AUTONOMOUS ON-BOARD OPERATION FOR THE HIGH-RESOLUTION EARTH OBSERVATION MISSION LISR ON THE ISS

Clemens Horch[(1)], Konstantin Schäfer[(1)], Andreas Brunn[(1, 2)], Atin Jain[(2)], Robin Speck[(1)], Marius Bierdel[(1, 2)] , Frank Schäfer[(1)]

[(1)] Fraunhofer Institute for High-Speed-Dynamics, Ernst-Mach-Institut (EMI), Ernst-Zermelo-Str. 4, 79104 Freiburg, Germany, +49 761 2714 518, clemens.horch@emi.fraunhofer.de
[(2)] ConstellR GmbH, Ernst-Zermelo-Str. 4, 79104 Freiburg, Germany, info@constellr.space

PAPER

The LisR (Longwave Infrared Sensing demonstratoR) payload is mounted on the Nanoracks Experimental Platform (NREP) outside the ISS and is acquiring high-resolution thermal infrared images for deriving highly precise land surface temperature (LST) data. LisR's brain is an advanced data processing unit (DPU). The DPU hardware is based on a heterogenous multi-processor SoCs (MPSoC) that combines a FPGA with several multi-core processors. The DPU core software runs atop of an embedded Linux system and includes low-level functionality like image acquisition, processing and data handling. For automation, the software includes an interpreter for the Lua scripting language. Lua is a lightweight and powerful programming language that is ideally suited for embedding into other software. Small Lua scripts access the functionality of the DPU through an API and implement the high-level functions of the software. The DPU software and its API are designed in a way that makes it almost impossible to accidentally damage the system with scripts. This approach removes the usual concerns of frequent in-flight software updates and enables an agile development workflow. In the LisR mission, the scripting system is used for automated data capturing and transmission based on the geolocation and the payload health.

1 THE LISR INSTRUMENT AND MISSION

The LisR (Longwave Infrared Sensing demonstratoR) mission demonstrates the capability to record the earth's surface temperature at high fidelity within a New Space approach. The payload is mounted outside the ISS on the Nanoracks Experimental Platform (NREP). Its purpose is acquiring high-resolution thermal infrared images for deriving highly precise land surface temperature (LST) data. The instrument is a technology demonstrator for the upcoming commercial satellite constellation "HiVE". The LisR payload has been launched on board of the Cygnus freighter mission NG-17 "Piers Sellers" on February 19, 2022. The installation on the NREP was on March 9, 2022. LisR has been enabled on March 16, 2022 and has been producing image data since this day.
The LisR payload comprises a cryo-cooled thermal infrared sensor, a free form optical assembly and an advanced data processing unit (DPU). Figure 1 and Figure 2 show the LisR payload with and without the outer housing. For details on the LisR mission and we refer to Bierdel et al. [1]. Brunn et al. [2] explains details on the LisR calibration and data processing. The free form telescope is described by Zettlitzer et al. [3] This paper is focused on the DPU, its software architecture and its approach to automated operation.

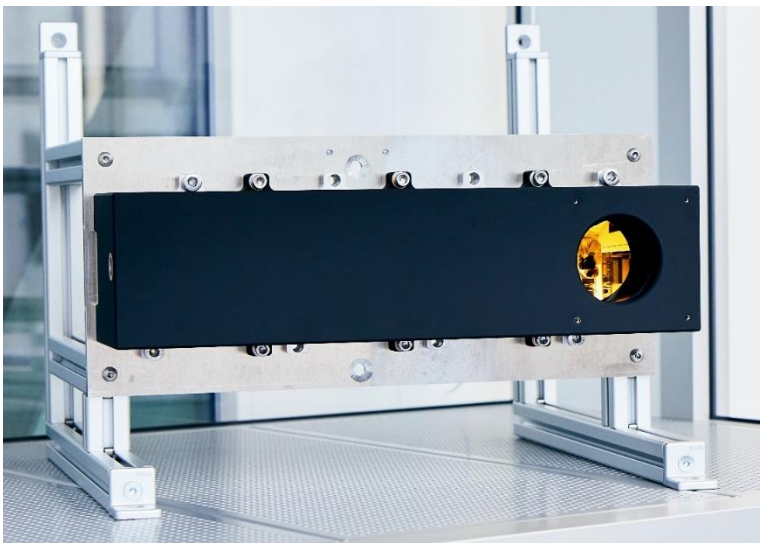

Figure 1. The integrated LisR payload in the cleanroom.

