Student's name and surname: Marcin Jasiukowicz
ID: 176214
Cycle of studies: postgraduate
Mode of study: Full-time studies
Interfaculty field of study: Space and Satellite Technologies
realized at: Wydział Inżynierii Mechanicznej i Okrętownictwa, Wydział Elektroniki, Telekomunikacji i Informatyki
Specialization: Engineering and Management of Space Systems
Profile: Aerospace Technology

MASTER'S THESIS

Title of thesis: Software Defined Radio Data Processing on board of OPS-SAT Space Laboratory

Title of thesis (in Polish): Przetwarzanie danych z platformy radia programowalnego na pokładzie orbitalnego laboratorium OPS-SAT

Supervisor: dr inż. Piotr Rajchowski

STRESZCZENIE

Celem niniejszej pracy jest podsumowanie działania misji Laboratorium Kosmicznego OPS-SAT, jego podsystemów, systemów naziemnych i procedur operacyjnych. W szczególności koncentruje się na podsystemie radia programowalnego, który przejawia możliwości usprawnienia jego operacji.

Głównym celem części eksperymentalnej pracy jest implementacja eksperymentu bazującego na oprogramowaniu, pokazującego koncepcję przetwarzania cyfrowych próbek sygnałów radiowych, z uwzględnieniem ograniczeń narzuconych przez platformę OPS-SAT. Eksperyment jest ostatecznie testowany w rzeczywistym scenariuszu, obejmującym komunikację z Ziemi do statku kosmicznego orbitującego na niskiej orbicie okołoziemskiej.

Rezultatem pracy jest usprawniony proces rejestracji i przetwarzania nagrań radiowych z orbity, składający się z implementacji oprogramowania eksperymentu, instrukcji obsługi, a także procedur operacyjnych umożliwiających zespołowi kontroli misji korzystanie z niego w przyszłości.

Słowa kluczowe: kosmos, satelita, cubesat, radio programowalne

ABSTRACT

The purpose of this thesis is to summarize the operations of the OPS-SAT Space Laboratory Mission, its subsystems, ground systems and operational procedures. In particular, it focuses on the Software Defined Radio subsystem, which provides opportunities for streamlining of operations.

The main goal of the experimental part of the thesis is to implement a software experiment showing a concept for processing the digital samples of radio signals, with regard to imposed limitations by the OPS-SAT Platform. The experiment is ultimately tested in a real world scenario, involving communication from Earth to the spacecraft orbiting Low Earth Orbit.

The outcome is the work is a streamlined process for recording and processing radio recordings from orbit, consisting of software implementation of the experiment, deployment instructions as well as operational procedures enabling the Mission Control Team to use it in the future.

Keywords: Space, Satellite, Cubesat, Software Defined Radio, SDR

Field of science and technology in accordance with OECD requirements: Engineering and Technology, Systems Engineering

ACKNOWLEDGEMENTS

I extend my sincere thanks to my friends and fellow Mission Control Team members for their support throughout this journey: Vladimir Zelenevskiy, Adrián Calleja Vázquez, Tim Oerther, Rodrigo Laurinovičs, Sam Bammers, Marcin Kovalevskij and Georges Labrèche.

Special thanks go to Dave Evans, OPS-SAT Project Manager, for his leadership and guidance.

Fig. 1: OPS-SAT Mission Control Team at European Space Operations Center Main Control Room.

TABLE OF CONTENTS

LIST OF IMPORTANT SYMBOLS AND ABBREVIATIONS

ADCS	–	Attitude Determination and Control System
CCSDS	–	Consultative Committee for Space Data Systems
COTS	–	Commercial of the Shelf
CPU	–	Central Processing Unit
d_{km}	–	Distance between transmitter and receiver [km]
DMA	–	Direct Memory Access
DSP	–	Digital Signal Processing
ESA	–	European Space Agency
ESOC	–	European Space Operations Center
ESTEC	–	European Space Technology Center
ESTRAC	–	European Space Tracking network
f_{GHz}	–	frequency of the transmission [GHz]
FFT	–	Fast Fourier Transform
FPGA	–	Field Programmable Gate Array
FS	–	File System
G_t	–	Gain of the transmitter antenna [dBi]
G_r	–	Gain of the receiver antenna [dBi]
GFSK	–	Gaussian Frequency Shift Keying
GNU	–	GNU is Not UNIX
HPS	–	Hard Processor System
HoPo	–	Horizontal Pointing
IOD	–	In Orbit Demonstration
IQ	–	In-phase and Quadrature
ISM	–	Industry, Scientific and Medical
L_p	–	Free space loss [dB]
L_o	–	Other losses [dB]
LEO	–	Low Earth Orbit
LEOP	–	Launch and Early Orbit Phase
LSF	–	Libre Space Foundation
MATIS	–	Mission AuTomatIon System
MCS	–	Mission Control System
MCT	–	Mission Control Team
MUST	–	Mission Utility and Support Tools
OS	–	Operating System
P_r	–	Power received [dBm]
P_t	–	Power transmitted [dBm]

RAM	–	Random Access Memory
RF	–	Radio Frequency
RX	–	Receive
SAR	–	Search and Rescue
SCOS	–	Satellite Control and Operation System
SDR	–	Software Defined Radio
SIDLOC	–	Spacecraft Identification and Localization
SMILE	–	Small MIssions Laboratory Environment
SSO	–	Sun-Synchronous Orbit
SSTV	–	Slow Scan Tele-Vision
TT&C	–	Telemetry, Tracking and Telecommand
TX	–	Transmit
UART	–	Universal Asynchronous Receiver-Transmitter
UTC	–	Universal Coordinated Time
UHF	–	Ultra High Frequency
UNIX	–	UNiplexed Information Computing System (Operating System)
eMMC	–	embedded Multi Media Card

1. INTRODUCTION

OPS-SAT is a 3 unit cubesat deployed by the European Space Agency into Low Earth Orbit. Its goal is to accelerate innovation within the operational domain for current and future space missions. The satellite is designed to allow external collaborators to run custom software-based experiments on board the spacecraft in a safe and secure manner. Experimenters can take over control over the satellite and implement novel functionalities and algorithms on board. The satellite, together with its supporting infrastructure and teams is known as the OPS-SAT Space Laboratory. Since the launch in 2019 the Laboratory has supported over 250 registered experiments with over 150 successful executions on board.

The author had the great opportunity to join the OPS-SAT Mission Control Team (MCT) as a Mission Control Engineer for a period of 6 months, starting in August 2023. During This time, they have familiarized themselves with the operations of the OPS-SAT Space Laboratory, the inner workings of its subsystems, both on the side of the ground segment as well as the space segment.

In the context of this Master Thesis, the OPS-SAT mission provides opportunities within the Software Defined Radio (SDR) Subsystem. Throughout the mission the subsystem has been underutilized. Moreover, after 4 years in orbit, the spacecraft runs in a degraded state. One of the issues is lowered pointing capability due to degradation of reaction wheels. Together with higher drag coefficient resulting in orbital decay, more and more experiments become impossible to run – especially ones related to the Attitude Determination and Control System (ADCS).

Due to the nature of the SDR subsystem, it provides the mission with a unique opportunity to provide value until the very end of the mission. To make it easier to operate by the MCT, the subsystem requires additional work, which needs to streamline its operations. The operational aspect requires finding solutions for safe storage of the captured radio samples on board, efficient downlink to earth and further processing pipelines on ground. Possibly, some of the processing can be done on board, provided verification of available resources and types of processing.

Overall, The Master Thesis focuses on streamlining the operations of the SDR subsystem on board OPS-SAT Space Laboratory. Special feature of the thesis is ceasing the opportunity to use real hardware currently orbiting in Low Earth Orbit. The outcome of the Thesis is a streamlined processing of data generated by OPS-SAT SDR subsystem.

2. MISSION BACKGROUND

Unified European efforts to pursue scientific progress in space started on June 14th, 1962, when Belgium, Denmark, France, Germany, Italy, the Netherlands, Spain, Sweden, Switzerland and the United Kingdom signed the Convention creating the European Space Research Organisation (ESRO) [1]. The charter called for the creation of several joined technology centers, including ESTEC (European Space Operations Center) in Noordwijk, Netherlands, ESRANGE (European Space Range) in Kiruna, Sweden and others.

The history of the European heart of space operations dates back to 1963, when the European Space Data Acquisition Center (ESDAC) was opened in Darmstadt. In 1967, the center was renamed to its current name, the European Space Operations Center, while also moving to its current headquarters building at Robert-Bosch Strasse 5, Darmstadt [2].

In its first years of operation, it was tasked with supporting European space activities by providing data analysis and processing capabilities. The new headquarters were the home of one of the first supercomputers: IBM 360 [2]. ESOC was responsible for developing and running computer programs for various space-related calculations, including orbital mechanics and data processing for space missions. The organization grew rapidly, hiring scientists, mathematicians, physicists, and support staff to handle tasks such as programming, data analysis, and mission planning [2].

Throughout the years the center operated many scientific missions launched by ESRO and since May 30th, 1975, the European Space Agency. The Mission Analysis office is tasked with finding and calculating orbits for future missions, taking into account their stated objectives and constraints. Once the missions are launched, the Flight Dynamics group tracks their orbits, and provides updates to Spacecraft Operations, which is tasked with controlling the spacecraft to fulfill their objectives. This couldn’t be done without the European Space Tracking network (ESTRAC), consisting of a global network of antennas. The operations of a typical ESA mission is best described by Madeleine Schafer in the book “ESOC - How to survive in space” [2]:

The Orbit Team would calculate the satellite's present and future orbits and produce 'pass' predictions, i.e. information on when and where the satellite would be 'visible' over the ground stations. This information would then be transmitted to the ground stations concerned, so that they would know in which direction to point their antennae. The Control Centre would also receive a computer printout listing all the passes for all the stations. On the basis of this list they would schedule the stations, indicating what was to be done during each pass, which commands to send, which data to receive, which parameters to extract, etc. During a pass, the ground stations would 'track' the satellite and send this data, surprisingly enough called a 'tracking message', back to ESOC. (They would also transmit a lot of other data, but that does not concern us here.) The Orbit Team would then use these tracking messages to compute a new orbit and produce new pass predictions, and - see the beginning of this paragraph.

Fig. 2: Control room at ESOC in 1968 [3].

Although different missions and their orbits have different characteristics, not much has changed in this rather simple procedure loop to this day. With the progress of technology, the center has developed, with help from the commercial industry, complex Mission Control Systems (MCS), aiding in the day to day operations of more and more missions. The progress over the years can be characterized by two factors: the individual orbits of the spacecraft and thus the frequency with which they can be communicated with and the overall diversity of missions which needed to be controlled at the same time.

For example, satellites orbiting in polar orbits, as was the case with some of the first ESRO missions, could be contacted several times a day. With a small number of overall missions, the operations could be handled by fewer teams by sending commands almost directly to the spacecraft. As the number of missions grew, the need for tools grew rapidly in order to keep up with the number of activities to be performed with many spacecraft in small communication windows. On the other hand, spacecraft placed in geostationary orbits benefit from continuous communication windows. As the spacecraft is always in the same spot in the sky, just a few ground stations are needed to provide redundant communication links. The commands can be sent at any moment, and the telemetry is flowing constantly. Due to this, these missions require less automation, but improvements of the MCS can allow the same teams to control more spacecraft at the same time.

Over the years ESA incorporated many new Mission Control Systems like the Multi-Satellite Support System (MSSS) in the 1980s, SCOS-1 and SCOS-2 in the 1990s and finally SCOS-2000 in the new millennium. The systems empowered Mission Control Teams to control more missions than ever before with certainty and safety. The heritage gained over the years by the systems and their respective ecosystems made them trusted, and thus made them the basis of new satellite development.

With the rise of “New Space” and greater than ever variety and number of satellites launching to space, Mission Control Systems play an ever more important role in ensuring safety of all spacecraft in orbit. The European Space Agency has made steady progress in this field for years, but the private sector has been quickly catching up. Especially with the rise of the small satellite market

and launch of first satellite mega-constellations, further technology advancements in the field of satellite operations are crucial to ensure safety of operations in the years to come.

In 2011 OPS-SAT mission was proposed in response to a significant challenge in the space industry: the hesitation to adopt innovative software and operational concepts in satellite missions due to the high risks and costs associated with space operations. The project was devised to address the "has never flown, will never fly" paradox that often hinders innovation in space technology [4]. Traditional space missions prioritize proven, reliable software over new, potentially more efficient solutions because of the unforgiving nature of the space environment and the substantial investments involved. This conservative approach, while ensuring mission safety, also restrains innovation and advancement in space operations [4].

The concept was first proposed to ESOC's Advanced Operations Concepts Office, and a feasibility study was conducted in 2012 using ESA's Concurrent Design Facility. Following overwhelming interest from potential experimenters, the project received funding from ESA's General Support Technology Programme (GSTP) in 2015 with support from Austria, Germany, Denmark, and later Poland. The mission progressed through various design phases, with TU Graz leading a consortium of European companies to design, test, and build the satellite [5].

Worthy of note is that the satellite was proposed and operated fully by ESOC. Typically, all technological development at the agency is held at the ESTEC center. This approach allowed the development of the satellite to be directly opinionated by the operational expertise of ESOC.

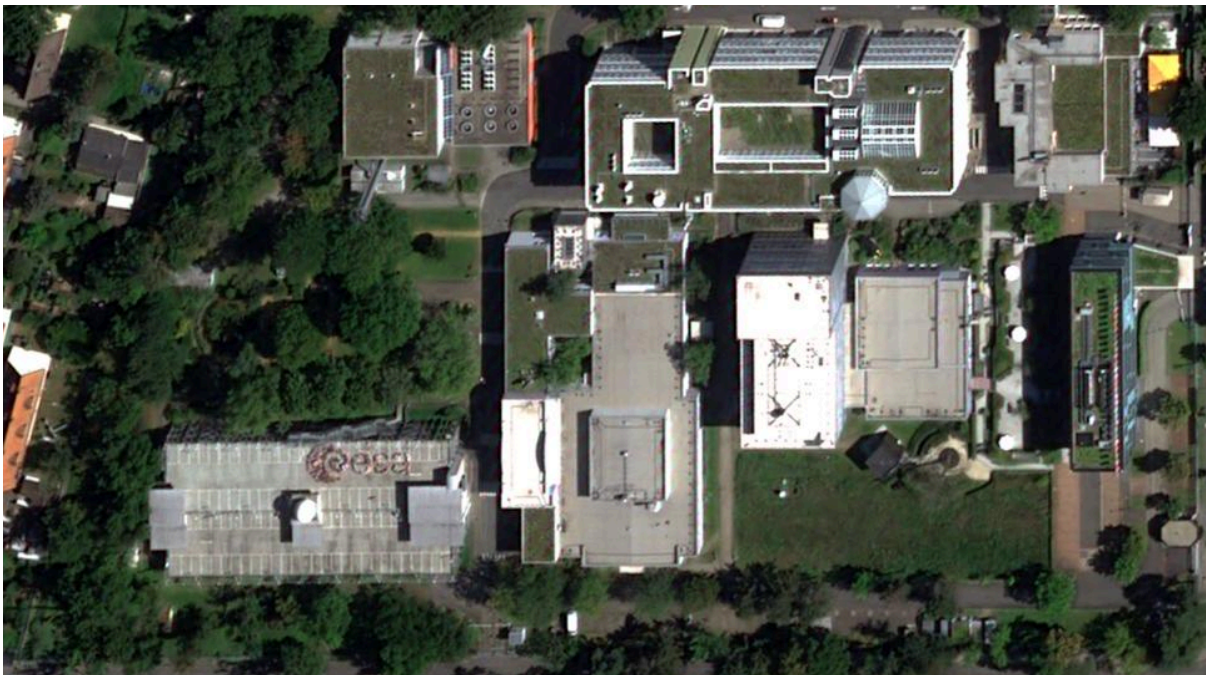

Fig. 3: ESA Sentinel picture of the European Space Operations Centre campus in Darmstadt, Germany. Human logo made by the teams during Team Day 2023 [6].

3. THE OPS-SAT SPACE LABORATORY

OPS-SAT 1 satellite, launched in 2019, is ESA's first mission specifically designed as a flying laboratory for testing and demonstrating cutting-edge space technologies and operational concepts. The mission boasts a unique architecture by joining cutting-edge Commercial-of-the-shelf (COTS) hardware and redundant design ensuring safety of operations at any time [7]. At the time of conception, the mission took advantage of the emerging small satellite “CubeSat” form factor and pushed development of key components bridging the gap between traditional operational concepts while allowing for experimentation and implementation of new approaches.

