machines/embedded-systems/jetson-agx-orin/. [Zugriff am 14 02 2022].

[11] christmann informationstechnik + medien GmbH & Co. KG, “RECS|Box Software Interface - Management WebGUI,” [Online]. Available: https://recswiki.christmann.info/wiki/doku.php?id=doc_recs4:software_interface#m anagement_webgui. [Accessed 21.01.2022].

[12] christmann informationstechnik + medien GmbH & Co. KG, “RECS|Box Software Interface - REST API,” [Online]. Available: https://recswiki.christmann.info/wiki/doku.php?id=doc_recs4:software_interface#re st_api. [Accessed 21.01.2022].

[13] christmann informationstechnik + medien GmbH & Co. KG, “RECS|Box Software Interface - Redfish API,” [Online]. Available: https://christmann.github.io/recs-redfish-api/index.html. [Accessed 21.01.2022].

[14] Prometheus, [Online]. Available: https://prometheus.io. [Zugriff am 14 02 2022].

[15] christmann informationstechnik + medien GmbH & Co. KG, „RECS|Box Software Interface – Prometheus Exporter,“ [Online]. Available:

https://recswiki.christmann.info/wiki/doku.php?id=doc_recs4:software_interface#prometheus. [Zugriff am 21.01.2022].

[16] Grafana, „christmann RECS|Box 4.0 Dashboard v02," [Online]. Available: https://grafana.com/grafana/dashboards/14622. [Zugriff am 14 02 2022].

[17] H2020 Project LEGaTO, "D2.3: Final release of hardware architecture, firmware, and communication infrastructure. ICT-2017-1, grant agreement: 780681," 2017-2020.

[18] H2020 Project M2DC, "D2.5: Final report on next-generation microserver system development. ICT-04-2015, grant agreement: 688201," 2016-2018.

[19] S. Tatham, „PuTTY: a free SSH and Telnet client," [Online]. Available: https://www.chiark.greenend.org.uk/~sgtatham/putty/. [Zugriff am 2024].

[20] IEEE, „IEEE 1588 Version 2," [Online]. Available: https://www.ieee802.org/1/files/public/docs2008/as-garner-1588v2-summary-0908.pdf. [Zugriff am 24 09 2008].

[21] USB.org, „USB Power Delivery," 20 11 2019. [Online]. Available: https://www.usb.org/sites/default/files/D2T2-1%20-%20USB%20Power%20Delivery.pdf.

[22] FormFactors.org, „To the microATX Motherboard Interface Specification Version 1.2," [Online]. Available: https://web.archive.org/web/20180205025218/http://www.formfactors.org/developer/specs/mini_itx_spec_V1_1.pdf. [Zugriff am 28 02 2022].

[23] Mini-Box, [Online]. Available: https://www.mini-box.com/M350-universal-mini-itx-enclosure. [Zugriff am 14 02 2022].

[24] Espressif, „ESP32 wi-Fi & Bluetooth MCU," [Online]. Available: https://www.espressif.com/en/products/socs/esp32. [Zugriff am 20 01 2022].

[25] „The Things Network," [Online]. Available: https://www.thethingsnetwork.org. [Zugriff am 10 1 2024].

[26] Espressif, „Espressif IoT Development Framework," [Online]. Available: https://github.com/espressif/esp-idf. [Zugriff am 19 01 2022].

[27] PlatformIO labs, „Platform IO Website," [Online]. Available: https://platformio.org/. [Zugriff am 19 01 2022].

[28] SGeT - Standardization Group for Embedded Technologies e.V., „SMARC Standard," [Online]. Available: https://sget.org/standards/smarc/. [Zugriff am 19 01 2022].

[29] christmann informationstechnik + medien, „RECS|Box Wiki," [Online]. Available: https://recswiki.christmann.info/wiki/doku.php. [Zugriff am 02 02 2022].

[30] VEDLIoT project, „VEDLIoT Deliverable 7.4 - Final report on testbed deployment and maintenance," 2024.

[31] VEDLIoT project, „VEDLIoT Deliverable 3.3 - Evaluation of the DL accelerator designs," 2023.

[32] VEDLIoT project, „VEDLIoT Deliverable 3.2 - Initial report on the DL accelerator design," 2022.

[33] Coherent Logix, „hx40416 Digital Signal Processor Rev.3," 01 08 2021. [Online].

[34] hailo.ai, „HAILO MODEL ZOO," [Online]. Available: https://github.com/hailo-ai/hailo_model_zoo/. [Zugriff am 14 02 2022].

[35] Antmicro, „GitHub atnmicro/scalenode," [Online]. Available: https://github.com/antmicro/scalenode.

[36] Antmicro, „GitHub antmicro/avrsom," [Online]. Available: https://github.com/antmicro/arvsom.

[37] Xilinx, „Versal: The First Adaptive Compute Acceleration Platform (ACAP)," 29 09 2020. [Online]. Available: https://www.xilinx.com/content/dam/xilinx/support/documentation/white_papers/wp505-versal-acap.pdf.

[38] J. Redmon, „YOLO: Real-Time Object Detection," [Online]. Available: https://pjreddie.com/darknet/yolo/. [Zugriff am 01 2024].

[39] Xilinx, „VIVADO ML Editions," [Online]. Available: https://www.xilinx.com/products/design-tools/vivado.html. [Zugriff am 14 02 2022].

[40] Xilinx, „Vitis Unified Software Platform," [Online]. Available: https://www.xilinx.com/products/design-tools/vitis/vitis-platform.html#gettingStarted. [Zugriff am 14 02 2022].

[41] NVIDIA, „https://developer.nvidia.com/embedded/develop/roadmap," 10 02 2022. [Online].

[42] NVIDIA, „NVIDIA Jetson Orin NX," [Online]. Available: https://www.nvidia.com/de-de/autonomous-machines/embedded-systems/jetson-orin-nx/.

[43] ANTMICRO, „Scalenode - server-oriented baseboard for Raspberry Pi 4 compute module," [Online]. Available: https://antmicro.com/blog/2021/04/scalenode-server-oriented-raspberry-pi4-baseboard/. [Zugriff am 2024 01 23].

7 List of figures