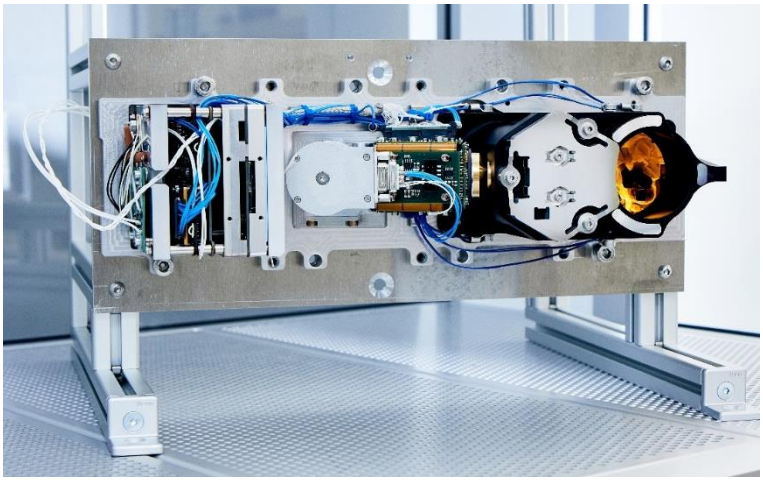

Figure 2. The LisR components from left to right: electronics stack with Data Processing Unit (DPU), Integrated Detector Dewar Cryocooler Assembly (IDCCA) and free form optical assembly

2 DATA PROCESSING SYSTEM

2.1 Hardware Platform and Interfaces

The so-called Data Processing Unit (DPU) of LisR as shown in Figure 3 is the central brain of the payload. Besides handling the communication with NREP and controlling the power system its main task is capturing, compressing and storing of the infrared images. The data interfaces to NREP comprise a serial connection and Ethernet, both transmitted over a single USB connection [4]. The power subsystem is connected via CAN bus and the IDDCA is controlled via I²C. The image data is acquired through a parallel digital interface from the digital ROIC (read-out integrated circuit).

The DPU version in LisR has originally been developed for the ERNST nanosatellite mission [5] which carries a mid-wavelength infrared (MWIR) imager [6]. The DPU's software heritage goes back to the Kent Ridge 1 microsatellite mission [7].

The system is based on a commercial off-the-shelf (COTS) system-on-module (SOM) with the heterogeneous multiprocessor system-on-chip (MPSoC) Xilinx Zynq UltraScale+ which combines a FPGA with several multi-core processors. This system provides enough processing power for generating higher-level data products on-board. At the same time, the resources of the MPSoC are used for increasing the reliability of the COTS system. The platform is completed by various

redundant memory systems for storing software and image data. The hardware platform and its reliability has been previously discussed by Schäfer et al. [8].

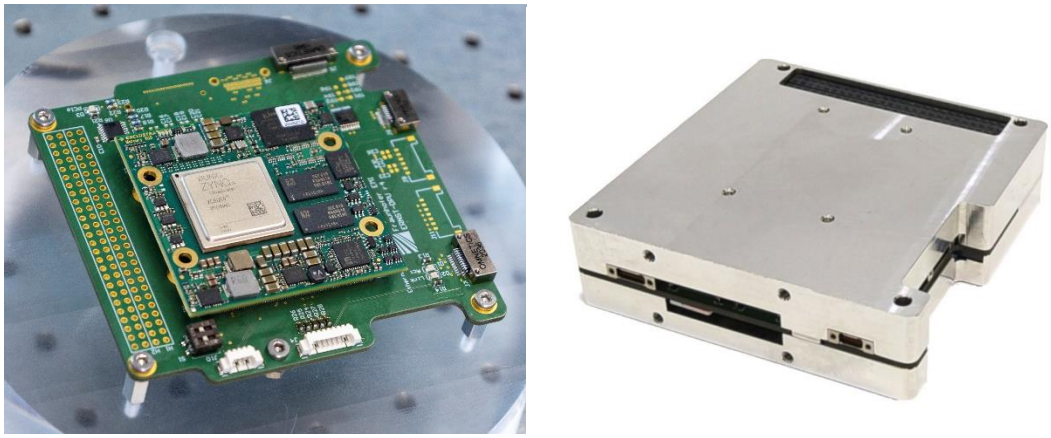

Figure 3. DPU electronics board (left) and integrated device (right)

2.2 Software Architecture

The DPU software follows a layered architecture as shown in Figure 4. The base layer is a tailored embedded Linux system. The operating system especially provides the USB and TCP/IP protocol stacks, file system and I/O drivers. The kernel has been extended with custom drivers for the interfaces implemented in the FPGA fabric.