The purpose of the mission is to provide a safe and reconfigurable platform for in-orbit experimentation. OPS-SAT allows for testing of new software, and operational techniques in a real space environment without jeopardizing larger, more expensive missions. To accelerate the development and validation of future space technologies, OPS-SAT enables rapid prototyping and in-flight testing, helping mature promising technologies, like high-speed communication protocols, and innovative mission operations software, for future ESA missions.

OPS-SAT fosters innovation and collaboration in the space sector by providing an open platform and accessible experiment submission process encouraging and giving means to researchers, engineers, and companies interested in advancing space technology. OPS-SAT embraces openness by providing experimenters with extensive documentation, access to onboard resources, and a supportive community platform for knowledge sharing and collaboration. Most importantly, access to the laboratory is free and unbureaucratic.

The OPS-SAT Space Laboratory consists of two integral parts: The Space Segment - OPS-SAT-1 satellite, as well as the Ground Segment - SMILE Laboratory at ESOC.

Fig. 4: ESA OPS-SAT Mission Patch [8].

3.1. Space Segment

The satellite is a 3 unit CubeSat, measuring 30x10x10 cm. The satellite has two deployable solar panel arrays on its side. The overall systems architecture of the satellite is split into the Satellite Bus and the Payload section. To ensure robustness and mission success, OPS-SAT incorporates redundancy in critical systems. The spacecraft is designed to be highly fault-tolerant, capable of withstanding errors in experimental software and even hardware failures. The architecture continuously monitors for anomalies and can automatically initiate recovery procedures to ensure the satellite's survival.

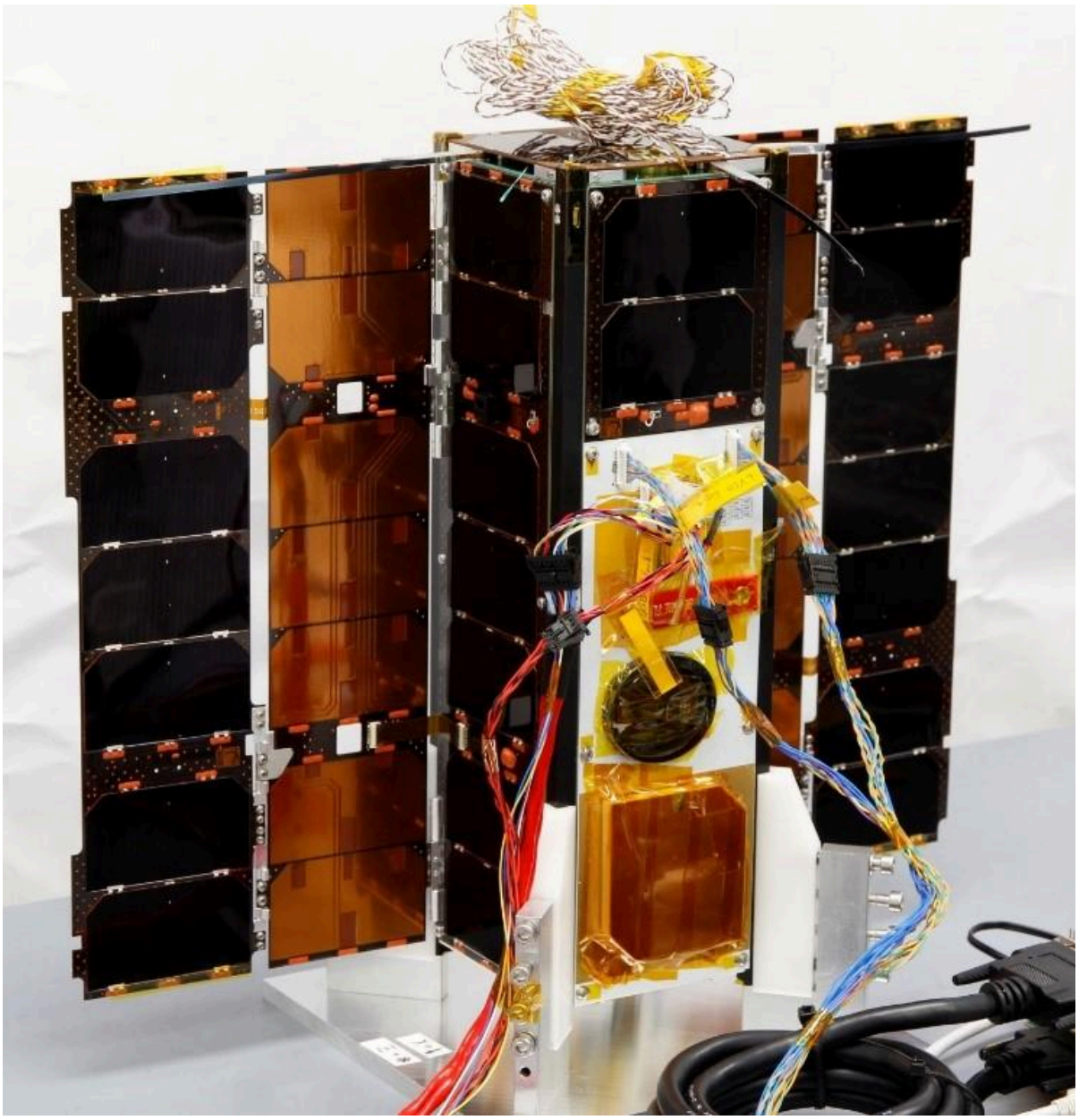

Fig. 5: OPS-SAT Satellite during testing [8].

3.1.1. Satellite Bus

NanoMind On-Board Computer (OBC): The NanoMind OBC is the central command unit of the satellite bus, responsible for executing commands sent from the Ground Segment and controlling the basic functions of the satellite. The OBC is responsible for execution of the mission's Master Schedule. It coordinates the activities of the other subsystems and provides the necessary interface for Telemetry, Tracking and Telecommand (TT&C), ensuring that the satellite operates correctly. Two units of the OBC are installed in a cold-swap configuration. This ensures that at least one unit is operable at a time, allowing for safe software updates of each unit.

Electrical Power System (EPS): The power subsystem consists of deployable solar panels and a power control unit that manages the generation, storage, and distribution of electrical power throughout the satellite. The solar panels can generate up to 30 W of electrical power at peak [9], which is stored in onboard batteries to provide continuous power to the satellite, especially when it is in the Earth's shadow and solar power is unavailable.

Nanocom AX100 UHF Communication Subsystem (Nanocom): The Ultra high frequency (UHF) communication subsystem provides a basic, low-data-rate communication channel (9.6 kbps half-duplex) for TT&C operations. The radio is paired with two antennas working in dipole configuration, providing a close to spherical transmission pattern. This enables communication with the spacecraft in any attitude, which is particularly important during critical phases of mission such as Launch and Early Orbit Phase (LEOP). The UHF subsystem operates in the amateur radio frequency bands tuned at 437.2 MHz center frequency [10], allowing for easy communications with ground stations using simple antenna setups. Thanks to this capability, the mission is followed by the Ham Radio community from all around the world, helping the MCT receive additional telemetry and gain valuable information for operations.

Basic Attitude Determination and Control System (ADCS): The basic ADCS includes sun sensors, magnetometers and magnetorquers. These components work together to determine the position of the sun and control the satellite's orientation by interacting with the Earth's magnetic field. The ADCS provides the minimal hardware ensuring that the solar panels are optimally positioned for power generation and that the antennas are correctly aligned for communication.

Fault Detection, Isolation, and Recovery (FDIR) System: System consisting of two redundant Micro Controller Units (MCU) separate from the OBC and other systems in the Platform section of the spacecraft. The system is a critical safety feature of OPS-SAT, designed to detect anomalies in the satellite's various subsystems and initiate corrective actions to maintain operational stability. The system continuously monitors the health of the satellite, including power levels, communication signals, and processing loads. When a fault is detected, the FDIR system can isolate the affected subsystem, reset components, or switch to backup systems, ensuring the satellite remains functional and can continue its mission [7]. For example, as part of the EPS subsystem, the system implements a Watchdog timer. In case the powered subsystems, especially the OBC, become unresponsive and fail to reset the timer, the system will perform a hard-reboot. This is designed as a last resort measure and will reset the spacecraft to a safe state.

3.1.2. Payload

Satellite Experimental Processing Platform (SEPP): At the heart of the experimental platform, the Altera Cyclone V System-on-Chip integrates a Hard Processor System (HPS) with an FPGA (Field Programmable Gate Array) fabric [11]. The HPS includes a dual-core ARM Cortex-A9 800 MHz CPU with 1 GB RAM (Random Access Memory). The computing platform is paired with 16 GB of embedded Multimedia Card Flash (eMMC) used for persistent storage [10]. The payload section includes two units of the SEPP, called SEPP-1 and SEPP-2 following the redundant design principles of the rest of the satellite [12].

The HPS runs GNU/Linux Operating System (OS) allowing for a great variety of modern software to be run. The OS comes with a set of tools and abstractions known to developers working in many industries outside of the space sector. The available resources allow for high-level programming languages like Python and Java to be run. The FPGA component is reconfigurable, meaning it can be reprogrammed in-flight to adapt to different experiments and processing needs. Additionally, many of the signal lines and buses connecting the various subsystems have been connected to the fabric [10].

The flexibility and performance of the SEPP platform enables rapid testing of new algorithms and processing techniques in a space environment, significantly reducing the time from development to deployment. Moreover, the powerful hardware combination has been touted as having 10x more computational power than previous ESA spacecraft.

Advanced ADCS (iADCS): The advanced ADCS, provides high-precision attitude control capabilities beyond what is possible with the basic ADCS. It includes a star tracker (ST-200), which uses celestial navigation by tracking star positions, and miniature reaction wheels for fine adjustments in satellite orientation [10]. This system allows for precise pointing necessary for high-resolution imaging and optical communication experiments, enabling the satellite to maintain stable alignment for extended periods.

IMS-100 Optical Camera: The IMS-100 is a high-resolution visible-spectrum optical camera capable of capturing still images and videos, which can be used for Earth and celestial observation [10]. The camera's images are processed by the SEPP, where they can be compressed and transmitted back to Earth via the high-speed communication links. The experiments can take advantage of the camera and use the powerful SEPP hardware for object detection, classification and processing of the images.

CCSDS Engine: The CCSDS subsystem on OPS-SAT is responsible for managing communication protocols in accordance with the standards set by the Consultative Committee for Space Data Systems (CCSDS). It includes the CCSDS Engine, a device that ensures data compatibility and efficient communication between the satellite and ESA ground stations [13]. The subsystem facilitates both standard operations and experimental setups by providing a “bypass” of the engine, enabling the implementation of custom protocols by specialized experiments run on SEPP and the FPGA [10]. This system is featured in the payload section, as it was one of the custom developed components,

bridging the gap between available commercial CubeSat components and ESA-mandated standards for satellite missions [4].

S-Band Communication Subsystem: The S-band communication system on OPS-SAT features a Syrlinks EWC27 EC31 transponder that supports full-duplex data transmission with a 256 kbps uplink and 1 Mbps downlink capacity. This system is primarily used for transferring large volumes of data between the satellite and the ground stations, including software updates, telemetry data, and experimental results to its dedicated ground station at ESOC. The system is paired with two patch-antennas placed on opposite sides of the spacecraft, creating two half-spheres of coverage around the spacecraft. The system works within the 2025 – 2100 MHz S-Band, at 2053.136 MHz for receive (RX) and 2229.65 MHz for transmit (TX) channels [10].

X-Band Communication Subsystem: The experimental X-band TX-only communication system allows for ultra-high-speed data downlinks of up to 50 Mbps, making it ideal for data-intensive applications such as large-scale data downloads. The X-band transmitter operates at much higher frequency (8177.5 MHz)[10] than the UHF and S-band systems, providing greater bandwidth and faster data rates. The subsystem is paired with an antenna on the -Z face of the spacecraft, same as the Camera subsystem. In order to establish communication with ground stations, precise pointing using the iADCS system needs to be employed.

Software-Defined Radio (SDR): The SDR on OPS-SAT is an experimental radio receiver based on Lime Microsystems LMS6002D chip, capable of processing a wide range of frequencies from 300 MHz to 3 GHz with an adjustable sampling rate from 750 kHz to 20 MHz. The SDR frontend shares one of the antennas used with the Nanocom subsystems, tuned to the same frequency. This effectively limits the operational range to the UHF 70 cm band at around 400 MHz. The frontend includes an optional low pass filter tuned at 1.575 GHz allowing for filtering out more powerful signals. After the filter, a Low Noise Amplifier (LNA) is placed. It provides a gain of approximately 20 dB for the UHF band. Overall the system achieves -98 dBm sensitivity. The SDR is connected to the SEPP FPGA via a 12-bit parallel bus, allowing it to capture and buffer in-phase and quadrature (IQ) samples of the received Radio Frequency (RF) signal. The samples can be used by the HPS using direct memory access (DMA) [10].

Optical Receiver: The optical receiver on OPS-SAT is part of an experimental payload designed to test secure optical communication in space. It uses a small photon-counting module with a multi-pixel photon counter to detect incoming optical signals, enabling data transmission at rates of up to 2 kbps [10]. This component allows OPS-SAT to explore new methods of communication that are less susceptible to electromagnetic interference and offer higher security.

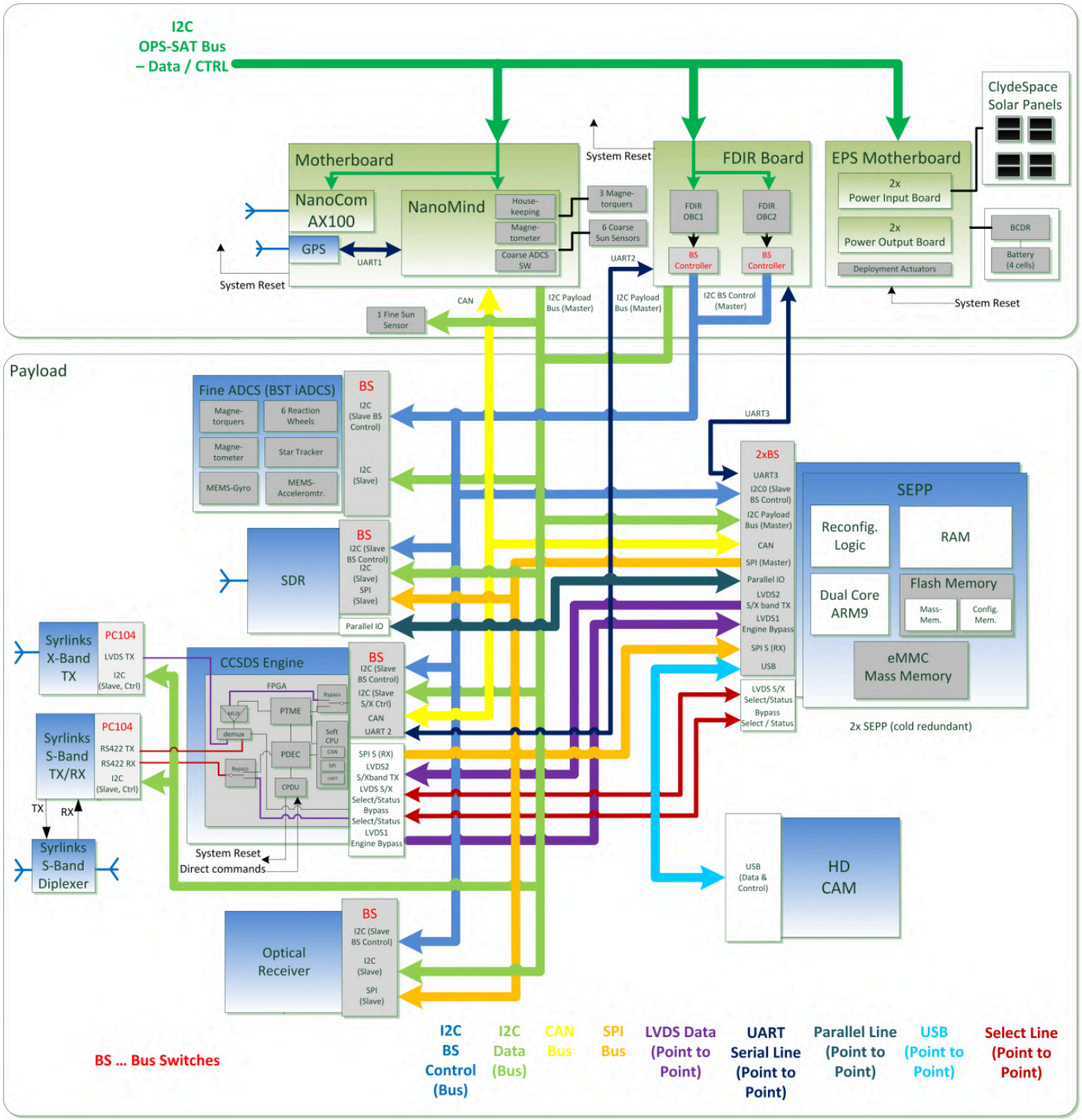

Fig. 6. OPS-SAT System diagram [10].