In the user space of the operating system runs a daemon that includes all necessary low-level functionality. This includes controlling the infrared detector and cooler, capturing, processing and compressing images as well as the management of telemetry data and communication with the NREP. It is referred to as the DPU core software and is implemented in C. This language was chosen for its high performance and its strengths with regards to low-level interaction with the kernel and hardware drivers.

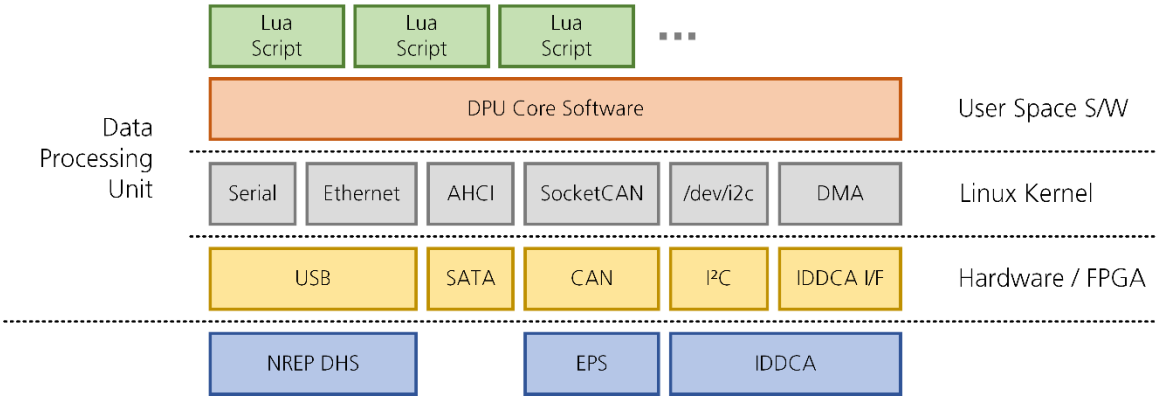

Figure 4. DPU software architecture

For automation of higher-level functions, the software includes an interpreter for the Lua scripting language [9]. Lua is a lightweight yet very powerful programming language that is ideally suited for embedding into other software. All functionality of the DPU is accessible from the Lua language through an API so the operation of the LisR payload can be fully controlled with Lua scripts. Another choice instead of Lua would have been microPython which is also used in the CubeSat community [10, 11]. The decision for Lua has been made because of its lean design and comprehensive C API for embedding. Its small code footprint makes it possible to use the interpreter also in smaller microcontrollers. While this is not relevant for the large DPU system, the possibility of using the

same scripts both on the DPU and a small microcontroller in an on-board-computer of a CubeSat is seen as an advantage.
The development workflow for Lua scripts is optimized for quick iteration. There is a software simulator of the DPU software that can be run on an ordinary PC. All functionality that does not require hardware access can be developed using this simulator. Every developer in a team can do testing with a representative simulation of the target platform without the need for actual hardware. In a second step, all scripts get tested on the engineering model of LisR. There is also a simulator for the NREP interface providing simulated data for orbit position and attitude.

3 AUTONOMOUS OPERATION

The possibility to use the vast communication infrastructure of the ISS makes NREP an ideal platform for experimental payloads. It is possible to access payloads on NREP through an IP-based network with almost no interruptions. In case of LisR there is SSH access to the DPU that can be used for debugging, upgrades and system maintenance. This level of access is normally not available on independent satellite platforms and most certainly not on CubeSats.
On the other hand, all commanding and data downloading can only be done by the operator of NREP, not the owner of the payload. While having direct access is extremely helpful for a demonstrator mission, it always involves human interaction and is not available as an automatable interface. This poses the requirement for on-board autonomy of the payload operation.
The normal operation of LisR with capturing and downloading images could be performed using the traditional approach of commands tagged by time or position. But this approach makes it very hard to react to anomalies like abnormal temperatures or hardware upsets. Using a full programming language can provide the flexibility needed for these tasks and removes the necessity to generate and upload lists of time-tagged commands on a regular basis.
The nature of a demonstration mission with a fast and agile development phase comes also with the need for frequent software upgrades. Upgrading the software on an on-orbit payload is always a difficult and potentially dangerous task. While there are techniques for reducing the risk of software upgraded, it remains complex and may lead to significant testing efforts. This is especially true if formal acceptance tests have to be repeated for each upgrade. In case of LisR, there is always the danger of breaking the communication with NREP through a software upgrade. Since there is only one USB connection, everything that can affect the USB stack or communication protocols can lead to critical failure. In case of small satellite missions with their often very limited uplink bandwidth, the transmission of large upgrade files can also be a problem for frequent software upgrades.
LisR and its DPU solve these problems by integrating a Lua interpreter and a respective programming interface (API). Lua scripts can be very small and by using the Lua compiler they get even smaller. This allows for frequent upgrade even through narrow communication channels. Lua is a Turing complete language and can therefore fulfil even the most complex requirements. The API is designed in a way that it does not expose potentially dangerous functionality to Lua scripts. Even if the interpreter crashes or a script hangs in an infinite loop the core software can still react and fulfil its low-level tasks like the communication with the platform. With a fully featured programming language it is always possible to do potentially harmful operations. But the point is not to protect the DPU against malicious programmers but against accidental errors and negligence.
In the LisR mission, the scripting system is used for automated data capturing and transmission based on the geolocation. In addition, all relevant system telemetry is monitored using scripts. So, if for example the cryo-cooler or any other subsystem would overheat, the script will react by turning it off and waiting until it has cooled down before it tries to resume the nominal operation of capturing images. Baking all this functionality into the core DPU software would have required much more elaborate ground testing and therefore a much longer development time.