3.2. Ground Segment

Ground Stations Network: OPS-SAT is supported by a network of ground stations. The spacecraft supports two channels of communication - high bandwidth “Chain-A” via S-Band and low-bandwidth “Chain-B” via UHF. The UHF channel is used for satellite provisioning during LEOP and during recoveries of the satellite platform, especially induced by the FDIR system, like EPS Watchdog resets. During nominal operations the S/X-band channel is used, providing high-speed data uplink and downlink, crucial for experimental data transfers and software updates. The primary ground station location is Darmstadt, Germany, but an additional UHF antenna is placed at Graz University of Technology in Austria.

Fig. 7: ESOC-1 (S/X-Band) and ESOC-2 (UHF) ground stations at ESOC, Author for scale.

Special Mission Infrastructure Laboratory Environment (SMILE): The majority of Ground Segment is located at ESOC's SMILE Lab. It hosts the back-end [14] of the ESOC-1 and ESOC-2 antenna systems, as well as a virtualized environment running the operational software. Thanks to this, the operations can be handled remotely, without the necessity of actually attending the room.

Fig. 8: Panorama of the SMILE control room at ESOC [8].

Engineering Model (EM): The Engineering Model, also known as FlatSat, is a replica of most of the flight hardware of OPS-SAT. It is used extensively for testing, troubleshooting, and validating software and operational procedures before they are used on the satellite in orbit. The Engineering Model allows engineers to simulate various scenarios, test new software, and experiment with different configurations in a controlled environment. This ensures that potential issues can be identified and resolved on ground, reducing the risk of failures in space. The model reflects the current state of the spacecraft, including anomalies, which have been replicated while developing working solutions to be deployed in space.

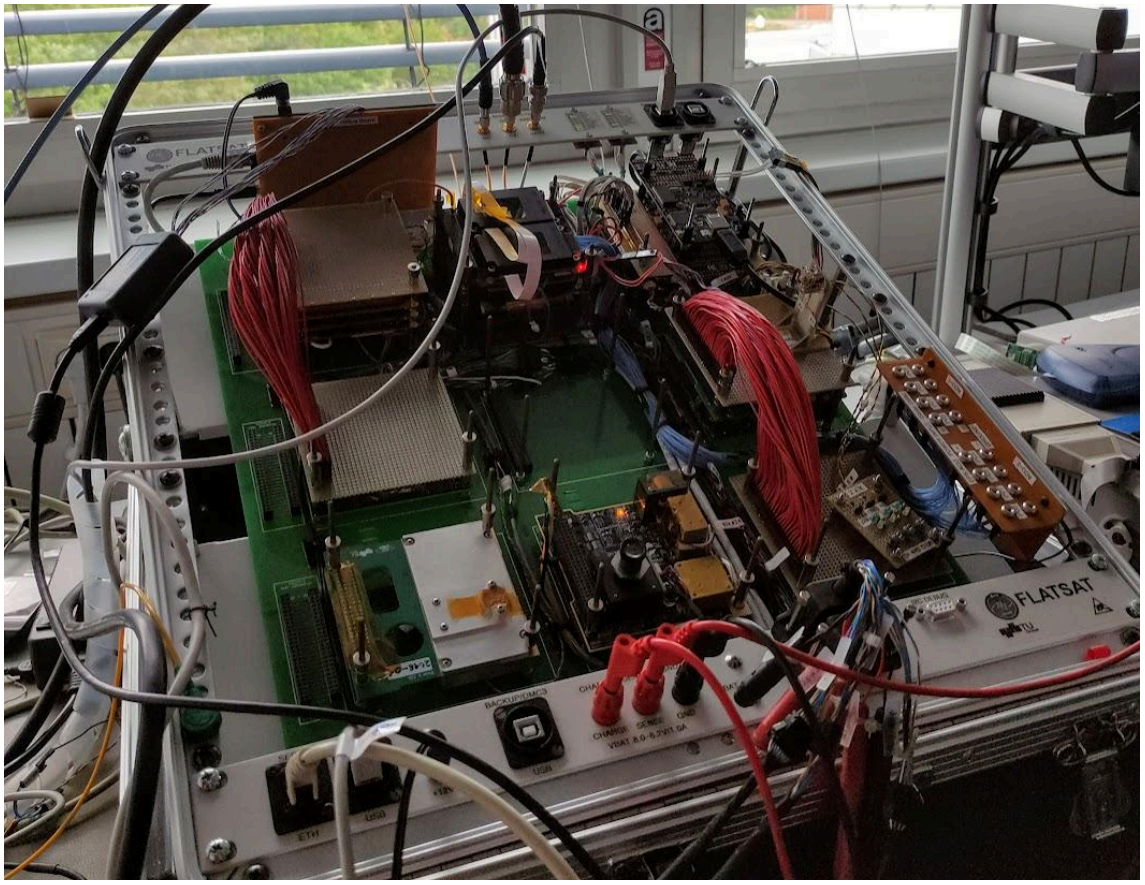

Fig. 9. OPS-SAT Engineering Model.

4. MISSION CONTROL TEAM – CHARACTERIZATION OF OPERATIONS

This chapter covers the learning curve of mission operations from the perspective of a member of the Mission Control Team. Additionally it provides the context of activities to be performed while executing experiments on the OPS-SAT Space Laboratory Platform, which is usually hidden from the experimenter perspective.

The satellite was launched into a 515 km polar, Dawn-Dusk Sun-Synchronous Orbit (SSO) on December 18th, 2019 [15]. Due to the selected orbit, the satellite overlies any point above earth at the approximate time of the local sunset and sundown, making one revolution around Earth about each 90 minutes. In the case of Darmstadt, the typical morning pass window occurs between 4:00 and 7:00 UTC, while evening passes range from 17:00 to 20:00 UTC. Each window, about 2-3 passes are made. To account for seasonal variation in local time, Universal Coordinated Time (UTC) is used for operations.

These times do not fit into the traditional notion of a work day. The passes usually start in the early morning, before a typical workday starts and the evening passes occur just after everyone usually goes home. As the Mission Control Team (MCT) only works in one shift, this highly influences how automation is used to maintain the mission.

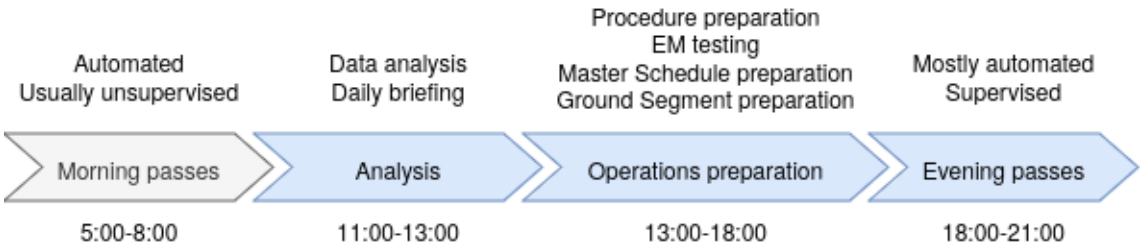

Fig. 10: Workday structure, times in local time.

A typical day for MCT members begins at 11:00 (local time) with a daily briefing, where the team reviews the previous day's operations, analyzes logs from morning passes, and determines the day's objectives. The responsibilities are shared within the team, with tasks changing daily depending on the needs of OPS-SAT. The remainder of the day is spent maintaining the mission, and working with outside collaborators to prepare their experiments for deployment in orbit.

Every 2-3 days a full master schedule needs to be prepared and loaded onto the spacecraft. The schedule includes all the experiments to be run in the given period and all MCT-specific procedures needed to maintain the mission, for example: periodic resets of the subsystems, resets of the hardware watchdogs and enforcing good attitude of the spacecraft. When the satellite is not within communication range, procedures and experiments are tested using the Engineering Model, ensuring readiness for actual operations.

The preparations last until the time of the evening pass window. The exact time, elevation and duration and number of the passes varies from day to day, and not all passes are deemed useful. A good pass needs to be well above 5 degrees of elevation above the horizon. Below this threshold communication using the S-Band link is blocked, as it might interfere with other ground systems, especially that it works adjacent to the terrestrial 2.4 GHz Industry Scientific and Medical (ISM)

networks. The higher the elevation of the pass is, the longer is the useful time of transmission with the satellite.

To attend the passes, the MCT can use their personal workstations to remotely connect to their respective virtual consoles at ESOC. Thanks to this, the passes can be overseen either from office or home. However, this flexibility also means that operators often need to remain vigilant into the night, occasionally revisiting and adjusting work done earlier in the day. The intensity of operations sometimes necessitates attending early morning passes in person, particularly during recovery efforts or when automation fails, although this is left to the team members' discretion.

The moment first telemetry from the spacecraft is received, is a true test of all the assumptions which have been the basis of work throughout the day. If the telemetry is nominal, all the prepared tasks and schedules can be uplinked to space. However, if anything goes wrong, it opens a 90-minute window to prepare any required corrective actions before the spacecraft loops around the Earth, and flies over Darmstadt again.

Over its four years in orbit, OPS-SAT has developed specific operational characteristics [9], some of which have evolved due to the satellite's gradual degradation, while others have improved thanks to the development of new tools that facilitate a variety of repetitive procedures [16]. This is especially apparent during rare instances when the satellite enters a “safe mode” enforced by its FDIR system, disabling most features and requiring a full system reinitialization from the ground up. What once took weeks at the start of the project, such as commissioning during the LEOP, now requires just a few passes, thanks to these incremental improvements. Other improvements allow for quicker OBC firmware updates, which previously needed multiple passes to uplink via UHF directly to the OBC. Nowadays they can be uploaded to the SEPP via high-bandwidth S-Band channel and deployed directly from within the spacecraft, highlighting the mission's ongoing quality-of-life improvements.

4.1. Mission automation tools

4.1.1. Satellite Control and Operation System 2000

The mission automation framework for OPS-SAT is structured into three distinct layers. At the foundational level, OPS-SAT employs ESA's standard mission control software, SCOS-2000 [17]. This software operates within a virtualized environment based on a Linux OS, reflecting its legacy from ESA's 20th-century mission control systems working on SUN Workstations [2]. Despite its age, SCOS-2000 remains a reliable and well-established component of ESA's infrastructure and is mandated for use across ESA missions due to its dependability.

SCOS-2000 facilitates CCSDS packet-based communication between the ground segment and the pre-programmed routines on the On-Board Computer (OBC) of OPS-SAT. The software defines a list of commands which can be organized into a Manual Stack and sent to the spacecraft during a pass. The commands are executed right away, or can be time-tagged and inserted into the spacecraft's Master Schedule for later execution. The Manual Stack can be prepared manually or loaded from a template file, containing a list of commands for a given procedure. The system is also responsible for receiving telemetry from the spacecraft and keeping history of all commands sent to the spacecraft for future reference.

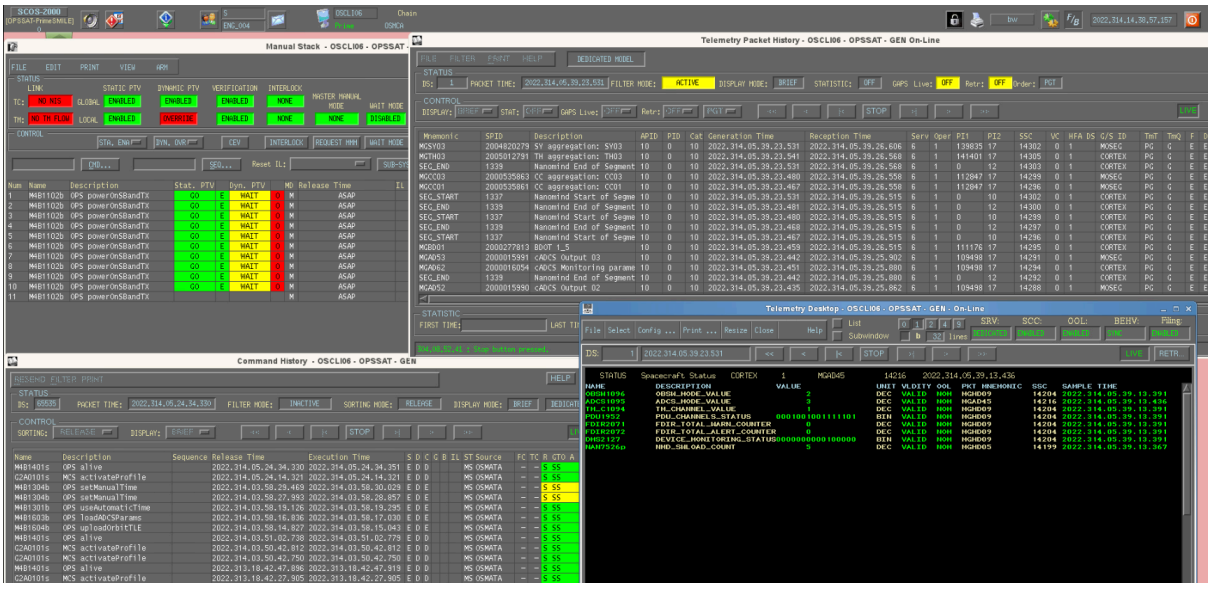

Fig. 11: View of the SCOS-200 interface of the OPS-SAT mission; Manual Stack, Telemetry & Command History windows are visible (Source: ESA/Own archive).

4.1.2. *MATIS - Mission Automation System*

To accommodate the increasing complexity of missions, ESA introduced the Mission Automation System (MATIS), which supports automated operations and minimizes the need for manual oversight. MATIS, integrated with the Service Management Framework (SMF) [17], enables seamless control over both space and ground segments, significantly enhancing operational efficiency and reliability. MATIS is built on the interface provided by SCOS-2000 and allows for the scripted and conditional execution of commands. This approach draws directly from the manual procedures used by operators in earlier years, where PLUTO scripts can emulate the logical steps a human operator would follow when executing paper-based procedures. The system allows for implementing new procedures based on the available commands implemented on lower levels, without need to change software on board the spacecraft. This allows for greater flexibility in operations.

For OPS-SAT, these procedures typically include system-level tasks such as enabling or disabling subsystems, altering subsystem operational modes, changing the state machine of the OBC, and executing custom commands on both the OBC and the Satellite Experimental Processing Platform (SEPP) [18]. The procedures can also be pre-conditioned and generate a list of commands to be inserted into the Master Schedule, allowing for mission control outside of direct communication windows during a satellite pass.

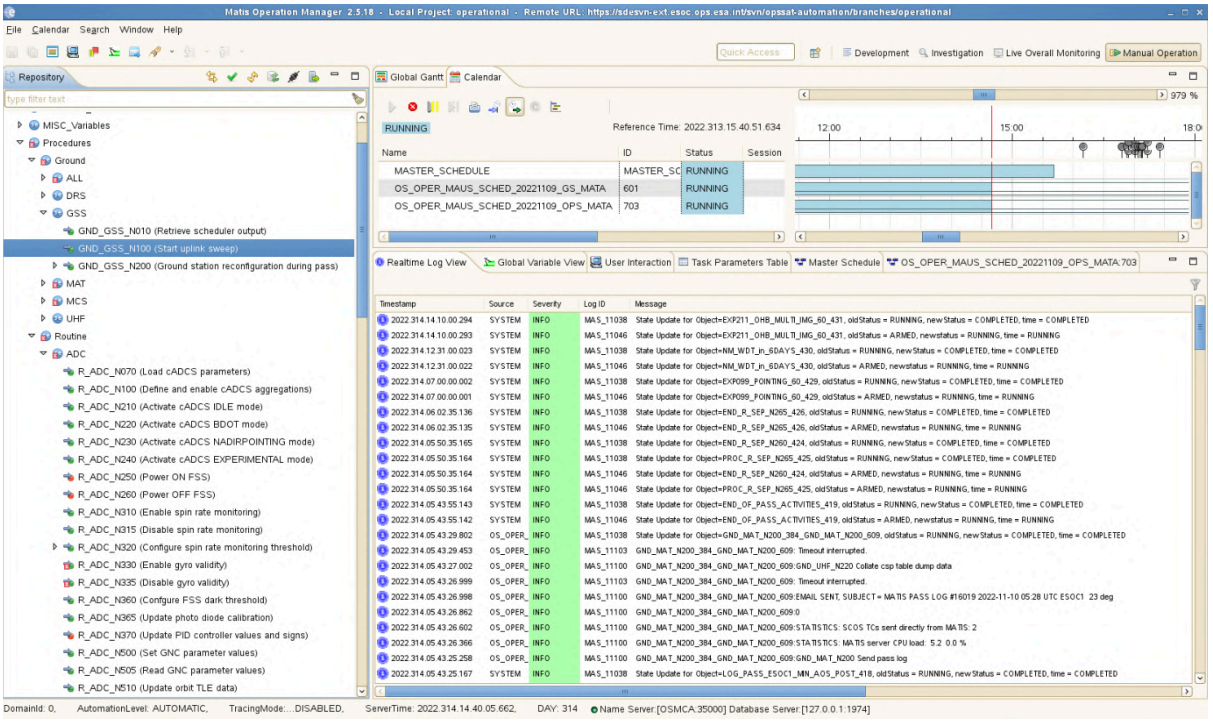

Fig. 12: The interface of MATIS: Operations Repository on the left, Mission Calendar and communication log on the right.

4.1.3. *Mission specific tools*

The most recent layer of automation has been developed by the MCT to further streamline the execution of custom procedures needed for software experiments provided by external collaborators. This layer consists of a combination of Python scripts and Visual Basic macros with Microsoft Excel serving as the user interface. The solution allows for scheduling of MATIS-defined procedures on the highest abstraction level [18]. The tool is effectively used for creation of the Master Schedule of all activities on board, including:

- Daily reset of software watchdogs. In the case the schedule fails to upload, the satellite automatically returns to a safe state.
- Enforcing satellite attitude by commanding iADCS subsystem.
- Execution of custom experiments, together with reconfiguration of the payload subsystem - enabling and disabling various subsystems.
- Scheduling data downlink and experiment data post-processing.

The Excel files also serve as a repository of often used procedures and custom procedures needed to execute experiments. New sets of commands can be easily prepared and tested using the EM, lowering the overhead of operations. Previous files are kept as a log of all operations ever done on the spacecraft. The output of the Scheduling Excel is an XML Mission Calendar file which can be loaded into MATIS. MATIS resolves all the procedure calls and prepares a list of SCOS commands to be placed into OPS-SAT OBC Master Schedule.

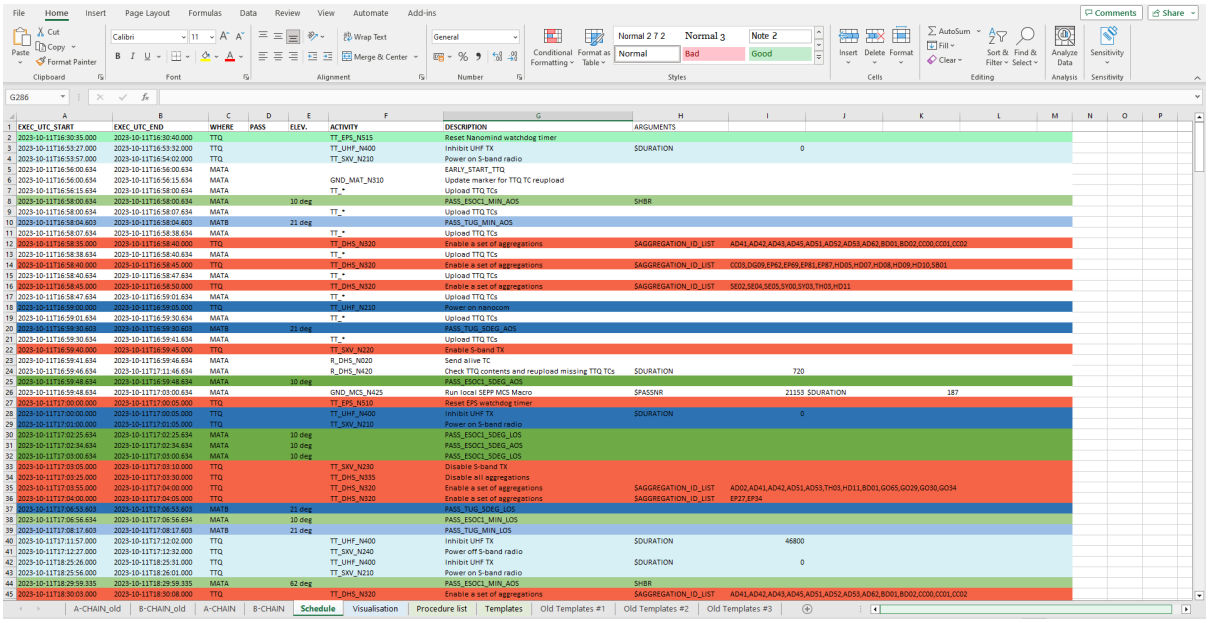

Fig. 13: View of the Excel file representing OPS-SAT Master Schedule. Each line is effectively a call to a Time-Tagged MATIS procedure.

5. CHALLENGES AND OPPORTUNITIES

Throughout the lifetime of the mission many challenges have been worked through to keep the mission running. The timeline of events and corrective actions applied is best described by Max Henkel in their Doctoral Thesis [19]. Developed solutions not only mitigated the issues, but in many ways pushed the mission forward, enabling new unique capabilities to the MCT.

5.1. SpaceWire implementation

In the original design of OPS-SAT, the Controller Area Network (CAN) Bus was used to communicate with the SEPP and other subsystems. Due to its specification and additional overheads, the effective speed of the Bus was limited to 500kHz. This speed was acceptable for the early phase of mission commissioning, but faster speeds and flexibility would be required to make use of the powerful hardware on board.

In anticipation of this, before launch, a raw Low-Voltage Differential Signaling (LVDS) bus was connected between the CCSDS Engine and the FPGA fabric. At the moment of launch no protocol was implemented on top of it. After launch FPGA firmware was developed, deployed and validated in space, implementing Space Wire protocol between the CCSDS Engine and the SEPP, via the FPGA fabric. By encapsulation of the relevant packets, this enabled high-speed Ethernet-like connection all the way to the ground segment via CCSDS-compatible S-Band link, which enabled TCP/IP connection to be opened on top of it. This in turn allowed for standard Linux OS network management tools to be used to remotely manage the SEPP, making full use of the connection speed.

In the end a variety of UNIX-style programs like "*ssh", "rsync"* and *"scp"* could be used to copy files and execute commands on board. Together with the flexibility of SEPP and the FPGA, the entire spacecraft could be managed from within by uploading software patches and experiments to the SEPP over S-Band chain. One of the operational tools taking advantage of this connection is Automax. This automation program allows for scheduling of bash scripts to be run before, during and after each pass, both on ground segment and directly on the SEPP.

Since Space Wire was implemented, the nominal state of the SEPP includes what is known as "standard image" loaded to the FPGA. The caveat is, that any experimentation using the FPGA necessitates a reboot in order to replace this image. Moreover, if any functionalities need to be added to the "standard image" they need to be integrated in such a way, as to not break the SpaceWire implementation.

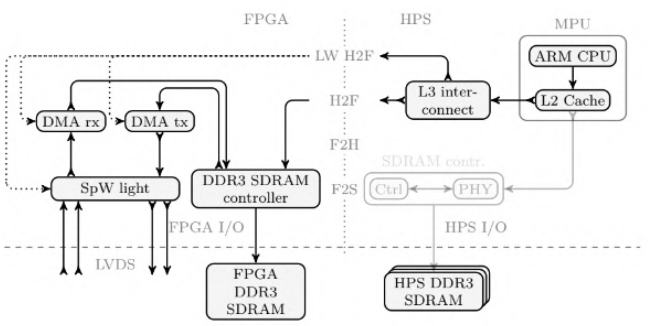

Fig. 14: Space Wire implementation diagram [19].

5.2. Gradual degradation of SEPP

Before launch, the OPS-SAT team discovered that the SEPP-1 unit exhibited intermittent boot failures or would sometimes hang during operation. Further investigation revealed a communication problem with one of the three memory chips making up the HPS RAM. This issue prompted the team to designate the SEPP-2 unit as the primary processing platform while developing a workaround plan for SEPP-1 in case of a SEPP-2 failure in orbit [9].

Approximately one year into the mission, the SEPP-2 unit experienced a critical failure, preventing it from booting up completely. Telemetry data indicated insufficient power draw by the unit, suggesting a failure to exit a reset state. This failure occurred during eclipse season, meaning lower power provided by the EPS to the subsystems. The operational concept at the moment involved powering the SEPP on and off multiple times per day (4–6 times) in order to save electrical power, potentially intensifying the issue due to thermal cycling [20]. With SEPP-2 inoperable, the team activated the workaround plan for SEPP-1, which involved circumventing the problematic HPS RAM and utilizing the FPGA RAM instead. While this restored operational capability, it came with a significant reduction in available RAM for applications, effectively halving it from 1 GB to 512 MB.

Throughout the year 2022, in the third year of operations for OPS-SAT, memory issues related to the SEPP-2 eMMC memory started to occur [21]. The used Flash Memory is not radiation-hardened, and most likely started to see degradation due to the elevated radiation levels seen in space. Moreover, the SEPP has been extensively used to run experiments, compounding the wear and tear of the memory. While these issues could initially be solved by rebuilding the file system, they eventually became so severe that the eMMC memory would enter a failure state and become unresponsive. Effectively in June 2022 the corruption made it impossible to communicate with SEPP-2.

To mitigate this issue, another way of communication had to be established between the satellite bus (the Nanomind OBC) and the SEPP via the Universal Asynchronous Receiver / Transmitter (UART) interface [21]. The implemented method of communication allowed the MCT for communication with the experiment platform starting at boot time. The OBC recorded the serial output of the SEP while booting up, which would later be downloaded to ground during a pass. After reviewing the log, a procedure could be implemented which power-cycled the SEPP and executed appropriate commands during boot by sending key-strokes and binary data via UART. With access to the SEPP bootloader, the boot parameters could be changed, and a partial read-only approach could be implemented to limit further corruption of the eMMC.

After a further boot failure of the SEPP in December 2022, a fully volatile concept for system operation has been implemented [21]. The solution utilized the system booting to a virtual partition placed in the RAM (ramfs) instead of the eMMC. The eMMC could be read, but any write operations were directed to the volatile memory, effectively creating an overlay over the memory (Overlay File System). Due to the nature of RAM it meant that all working data saved in this memory was deleted after each reboot.

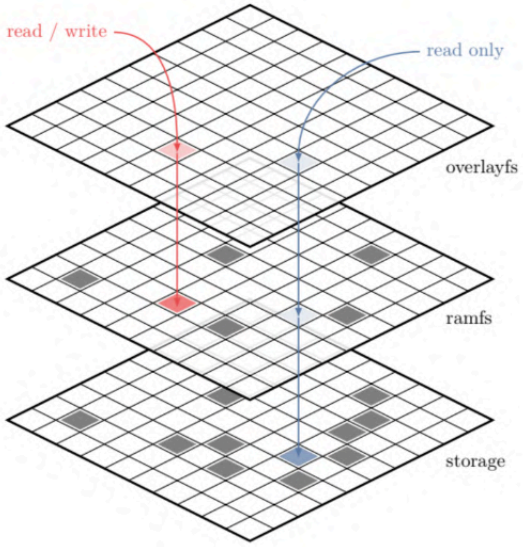

Fig. 15: Overlay File System concept implemented on OPS-SAT [21].

Further debugging showed that not all blocks of the eMMC memory were corrupted. In order to allow for persistence between reboots, the known good parts of the memory have been used to store data between reboots. The physical eMMC module could still be mounted as a bulk device in the Linux filesystem, but due to the corruption, no filesystem could be used on top of it. In order to make use of the available memory, a strategy involving direct binary read-writes to the eMMC has been developed. The testing showed that the memory could be accessed using programs like “dd” [22]. The MCT created a manual map of memory addresses which could be read and written. To add structure to the data, the Tape Archive (TAR) [23] format headers have been used to append metadata to the saved binary data. Thanks to this, the data could be easily retrieved from memory using standard tools.

```
# Part of the eMMC is mounted as a partition
> blkpg-part add /dev/mmcblk0 180 7004487680 192937984
# File is written to the partition using the tar command;
> tar cvf /dev/mmcblk0p180 $file_to_write
# Even without mounting the partition, the file is accessible under the memory
addresses:
> dd if=/dev/mmcblk0 bs=512 skip=13680640 count=376832 | tar xv -C /tmp
```

List. 1: Sample of Bash commands used for eMMC data storage and retrieval.

With the ability to persist data over reboots, the SEPP became fully operational again and has served dozens of experiments since. Many operational procedures have been developed to make working around those limitations manageable.

In the last year of operations of OPS-SAT, additional issues related to one of the CPU cores of the HPS started to occur. It resulted in one of the cores being disabled, and subsequent one-core operations. All in all, this further limited the processing power available to experiments, together with the limited RAM and persistent memory issues.

5.3. Prior experiments utilizing the SDR

One of the first experiments utilizing the SDR subsystem was an In Orbit Demonstration (IOD) conducted by Tom Mladenov et al. [24] The experiment focused on detecting and decoding signals from COSPAS-SARSAT beacons – specifically Emergency Position-Indicating Radio Beacons (EPIRBs) used by ships and Emergency Location Transmitters (ELTs) used in aviation. Those systems transmit distress signals on 406 MHz frequency well within the operational range of the SDR.

The experiment captured data from the SDR subsystem in bursts, reading IQ samples from the FPGA's RAM and transferring them to the SEPP's HPS portion using DMA. There, a custom application developed in GNU Radio, an open-source software development toolkit, was used to process the received RF samples from the SDR. It provided the flexibility to implement the complex signal processing algorithms required for beacon demodulation. It performs tasks such as:

- Signal detection using the characteristic preamble sequence of COSPAS-SARSAT beacons.
- Synchronization with the beacon's timing.
- Demodulation of the signal to extract the encoded data bits.
- Decoding of the data to retrieve the beacon's message, including identification and location information.

After decoding, the GNU Radio application generated JSON metadata files containing essential information extracted from the beacon signals, such as the encoded message, burst frequency, and any detected errors. Both the JSON metadata files and the decimated IQ samples of the detected bursts were stored on the SEPP's eMMC permanent storage for later downlink.

The experiment had to work around the limitation of the DMA interface with the SDR, having to capture and process data in bursts. With the ability to process data on board, it could capture a batch of samples, process them, delete, and move on to another batch, allowing for almost continuous operation until the space for experiment results was used up.

Overall the experiment was a show of capabilities of the SDR and the entire OPS-SAT system. It also highlighted avenues for improvement, hinting at improvements around the interface with the SDR.

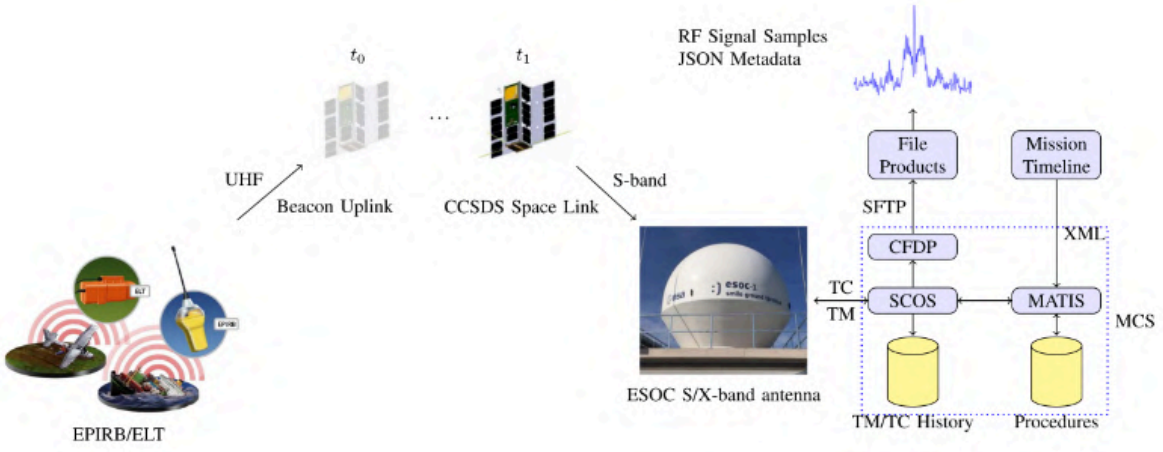

Fig. 16: Concepts of operations of the SAR Beacon detection experiment [24].

5.4. Libre Space Foundation SIDLOC experiment

Spacecraft Identification and Localization (SIDLOC) is a telecommunications scheme developed by the Libre Space Foundation and funded by the European Space Agency (ESA). Its primary purpose is to enable the identification and localization of spacecraft, particularly during the crucial LEOP operations, utilizing low-power RF beacons operating in the UHF band [25].