4 CONCLUSIONS

In this paper we presented the Data Processing Unit (DPU) of the LisR (Longwave Infrared Sensing demonstratoR) mission on the ISS and its solution for autonomous operation. The need for frequent software upgrades and fast development cycles without the danger of compromising the system's availability is solved by splitting the software into a fixed part with core functionality and a script interpreter for high-level functions. The core software contains all interface code including the communication with the NREP and basic functionality like telemetry collection or image processing algorithms. All code that automates operation is implemented using Lua scripts that can access the core software through an API. The usage of a Turing complete programming language provides maximum flexibility. At the same time, this concept provides good protection against accidental programming errors and subsequently enables a fast and agile development workflow. Finally, upgrades for the scripts are very small in file size and can be uplinked quickly even in small satellite missions with significant bandwidth limitations.

5 REFERENCES

[1] M. Bierdel *et al.,* "Towards high-revisit, high-resolution thermal monitoring: LisR – A land surface temperature monitoring mission on the ISS," in *4S Symposium 2022*, Vilamoura, Portugal, 2022.

[2] A. Brunn *et al.,* "Towards high-revisit, high-resolution thermal monitoring: LisR - Data, Calibration and Processing of Thermal Infrared Data from the LisR ISS Mission," in *4S Symposium 2022*, Vilamoura, Portugal, 2022.

[3] L. Zettlitzer, "Design and fabrication of a miniaturized metallic telescope for Earth observation," in *4S Symposium 2022*, Vilamoura, Portugal, 2022.

[4] Nanoracks LLC, "Nanoracks External Platform (NREP) Interface Definition Document (IDD)," NR-NREP-S0001, Dec. 2021. [Online]. Available: https://nanoracks.com/wp-content/uploads/Nanoracks-External-Platform-NREP-IDD.pdf

[5] C. Horch, M. Schimmerohn, M. Gulde, and F. Schäfer, "ERNST: A 12U Infrared Imaging Nanosatellite based on CubeSat Technology," in *4S Symposium 2018*, Sorrento, Italy, 2018.

[6] C. Horch, N. Schnelle, M. Gulde, E. Watson, M. Schimmerohn, and F. Schäfer, "An MWIR payload with FPGA-based data processing for a 12U nanosatellite," in *Small Satellites for Earth Observation. 11. International Symposium of the International Academy of Astronautics 2017*, 2017, pp. 339–342.

[7] C. Horch, N. Domse, F. Schäfer, and M. Buhl, "High level software development for a stand-alone data processing and storage unit for a microsatellite hyperspectral payload," in *Small satellites for earth observation. Digest of the 10th International Symposium of the International Academy of Astronautics (IAA) 2015*, 2015, pp. 207–210.

[8] K. Schäfer, C. Horch, S. Busch, and F. Schäfer, "A Heterogenous, reliable onboard processing system for small satellites," in *2021 IEEE International Symposium on Systems Engineering (ISSE)*, Vienna, Austria, Sep. 2021 - Oct. 2021, pp. 1–3.

[9] R. Ierusalimschy, L. H. de Figueiredo, and W. Celes, "The evolution of Lua," in *Proceedings of the third ACM SIGPLAN conference on History of programming languages*, 2007, 2-1.

[10] S. Plamauer and M. Langer, *Evaluation of MicroPython as Application Layer Programming Language on CubeSats*, 2017.

[11] M. Holliday, A. Ramirez, C. Settle, T. Tatum, D. Senesky, and Z. Manchester, "PyCubed: An Open-Source, Radiation-Tested CubeSat Platform Programmable Entirely in Python," in *33rd Annual AIAA/USU Conference on Small Satellites*, Logan, UT, USA, 2019.