Because the OPS-SAT SDR subsystem doesn't support transmission, a "reverse experiment" was conducted as part of the SIDLOC project. This involved ground stations transmitting signals to the satellite to assess the impact of radio channel impairments and spectrum occupancy at the target frequency. While implementing the decoder part of the experiment on board OPS-SAT, the team encountered the same challenge with the non-continuous SDR output due to the DMA interface. To enable continuous IQ data streaming for future experiments like SIDLOC, the Libre Space Foundation team introduced a FPGA implementation accommodating continuous data streaming. This involved incorporating a Scatter-Gather DMA engine and additional buffers to manage the continuous data flow from the SDR to the SEPP. The use of Scatter-Gather topology allowed for using larger memory buffers between systems as opposed to standard DMA, limited in size by shared memory space between platforms. A dedicated user-space driver was created to interact with the modified FPGA image and facilitate the retrieval of the continuous IQ stream by applications running on the SEPP.

Despite the implementation of a streaming interface, the SIDLOC experiment encountered another issue. The software for decoding the packets was implemented in GNU Radio, which is a rather computationally heavy framework. Since the previous experiment utilizing it [24], further SEPP degradation occurred, rendering one of the SEPP CPU cores unavailable. Testing has shown that due to the limited CPU resources available, GNU Radio was no longer viable, especially for real time applications. The experiment instead opted to store the continuous IQ data stream on the satellite's onboard storage and later downlink it for post-processing on the ground.

Fig. 17: Libre Space Foundation manifesto for open space exploration.

6. UNDERSTANDING OPERATIONS

This chapter covers the chronological learning curve of various types of experiments executed on board OPS-SAT Space Laboratory and showcases the different use cases for the operational tools.

6.1. Chess experiment

Developed by fellow interns - Nuno Carvalho and Stefanos Digenis - chess-ops.space [26] experiment is the simplest, yet the most representative of OPS-SAT unique capabilities.

The Chess-OPS homepage shows a chess board on which users can vote for the next move, starting with white figures. Each day, just before the evening passes of OPS-SAT over Darmstadt, the most popular move is selected. When the satellite comes into view of the ground station, and connection via S-Band is established, the ground segment (Automax) copies a text file containing the selected move to the SEPP.

Fig. 18: Chess-OPS Website.

On board the spacecraft. The experiment installs a simple chess engine - Stockfish [27]. The chess engine is triggered, and returns another file with the response move. The file gets downlinked with other experiment results. The entire process takes just a few seconds, depending on the difficulty settings of the chess engine. The result is then displayed on the website, giving a chance for players to respond.

Despite its apparent simplicity, the experiment serves as a vital health-check of the satellite. Any failure of the experiment could be traced to underlying issues with the satellite. Some failure modes include:

- Experiment executed successfully but no move was downlinked - connection could be interrupted due to satellite rotation or line of sight obstruction of the S-Band antenna beam.
- Connection is established, but no experiment is installed on board - it means that there has been a SEPP reboot since the last pass.

- Connection timed despite good signal strength - it could mean that the FPGA Space Wire stack is not loaded, or the SEPP is off, indicating the spacecraft is in Safe Mode.

Quick review of the experiment log could give a good estimate for the health of a given satellite pass. Thanks to its usefulness and small size, the experiment was one of few packages stored in the persistent eMMC memory and could be easily reinstalled after each SEPP reboot.

Over 6 months, 3 correspondence games with the satellite in orbit have been completed in an ultimate chess match between Humanity and OPS-SAT. The tally of the game was 1:1, and the third game started at the beginning of 2024. The match was never finished before the satellite deorbit in May 2024. Thus, we might never know who the grandmaster of chess is - humankind hive-mind, or the certainty of satellite.

6.2. Taking pictures and attitude modes

Another end-to-end example of OPS-SAT operations is the case of picture taking. It involves the use of two subsystems - Camera and iADCS, and requires precise scheduling of a sequence of events to achieve good results.

After 4 years in orbit, the satellite has already decayed and lost several dozen kilometers of altitude lowering from its original 550 km spherical orbit. The process is only speeding up, as the spacecraft enters the denser parts of Earth's atmosphere. Any changes to the drag coefficient are quite visible in the orbital elements. Due to this, during nominal operations, the satellite is kept in so-called “horizontal-pointing” attitude, maintaining parallel orientation of the solar panels with the ground and pointing the camera-end of the satellite along the vector of velocity. It is done in order to lower the drag of the satellite on the oncoming atmosphere, but makes the camera pointed away from Earth.

The ADCS subsystem supports the following modes [10]:

- B-Dot Detumbling - stabilize the attitude against earth's magnetic field, using only the magnetorquers. The reaction wheels are off.
- Sun Pointing - follow the sun as detected by sun-sensors, using magnetorquers and a subset of reaction wheels.
- Nadir Pointing - point the -Z axis of the satellite to Earth using reaction wheels.
- Target Pointing - quaternion or earth coordinate based tracking using reaction wheels.

One can notice that horizontal pointing is not among the supported modes. This attitude mode has been implemented as one of MCT-developed SEPP experiments, also known as HoPo (Horizontal Pointing). The experiment moves the control of OPS-SAT attitude to space, by periodically toggling between the B-dot Detumbling and Target Pointing modes. In the Target Pointing mode, the experiment every few seconds calculates the position of the vector parallel to the vector of velocity and updates the iADCS target. After approximately 1.5h the reaction wheels are assumed to be saturated, meaning that they rotate at the maximum designed speed and no longer can control the spacecraft orientation. The experiment then changes the iADCS mode to B-Dot Detumbling for 30

minutes. At first this causes the spacecraft to start rotating, as the momentum is transferred from reaction wheels to the spacecraft. It then gradually de-tumbles against Earth's magnetic field using magnetorquers.

To take a picture using OPS-SAT, we need to first turn off this experiment and wait for the B-dot detumbling to be complete. Then, we can enable the Nadir-pointing iADCS mode, and after about 10 minutes we can trigger the camera to take a picture. Due to this procedure, the spacecraft briefly enters high-drag configuration.

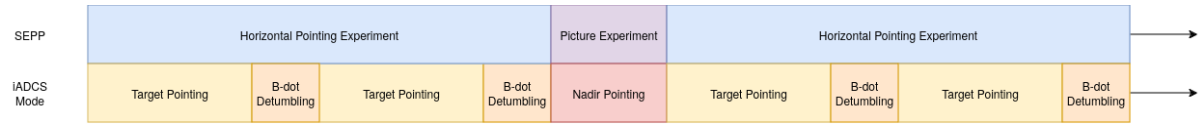

Fig. 19: Approximate timeline of iADCS and SEPP modes during the picture-taking sequence.

Scheduling a picture sequence requires the use of all tools in MCT disposal. First, the exact time of the picture needs to be established. Then, a Master Schedule template can be selected in the scheduling Excel, providing a baseline for the following steps to be executed:

- At T-40 minutes, send a stop signal to the Horizontal Pointing experiment, making it start detumbling.
- At T-10 minutes, kill the HoPo experiment and enable iADCS Nadir pointing mode.
- At T-0 the camera experiment is triggered.
- At T+5 minutes power cycle iADCS.
- At T+10 minutes start the HoPo experiment again.

The time of each event is automatically calculated by Excel macros, relative to the selected T-0 time. Once the entire Master Schedule is prepared, it can be placed in MATIS, awaiting upload to the spacecraft.

Control of the camera parameters like exposure, is conducted via a separate config file. Some of the available configuration parameters include:

- Exposure time of capture in milliseconds.
- Number of pictures to be taken in sequence.
- Delay between each picture in sequence.

The config file can be modified in two ways. First, additional commands can be added to the Master Schedule, making the OBC send UART commands to the SEPP including appropriate Linux commands, modifying the experiment config. Programs like “sed” [28] can be used to modify the content of the configuration file already present on board the spacecraft. Another approach involves the use of the Automax tool to sync the entire configuration file from the Ground Segment to the SEPP via the Space Wire connection.

Once the sequence is executed in space, the camera experiment automatically generates thumbnail preview pictures to be downlinked. During the next pass, the previews are fetched using the Automax tool. Once reviewed by MCT, the full resolution pictures can be scheduled to downlink in subsequent passes.

To improve this operational loop, SmartCam [29] experiment has been introduced. It is a drop-in replacement for the camera experiment software package. It supports a similar configuration and scheduling procedure, but improves it by including a pre-trained Machine Learning (ML) model to detect features in the captured pictures. The experiment automatically categorizes pictures in the following categories, with each category being assigned a percentage of certainty:

- Earth - Category trained on pictures including landmass and otherwise esthetically pleasing.
- Bad - Category trained on known bad examples of pictures - water mass, overcast, over- or under exposed.
- Edge - Category trained on pictures including Earth's curvature, pointing off-Nadir angle.

SmartCam can be configured to keep or reject the pictures based on the assigned category and threshold. Thanks to this, the pictures can be downlinked directly to ground, lowering the load on MCT. Additionally, the experiment supports a variety of Open Source ML models, allowing for different categorizations and actions. Other experiments can extend this functionality in unexpected ways, as shown by the OPS-SAT DOOM project, utilizing the SmartCam picture output as a backdrop to a Doom game executed in space [30].

Overall, operations of the camera subsystem require complex operations over the entire spacecraft with regard to orbital mechanics and the environment of space. All operational tools available to the MCT are used, and new approaches have been developed to make operations easier. The two camera experiments have been used over the years for science, training and a show of capabilities of OPS-SAT. They resulted in many beautiful captures [31], selection of which can be seen below.

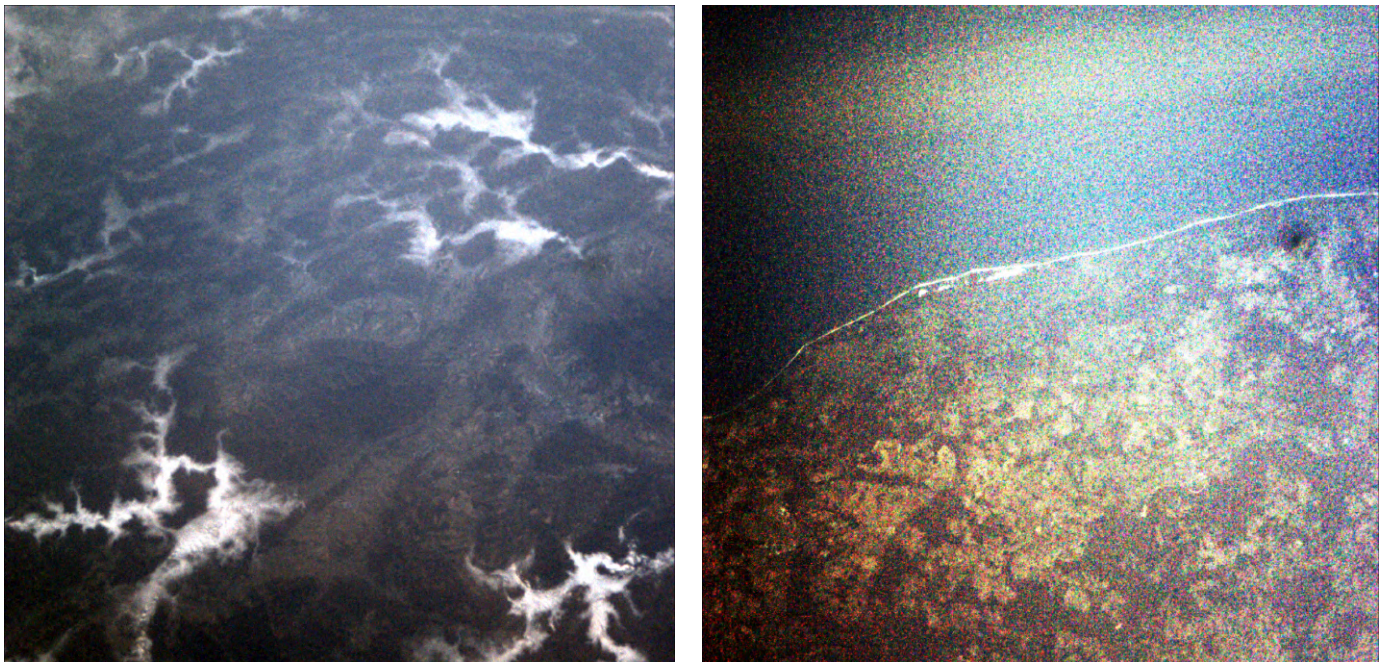

Fig. 20: Left: Early morning fogs in the valleys of Tatry mountains, Poland.
Right: Polish Baltic sea coast. Contrast adjusted. [8]

Fig. 21: Pictures of New Zealand over Hakataramea Valley (left) and Mount Ruapehu (right). [8]

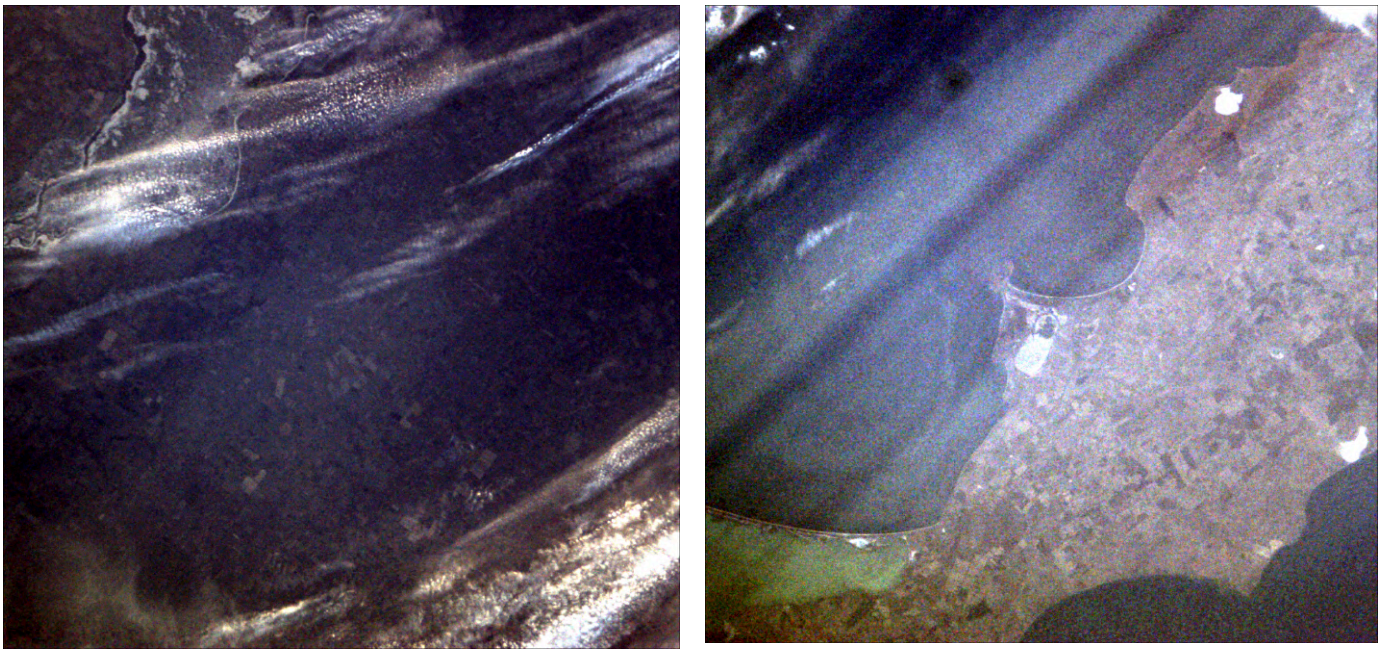

Fig. 22: Left: Picture of former Kakhovka Reservoir, Ukraine.
Right: Part of the Russian-occupied Crimea peninsula. [8]

6.3. *SDR test recording of M17 protocol with SP5WWP*

To gain hands-on knowledge of the SDR subsystem and its operations, a test recording has been arranged. The goal of the test was to assess the current procedures for scheduling, capturing and handling of data generated by SDR experiments. The test used the original “provisioning” experiment software used at the very beginning of the mission to verify the operations of the SDR subsystem. To provide real world context, the test setup involved cooperation with the amateur radio community. Cooperation with operator *Wojciech* SP5WWP from Poland has been established with the aim to test M17 protocol communication between earth and space.

The M17 protocol is a new, open-source digital communication system created for amateur radio users. Unlike many existing digital voice systems that are often closed and proprietary, M17 is completely free and community-driven. It uses a technique called Gaussian Frequency Shift Keying (GFSK) for sending signals and an open-source codec, Codec2, to deliver clear voice quality at low data rates [32]. Besides voice communication, M17 also supports sending text messages and data, making it useful for different kinds of radio communication. It can work with both software-defined radios (SDRs) and regular radio devices with some modifications. Ongoing development aims to add more features, like encryption and better compatibility, making M17 a flexible and accessible option for the amateur radio community.

To find a good observation time of the amateur station, the same approach as with camera experiments has been used. VTS Timeloop [33] software from CNES is used to visualize the future trajectory of OPS-SAT. A perfect pass opportunity has been found, with the satellite making a high elevation pass above the transmission site in Poland. The pass is also visible from Darmstadt, allowing for direct supervision using the S-Band connection.

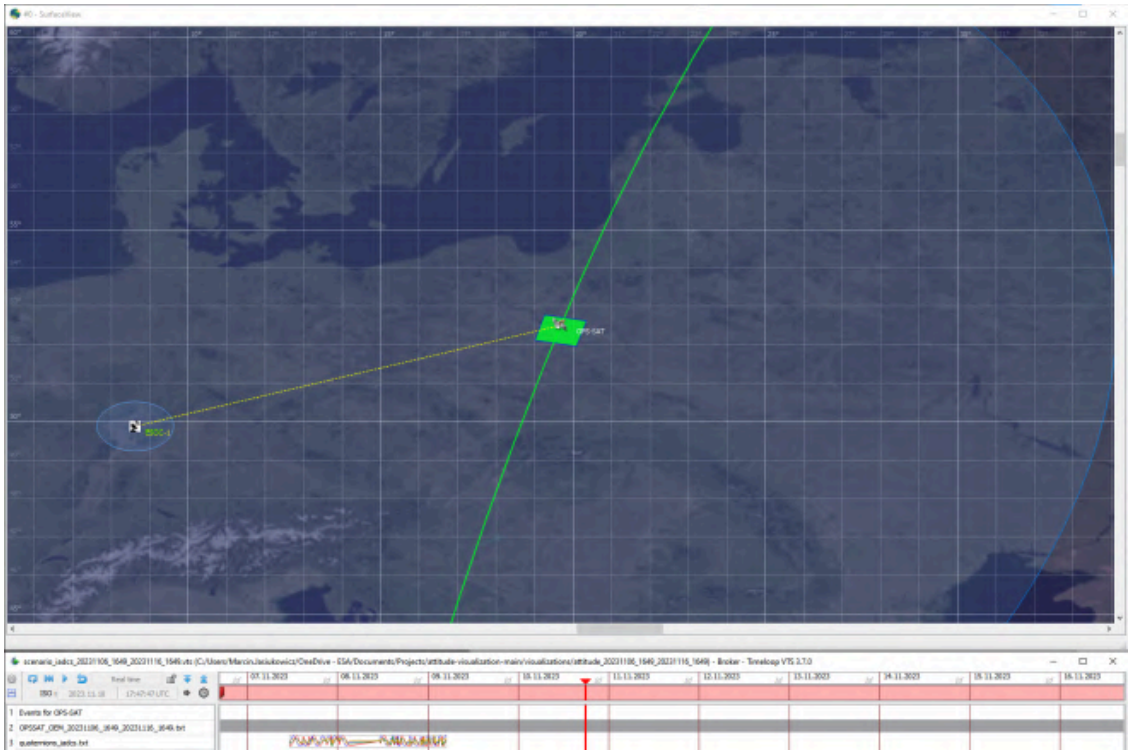

Fig. 23: VTS Timeloop visualization of OPS-SAT trajectory. Moment of pass over SP5WWP station.

On ground, a simple setup using a TYT MD380 handheld radio modified to support M17 protocol and a 7 element Arrow II Yagi antenna was used. To verify that the transmission can reach the spacecraft using this setup, a link budget has been calculated.

The link budget takes into account the transmission power, the gain of the antenna system, the signal attenuation caused by the transmission distance and the gain of the receiver setup. Then, the calculated received power can be compared to the sensitivity of the SDR system to verify whether the signal can be properly received and interpreted. The approximate formula looks like following:

$$P_r = P_t + G_t + G_r - L_p - L_o \quad (1) \tag{1}$$

Where:

$$P_r - \ Power\ received\ [dBm]$$

$$P_t - \ Power\ transmitted\ [dBm]$$

$$G_t - \ Gain\ of\ the\ transmitter\ antenna\ [dBi]$$

$$G_r - \ Gain\ of\ the\ receiver\ antenna\ [dBi]$$

$$L_p - \ \ Free\ space\ loss\ [dB]$$

$$L_o - \ \ Other\ losses\ [dB]$$

Free Space loss can be calculated using formula:

$$L_p = 20log_{10}(d_{km}) \ + \ 20log_{10}(f_{GHz}) \ + \ 92.45 \tag{2}$$

Where:

$$d_{km} \ - \ Distance\ between\ transmitter\ and\ receiver\ [km]$$

$$f_{GHz} - \ frequency\ of\ the\ transmission\ [GHz]$$

Additionally, the gain of the Yagi antenna has been calculated using a formula from Karl Rothammel “Antenna Book” [34]. The result of Link Budget calculation is seen in Table 1.

Table 1: Link Budget.

Component	Description	Value	Unit
Transmitter	Transmit Power	34.77	dBm
	Yagi elements	7	-
	Antenna Gain	9.88	dBi
	Cable Loss	0.25	dB
Path	Distance	500000	m
	Frequency	433.475	MHz
	Free Space Loss	139.169	dB
Satellite	Antenna Gain	19.5	dBi
	Cable Loss	0.25	dB
	Receiver sensitivity	-98	dBm
System	Expected Signal Level	-75.519	dBm
	Link Margin	22.481	dBm

The recording was scheduled exactly at 17:47:40 UTC on November 10th, 2023, and lasted 30 seconds. The test happened during a connectivity window with the ESOC-1 station in Darmstadt, and the successful execution of the experiment has been confirmed right away by verification of experiment log files. In the same connectivity window, a script has been executed manually via SSH connection over Space Wire, instructing SEPP to store the captured samples in the eMMC memory. From then on, nominal operations have resumed, and a new Master Schedule has been uploaded to the spacecraft, reconfiguring it for next experiments.

Over the coming days, the entire 100 MB recording has been downlinked. The process lasted about 2 days, requiring a few satellite passes over ESOC-1. The process involved scheduling of the downlink in one of the evening pass windows, with the data downlink lasting through the morning and next day's evening passes. The reception of the full recording was confirmed the next day. As parts of the raw recording became available, the visibility of the transmission could be confirmed in the spectrum.

After the entire recording has been downlinked, decoding of the transmission has been attempted by the amateur radio community. Sadly, although the transmission is visible, the attempts were not successful. This could be caused by additional losses in the Link Budget caused by pointing losses of the antenna on ground, or the characteristic encoding of the M17 protocol itself making it susceptible to distortion or interference from other signals. Additionally, unknown interference has been observed across the spectrum, which might have contributed to this problem. The cause of interference could be caused by aliasing of higher frequency signals, but the root cause has never been identified.

Overall, this was one of the longest recordings attempted on OPS-SAT, made possible by using the lowest available sampling rate. Previously, the same setup has been used to make recordings using higher bandwidth, trying to detect Global Positioning System L1 frequency at 1.575 GHz. The surprising outcome of this test was the detection of many other signals in the 70 cm spectrum. Other unencrypted voice transmissions could be heard at different frequencies in the same spectrum, however they were not powerful enough to be correctly interpreted.

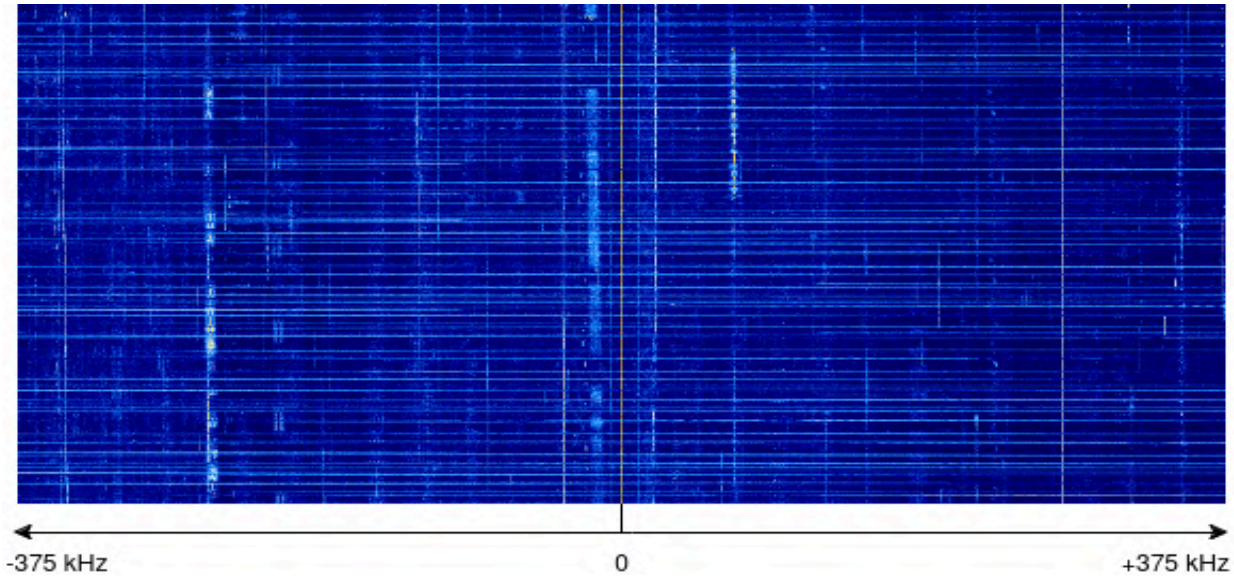

Fig. 24: Waterfall plot of the captured radio spectrum in M17 test.

7. SDR EXPERIMENT IMPLEMENTATION

7.1. Statement of purpose

With the operational landscape of OPS-SAT changing throughout the mission, the SDR subsystem had seen limited use. At first, the SDR operations were limited by the DMA interface, by not being able to process data continuously, despite having computational power to do so. Then, once the streaming interface has been established, the data could not be processed in real time, due to CPU degradation. To make SDR experiments viable again, a coherent set of tools and procedures needs to be established, working around the limitations of the system, and empowering the MCT to operate the subsystem with ease. This becomes ever more important as the lowering of OPS-SAT orbital altitude due to drag makes other experiments, like those making use of the camera subsystem, less viable to perform.

With the streaming capability enabled, but no processing power to make use of it using current methods, another direction needs to be established. Due to the nature of the SDR subsystem, the recordings require a substantial amount of memory to be handled. The current approach to operate the SDR makes use of about 100 MB of freely accessible memory, without jeopardizing stability of the SEPP. Additional steps are taken to store captured data in the eMMC, making it persistent and freeing space for other experiments. Despite the bottlenecks, the underlying 16 GB eMMC chip is still accessible, but requires new approaches to make better use of it. Theoretically, it could enable SDR recordings above 100 MB in size by implementing direct, automatic writes to it. Once the data is stored in the eMMC, any processing abiding by the CPU limitations may be applied. Due to slower processing speed, the processing might be non-real time.

Another limitation is the speed of the S-Band connection with Ground Segment. Despite it being state-of-the-art, the download time of experiment results is substantial, especially at this size. The MCT lacks tools to efficiently manage SDR recordings and resorts to downloading entire raw recordings to ground for later processing. New approach could be proposed to limit the amount of data needed to be downlinked.

Overall, the goal of the experiment is to streamline operations of the SDR platform by:

- Circumventing the limitations of available read-only memory and introducing a way to store data persistently to the eMMC at the moment of recording,
- Suggesting processing methods for captured data, making it easier to handle.
- Providing a way to efficiently downlink recorded signals.
- Applying all knowledge gained in operations to make the experiment operationally viable.

7.2. Working environment

To begin work with the OPS-SAT Space Laboratory, the experiment first needs to be registered with the MCT to gain access to the Experimenter Platform. The Platform allows users to upload source code in appropriate form, which gets automatically formatted into an installable package suitable to deploy on the SEPP. A new SDR experiment has been registered, and received an identification number 266. For simplicity it can be referred to as EXP266.

The submitted experiment source needs to follow a template, to make it easier to handle. Each experiment shall provide:

- start_expXXX.sh (where XXX is the experiment number) - entry point script executed by Mission Automation Tools to launch the experiment at scheduled time.
- stop_expXXX.sh - optional script executing graceful shutdown of the experiment at the end of scheduled execution window.
- config.ini - configuration file with adjustable parameters for the experiment.
- “toGround” folder - place where experiment results will be placed. The folder is synchronized with the analogous “fromSpace” folder on the Experimenter Platform, best to the ability of MCT.
- “fromGround” folder - place where any input files from ground will be placed.

Anything outside of this template is up to the experiment developer. All dependencies need to be included in the package or negotiated with MCT to be installed separately. By default Python 3 and Bash environments are supported. Upon request, Java with CCSDS Mission Operations library [35] support can be made available. The SEPP supports any language that can be compiled to its architecture. Compilation environment based on Yocto toolchain is provided [36]. All binaries required to run the experiment need to be provided, with some shared libraries being available in the Linux OS environment already.

Once a user uploads their software in appropriate format, the Experimenter Platform triggers a pipeline creating an installable file in OPKG format [37]. This file can be used by the MCT to test the experiment on the Engineering Model. Users can schedule debugging sessions with MCT members to validate their experiments and receive feedback for further development. Once the experiment is stable, it can be scheduled to be deployed to the SEPP. The experimenters provide a set of parameters when their software should be run in space, for example a particular location above earth surface. Then, the experiment can be scheduled in the next Master Schedule planning window.

The deployment to space is conducted via the Automax tool. The tool copies the files from OPS-SAT Data Relay System (OSDRS) server over SSH connection to the SEPP and installs it. Once installed on board the satellite, the experiments can be triggered via Master Schedule using mission automation tools. After successful execution in orbit, the results of the experiment are downloaded to OSDRS, again via Automax. The data is forwarded to the user and available on the Experimenter Platform to download, closing the experiment cycle.

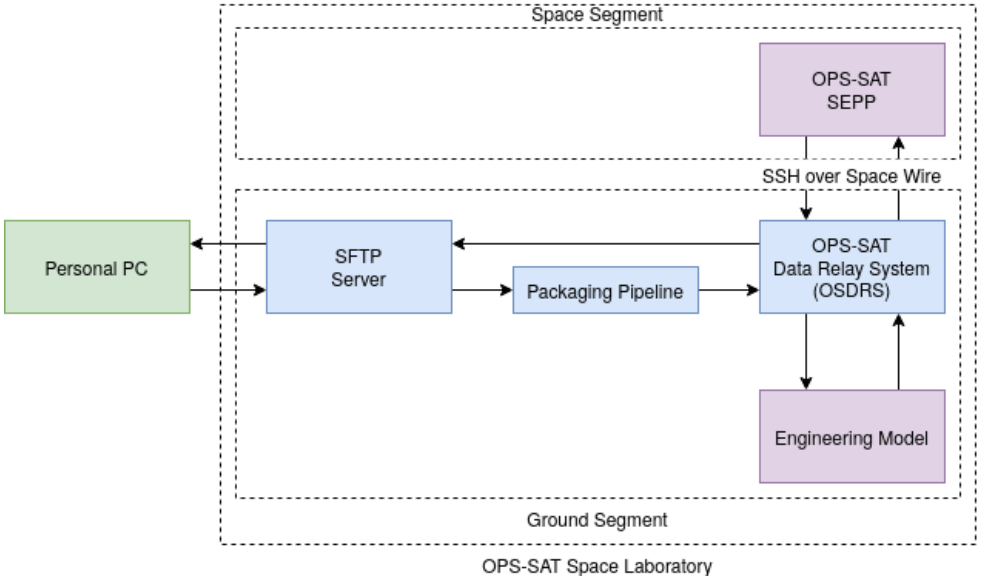

Fig. 25: Approximate system outline of the OPS-SAT Space Laboratory and the Experimenter Platform.

7.3. Implementation Concept

EXP266 is set to make use of recent contributions to the SDR subsystem made by the Libre Space Foundation (LSF) in the Sidloc experiment (EXP202). On top of this development, the experiment shall implement data storage and retrieval from the eMMC memory, best to the current MCT practice.

Out of the scope of this experiment, work is being done on creating a new FPGA “standard image”, merging the Space Wire implementation with SDR streaming interface implementation made by LSF [38]. The new image is developed and tested within EXP223. EXP266 assumes that FPGA firmware provided by either EXP202 or EXP223 is installed and accessible by the SEPP HPS.

On the side of HPS, the experiment shall modify the software driver provided by LSF [39] to support writing directly to eMMC memory. The experiment shall use a pre-made 192 MB eMMC partition designated for storage of experiment results, with option to expand to a new larger partition once the approach is validated. The partition does not offer any file system on top of it, and supports only direct binary reads and writes. The writing process shall store additional metadata of the recording allowing for easier retrieval and handling of the stored data. For this purpose, the TAR format shall be used to make it compatible with current MCT tools and procedures.

In order not to influence other experiments running on the platform, the experiment shall make as small a memory footprint as possible. A constraint of 100 MB of memory is imposed to both the intermittent RAM usage as well as the files stored within the filesystem. This limitation is dictated by the use of Overlay FS in which the operational memory (RAM) is also used as a storage medium. Values above this size have shown SEPP instability causing Out of Memory errors and could cause subsequent reboot of the platform, causing permanent loss of all experiment data.

Although the SDR produces 12 bit samples, each containing the In-phase (I) and quadrature (Q) components of the signal, the implementation of the streaming interface makes them aligned to 16 bit buffers. The data-rate of the samples is equivalent to the sampling rate of the SDR. The lowest SDR clock setting is 1.5 MHz, equivalent to 750 k samples per second (kSps). To adhere to the memory constraint of the SEPP the total recording time is limited to:

$$Total\ Size\ [bits]\ = T_{rec}[s]\ \cdot\ f_s[\tfrac{1}{s}]\ \cdot\ 2\ \cdot\ 16\ bits\ =\ T_{rec}\ \cdot\ f_s\ \cdot\ 32bit \tag{3}$$

$$T_{rec} = \frac{Total\ Size}{f_s \cdot 32\ bits} \tag{4}$$

$$Total\ Size\ =\ 100\ MB,\ f_s = 750\ kSps\ \Leftrightarrow T_{rec} = \frac{100\ MB \cdot 8\ bits}{750\ kSps \cdot 32\ bits}\ =\ 33.(3)\ seconds \tag{5}$$

The use of the streaming interface together with the direct eMMC writes can allow for recordings of bigger size than the constrained operational memory of the SEPP. In this approach the total size of the recording is only limited by the eMMC partition size. The use of additional space expands the operational envelope of the SDR allowing for longer recordings, or higher bandwidth recordings of the same length. With the current size of the partition (196 MB), the possible recording length can be expanded to:

$$Total\ Size\ =\ 196\ MB,\ f_s = 750\ kSps\ \Leftrightarrow T_{rec} = \frac{196\ MB \cdot 8\ bits}{750\ kSps \cdot 32\ bits}\ =\ 65.(3)\ seconds \tag{6}$$

Even with the base size of the recording, handling of the samples is already troublesome, due to long downlink times over the S-Band chain. Moreover, due to the additional size enabled by the eMMC partition, the recordings can no longer be processed directly in SEPP memory using current methods. To allow for easier handling of the longer recordings, a strategy involving lowering the total file size needs to be introduced.

At first, compression algorithms were considered. Due to the way the 12 bit SDR samples are aligned to 16 bits in memory, we can claim back 4 bits of memory per sample, resulting in theoretical 25% memory savings. To implement this, the simplest available compression algorithm - DEFLATE, as implemented by "*gzip*" [40] - can be used. Additionally, the compression algorithm supports streaming, and does not create an intermediary file, helping to abide by the memory constraints. Additional tests have shown the real compression rate of gzip used on the SDR samples to yield 76% of original size. This aligns with the theoretical memory savings by stripping the unused bits, however is the upper bound of possible compression using this method.

Due to the nature of the radio spectrum, even a recording with no detectable signal in it contains noise. Because noise contains random information with no discernible features, it does not compress using available methods which operate on removing redundant information. Furthermore, In order not to introduce any distortions to the signals, only lossless compression approaches are considered. To lower the file size further, a signal-aware approach needs to be established to allow for lossless handling of the radio samples.

To find further memory savings, we can look to the frequency domain of the captured signal. The lowest sampling rate provided by the SDR - 750 kSps - corresponds to ± 375 kHz of bandwidth around the center frequency in the radio spectrum. As shown by the M17 test, this bandwidth covers a

spectrum containing multiple signals, especially in the 70 cm ISM and Amateur Radio band. However, the bandwidth of any individual signal is as low as 12.5 kHz - less than 2% of the entire spectrum. Applying the Nyquist–Shannon sampling theorem, by lowering the bandwidth of the recording and focusing only on interesting signals, we can lower the sampling frequency even more, thus lowering the total amount of samples required to store the signal.

To implement this approach, lightweight Digital Signal Processing (DSP) tools can be used. The DSP implementation shall be resource efficient in order to abide by the CPU limitations. After saving the full spectrum recording to memory, software tools can be used to filter out and downsample subsets of the entire frequency spectrum resulting in much smaller file size. This approach works on the assumption that the user of the SDR subsystem knows which signals are interesting to them, and can pinpoint the center frequency and bandwidth of the designated signal.

To aid in the selection of parameters for signal processing, additional tools need to be introduced. Borrowing from the camera experiment, the experiment shall provide a way to preview the captured signal. Once the recording is finished, a preview can be generated and downlinked to Earth. Then it can be used by the MCT to make operational decisions about the downlink of data. By inspecting the preview of the spectrum, a single signal can be selected and parameters for data processing can be established. Once fed back to the experiment, the DSP tools shall be used to extract the recording from memory and prepare it for downlink.

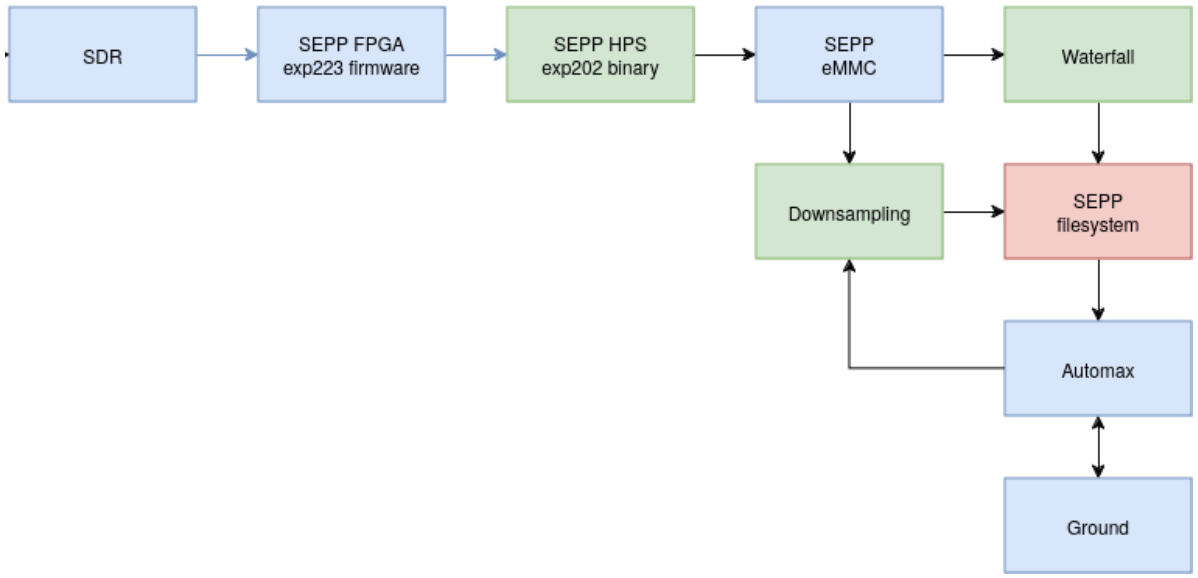

Fig. 26: System diagram of the implementation concept.

7.4. The implementation

Influenced by prior knowledge of operations, the experiment implementation was done with focus on ease of use and configurability. Because the experiment is set to implement a set of distinctive functions used at different times in the experiment lifecycle, the work has been organized into “actions”, each corresponding to its own subroutines with distinctive configuration parameters. The actions are:

- Record - the main action of the experiment, configuring the SDR and starting the capture of samples, writing directly to eMMC.
- Waterfall - generating a preview of the captured signal.
- Downsample - extraction of signals from the full recording stored in memory
- Downlink - special action allowing for download of the entire recording stored in the eMMC, replicating the legacy behavior and used for development.

Fig. 27: Experiment actions.

7.4.1. Record action

Main work on the eMMC direct write approach has been done by making changes to EXP202 (SIDLOC) source code. The original implementation used a fixed path for storing the samples, by the “toGround” folder of the experiment. The filename of the recording could be defined in the configuration file and the current timestamp was always appended to it.

First modification made the output path of the experiment customizable via the configuration file. This made it possible to change the output location to anywhere in the SEPP filesystem, not only the “toGround” folder. Due to the nature of Linux OS, portions of external memory can be mounted to the filesystem and used like a regular file (as shown in List. 1). In the case of EXP266, a portion of the eMMC destined to store the recorded samples is mounted during initialization of the experiment. The mounting process creates a file in the Linux filesystem, which can be specified as the write path for the SDR driver. Once the recording is started, the samples no longer take up the space in the Overlay FS of the SEPP, but are instead forwarded directly to the mapped memory space on the eMMC.

Due to the fact that the mounted memory partition does not have any file system on it, the raw samples saved to it are indistinguishable from the data already present in this memory space. Previously in OPS-SAT operations, the GNU TAR [23] program has been used to circumvent this issue. The program takes metadata (like the name, time of creation etc.) of the files and stores them in appropriate headers. Then it outputs the headers together with the binary data of the files, which can be stored in an arbitrary storage medium. Historically, this format has been used to store data on non-digital mediums like magnetic tapes, hence the name “Tape Archive”. When retrieving the data back from the medium, the file headers are stripped, and the data together with its metadata

information is replicated back in the filesystem used on the digital storage medium. This use case is still relevant today, and can be used in this scenario.

To make use of the TAR file format in the SDR experiment, a lightweight C library implementing the TAR headers has been used [41]. Before beginning the recording, a TAR header is written out to the output. Then a filename header is written containing configuration parameters of the SDR. Only then the experiment starts the capture of RF samples. Once the recording is complete, the appropriate TAR footers are placed to finish the file. When the eMMC memory is inspected using the TAR program, it detects the headers, and writes back the metadata information. Additional scripts can read this data and extract the information about SDR parameters from the filename.

Fig. 28: Structure of files saved to eMMC.

```
  /* Set the output path from config file */
const std::string output(output_path);
  /* Generate filename using SDR configuration parameters*/
  std::stringstream s;
  s << "sdr_exp202-f_sampling_index=" << f_set << "-lpf_index=" << bw_code <<
"-f_center=" << carrier_frequency_ghz << "-gain=" << gain_db << "-timestamp=" <<
std::time(0) << ".cs16";
  const std::string samples_filename(s.str());
  iq_capture(output, samples_filename, is_constrained, nsamples);
```

List. 2: Code sample of the filename generation.

```
## Stored recording parameters:

Stored filename:  sdr_exp202-f_sampling_index=0-lpf_index=15-f_center=0.43355-gain=60-timestamp=1703876618.cs16;
Center frequency: 0.43355 GHz;
Sampling Rate:    1.5 MHz (id: 0);
Low Pass filter:  0.75 MHz (id: 15);
Gain:             60 dB;
```

Fig. 29: Recording metadata saved in the TAR filename headers.

```
  /* Open archive for writing, write file header with filename */
  mtar_t tar;
  mtar_open(&tar, output.c_str(), "w");
  mtar_write_file_header(&tar, samples_filename.c_str(), nsamples*4);

// Recording implementation omitted

  /* Finalize the archive */
  mtar_finalize(&tar);
  mtar_close(&tar);
```

List. 3: Code sample of the filename generation.

After modifying the SDR driver, the Source Code has been compiled for the SEPP architecture using the toolchain provided by the OPS-SAT Experimenter Platform [36]. The resulting binary file has been included in the experiment package of the EXP266.

To implement the “Record” action itself, the binary has been wrapped in an initialization script written in Bash shell environment [42] . The script reads the configuration file of EXP266 and extracts parameters relevant to the operations of the SDR in this action. Then, it prepares the environment by ensuring that the eMMC memory is available, and the FPGA firmware required to run the SDR streaming interface is present. Then, the modified SDR driver is executed, starting the recording process. Once the recording is done, the script runs the TAR command to extract metadata stored in the eMMC, confirming the successful execution of the experiment. All messages generated by the experiment are stored in a log file saved in the “toGround” folder of the experiment, which can later be downlinked to ground. In this implementation, the log is not stored in the persistent memory, but other tools can be used to append it to the TAR archive in eMMC memory.

The configuration parameters of the Record action are:

- **carrier_frequency_GHz** – Middle frequency the SDR shall be tuned to. Accepts arguments in range from 300 MHz to 3 GHz (bearing in mind limitations introduced by the hardware low pass filter and the tuning frequency of the antenna). To achieve good results, the frequency needs to be offset by a small value from the expected signal carrier frequency to account for DC-offset noise.
- **samp_freq_index** – Number index defining the sampling rate value to be used. The number reflects the position of pre-defined sampling rates in the firmware of the SDR subsystem. The values are: 1.5 1.75 3.5 3 3.84 5 5.5 6 7 8.75 10 12 14 20 24 28 32 36 40 60 76.8 80 [MHz].
- **lpf_bw_cfg** – Configuration of the hardware Low Pass Filter included in the SDR subsystem. The value needs to be lower than half of the value of the sampling rate set above. The available values are: 14 10 7 6 5 4.375 3.5 3 2.75 2.5 1.92 1.5 1.375 1.25 0.875 0.75 [MHz].
- **gain_db** – Software gain applied by the SDR subsystem. Values from 12 to 72 dB are supported. 60dB has been used in all tests with good results.
- **number_of_samples** – Number of samples to be requested from the SDR subsystem. Each sample is two 16 bit values of I and Q signal parameters. Two values are used:

  25000000 – Matches the size of 100 MB, keeping the recording under the memory constraints of the SEPP.

  48000000 – Calibrated to the size of the eMMC partition, just under 190 MB.

  This value can be modified once a bigger eMMC partition is established, further expanding the capabilities of the experiment.

These configuration parameters have been implemented all the way back in the original implementation of the SDR subsystem DMA driver. The streaming driver and now EXP 266 uses the same parameter names, as they are well established and understood by the MCT.

7.4.2. *Waterfall generation action*

Once the recording action completes, a preview image of the captured signal can be generated using waterfall plot generation software. For this purpose Open-Source Renderfall [43] program is used. It has been compiled and included in the experiment package.

Waterfall plots are generated by dividing a signal into short time segments and applying the Fast Fourier Transform (FFT) to each segment to show how frequency content changes over time. The program needs to be provided with a FFT window size, determining the horizontal resolution of the plot. Larger FFT size results in finer frequency details per each line of the plot but poorer time resolution, whereas a smaller size offers better time accuracy, but less frequency detail. A windowing function is applied to each time segment to minimize distortion, reduce artifacts and improve the clarity of the plot.

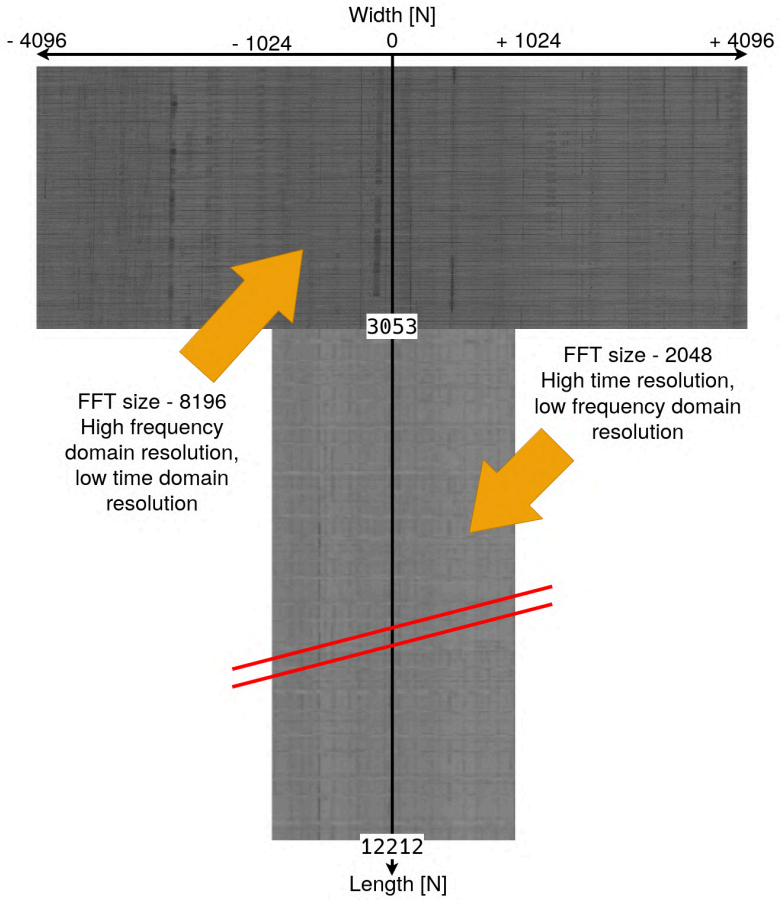

Fig. 30: Impact of the FFT size on waterfall plot generation.
Example of Renderfall generated waterfall plot.

The program is provided with the eMMC partition mount point as input file. The program reads samples out of the eMMC and generates a PNG picture into the SEPP filesystem. The resulting file size is proportional to the length of the SDR recording. For a 100 MB recording, the PNG waterfall image weighs about 20 MB. This file size is substantially lower and easier to handle than the original recording. Using this approach, the SDR recording size in eMMC could be expanded up to 5 times before the waterfall image size hits the size constraint of 100 MB. Optionally, the PNG file can be

converted to JPG format, reducing the file size furthermore by about a half, depending on the content of the file. This process however generates another file alongside the original PNG file which might break the set memory constraint. Once the JPG is generated, the PNG is deleted.

The EXP266 exposes key configuration parameters to the user, which are passed to the Renderfall program at launch. The parameters are:

- **action=waterfall** - tells EXP266 to run this action.
- **waterfall_fft_size** - a valid integer going up in powers of 2, like 1024, 2048 … 16384 etc.
- **waterfall_window** - name of one of the supported windowing functions:
    - hann, gaussian, square, blackmanharris, hamming, kaiser, parzen.
- **waterfall_convert_to_jpg** - optional parameter to turn on or off the JPG compression.

7.4.3. Downsampling action

The downsampling action is the most important one from the perspective of data processing. Due to unavailability of the GNU Radio suite, another lightweight implementation of signal processing tools has been used to implement it. After verifying the available resources, iq_toolbox [44], licensed under BSD License, has been selected.

The program provides a selection of optimized C++ Digital Signal Processing programs following the principles of UNIX Philosophy [45]. Each program implements a single functionality which can be executed in the Bash shell environment [42]. Data can be redirected between files and programs using UNIX Pipelines [46]. This allows for non-graphical, programmatic setup of Signal Processing pipelines directly in the shell environment of the Linux OS on the SEPP. This approach is akin to the way GNU Radio operates in its graphical user environment, but can be executed and modified in flight. The entire program suite has been compiled and included in the EXP266 package.

The name of the action refers to the process of reducing the sampling rate of a signal - Downsampling. This is done by keeping only every *N*-th sample of the original signal and discarding the rest, where *N* is the downsampling factor. This process enables great memory savings, but before it can be performed, requires additional steps to preserve signals of interest to the user.

To keep the signals from being rejected, they first need to be shifted to the center of the recorded spectrum. For this purpose the iq_mix program is used. The program adds a value to each recorded sample, which is equivalent to mixing additional frequency in the frequency domain. causing all signals in the spectrum to be shifted up or down (or right / left from the point of view of the waterfall plot). To move the signal to the center, the iq_mix is provided with the frequency offset of the signal from the center of the spectrum. When the negative value is applied to the signal, the signal is moved to the center of the spectrum.

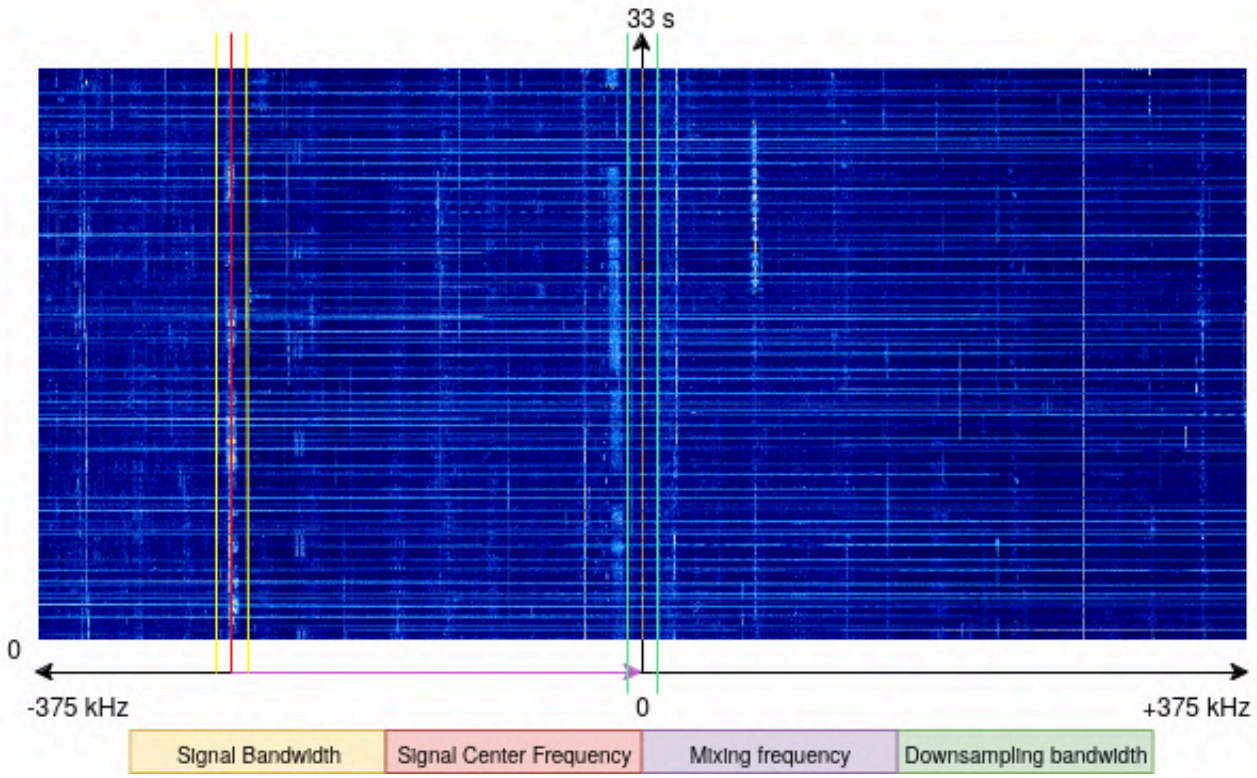

Fig. 31: Visual representation of the mixing, filtering and downsampling process of the radio spectrum.

Once the signal is in the center, another program is used - iq_decimate. It employs two functions at the same time. First, it applies a software low pass filter to the spectrum, to remove all signals which are outside of the bandwidth which is of interest to the user. Then, the downsampling is performed, by rejecting underlying samples in the signal. The digital Low Pass Filter setting is defined by the user by providing a cut-off frequency. It is equivalent to the bandwidth of the signals to be kept, plus a margin accounting for doppler shift. The decimation rate is automatically calculated to keep the lowest possible bandwidth without intersecting with the non-filtered signal. The entire pipeline is executed as one line in the Bash shell:

```
> stream_emmc.sh | tar -xvO | iq_mix -s $sampling_Hz -m $downsample_shift |
  iq_decimate -s $sampling_Hz -f $downsample_cutoff_frequency -o $filename
```

List. 4: Downsampling pipeline.

The pipeline reads data from eMMC using a helper script stream_emmc.sh, and extracts the stored tar archive. After stripping the tar headers, the binary data is fed to the iq_mix and iq_decimate programs respectively. The data is streamed between programs and just the final downsampled signal is stored on the filesystem. For the programs to run correctly the sampling rate of the signal needs to be provided. This value is extracted beforehand from the filename tar header stored in eMMC. The output of the pipeline is stored to a file in the toGround folder of the experiment, ready to be downlinked. The output signal is furthermore compressed using gzip [40], to reclaim space lost due to padding of the 14 bit IQ samples to 16 bit buffers in the filesystem. The size of the output is dependent on the downsampling parameters, with the size of the file being inversely proportional to

the selected decimation factor. For example, the filesize of a downsampled signal with the resulting bandwidth of 100 kHz is 8.75 MB at 33 seconds of recording length. Respectively, a 25 kHz bandwidth output is about 2.2 MB. Thanks to these improvements, the signals can be downloaded as quickly as in one satellite pass over Darmstadt, quicker than the waterfall preview of the entire spectrum.

Same as with other EXP266 actions, the behavior is configurable via the configuration file:

- **action=downsample** - specifies this action to be run.
- **downsample_shift** - Definines how far from the middle of the recording is the requested signal. Value in Hz.
- **downsample_cutoff_frequency** - Defines how wide the requested signal is. Value in Hz.

7.4.4. Frequency picker

To aid in the selection of appropriate parameters for the Downsampling action, especially to MCT members who are not versed in the operations of SDR, a simple graphical interface has been developed in Python. The application is meant to be run on a personal computer of MCT members or experimenters working with the SDR.

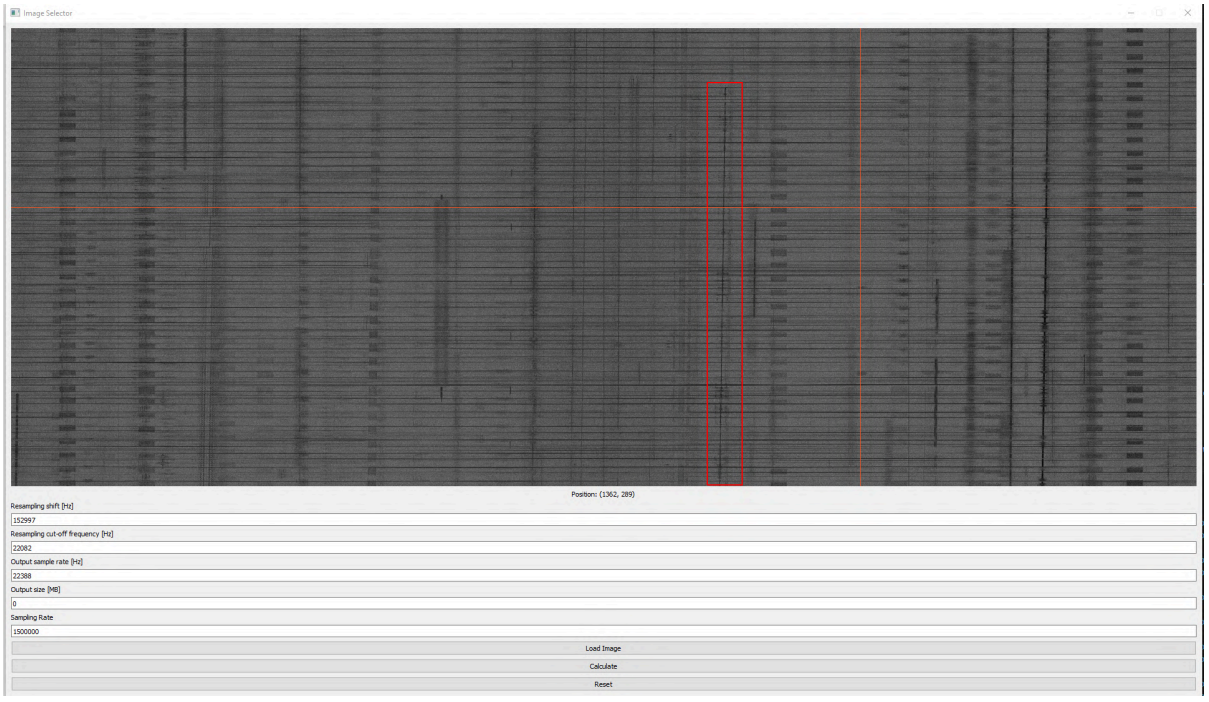

Fig. 32: EXP266 Frequency Picker interface.

In the application window, the user can open a waterfall image generated in space using the waterfall action of EXP266. Additional metadata of the captured signal needs to be provided - the number of samples (length of the recording) and the bandwidth, which have been used to schedule the experiment in the first place. Then the operator can make a visual inspection of the plot and select one of the many signals visible in the spectrum. A best-effort analysis needs to be performed based

on the visual characteristics of radio signals in order to find the signal of interest. A good estimate can be made by selecting the transmission closest to the expected transmission frequency.

As seen in the M17 test the signals are slightly shifted due to Doppler-shift, which needs to be taken into account. For signals originating from Earth, the approximate doppler shift in the 400 MHz band at the orbital speed of OPS-SAT can be calculated:

$$V = 7.712\ km/s\ (at\ 330\ km)$$
$$c = 300\,000\ km/s$$
$$f = 400\ MHz$$

$$\Delta f = \frac{V_r\ f}{c} = \frac{7.712\ [\frac{km}{s}] \cdot 400\ MHz\ [\frac{1}{s}]}{300\,000\ [\frac{km}{s}]} = \pm 0.01\ MHz = \pm 10 kHz \quad (3)$$

Although subtle compared to the entire capture spectrum of the SDR, while trying to pinpoint a particular signal, an uncertainty band of ± 10 kHz needs to be taken into account. This can be done for the most part with just visual inspection of the spectrum.

After selecting the signal using a mouse, the program outputs downsampling parameters - the center frequency and cut-off frequency - which need to be included in the configuration of the downsampling action and scheduled to execute in orbit. Additionally, the program calculates the time-domain parameters of the start and end of the signal, but those parameters are not implemented within EXP266. This leaves the opportunity to introduce additional processing in the future that can be used to reject samples in the time domain from the downsampling action output.

7.5. Concept of operations

The implementation assumes that the FPGA firmware supporting the SDR streaming interface is loaded. In such a case the spacecraft can keep its Horizontal Pointing attitude at all times without interruption. Otherwise, the SEPP requires a reboot in order to load the required firmware, which might affect the attitude of the spacecraft, increasing drag during the execution of the experiment.

The EXP266 SDR operations begin with establishing the parameters for the recording: time, length, center frequency and bandwidth. If the signal to be recorded is expected to come from Earth's surface, tools from camera subsystem operations, like VTS Timeloop [33], can be used to establish a timeframe for the recording. If the length of the recording is expected to be longer than 60 seconds, additional steps need to be taken to prepare a new eMMC partition for storage of the recording.

Once the baseline is established, a configuration file can be prepared, including the set parameters and selecting the “Record” action. The configuration needs to be placed on the spacecraft using Automax or equivalent commands via the Master Schedule. Then, the experiment can be scheduled to be executed at the desired time frame using the Master Schedule.

10min SDR in EXP223	FPGA Firmware: RECORD		
Reference event	SDR_EXP266_CUSTOM		
event - 1min30s	TT_UHF_N220	Power off nanocom	
event - 1min15s	TT_SDR_N210	Power on SDR	
event - 1min10s	TT_SYS_N510	Config: Carrier Frequency [GHz]	sed -i 's/action=.*/action=record/' /home/exp266/config.ini
event - 1min	TT_SYS_N510	Config: Carrier Frequency [GHz]	sed -i 's/carrier_frequency_GHz=.*/carrier_frequency_GHz=0.433550/' /home/exp266/config.ini
event - 55s	TT_SYS_N510	Config: Sampling Frequency Index {0...21}:	sed -i 's/samp_freq_index=.*/samp_freq_index=0/' /home/exp266/config.ini
event - 50s	TT_SYS_N510	Config: Low Pass Filter Bandwidth {0...15}:	sed -i 's/lpf_bw_cfg=.*/lpf_bw_cfg=15/' /home/exp266/config.ini
event - 45s	TT_SYS_N510	Config: Gain {12..72} [dB]	sed -i 's/gain_db=.*/gain_db=60/' /home/exp266/config.ini
event - 45s	TT_SYS_N510	Config: Calibrate Frontend	sed -i 's/calibrate_frontend=.*/calibrate_frontend=1/' /home/exp266/config.ini
event + 0s	TT_SYS_N510		su - -c "cd /home/exp266;./start_exp266.sh"
event + 9min31s	TT_SDR_N220	Power off SDR	
event + 10min	TT_UHF_N210	Power on nanocom	

Fig. 33: Prepared Master Schedule templates for EXP266 actions.
Configuration file modifications are made using sed program [28].

Once the experiment is triggered, it automatically configures the SDR and its HPS driver starting the recording straight to the eMMC partition. Once finished, automatically triggers the Waterfall generation action, placing the resulting plot in the “toGround” folder of the experiment. During the next satellite pass over Darmstadt, the preview is downlinked, and can be verified by the MCT or experimenter requesting the recording.

The waterfall preview can be loaded into the Frequency Picker program, in which the user can select the desired signal, generating parameters for extraction via the Downsampling action. Then, the action can either be scheduled in the next Master Schedule, or performed live during a pass, using the Automax tool. The desired signal will be downlinked subsequently. This action can be repeated as many times as required, extracting as many signals as necessary or until the entire recording is downlinked.

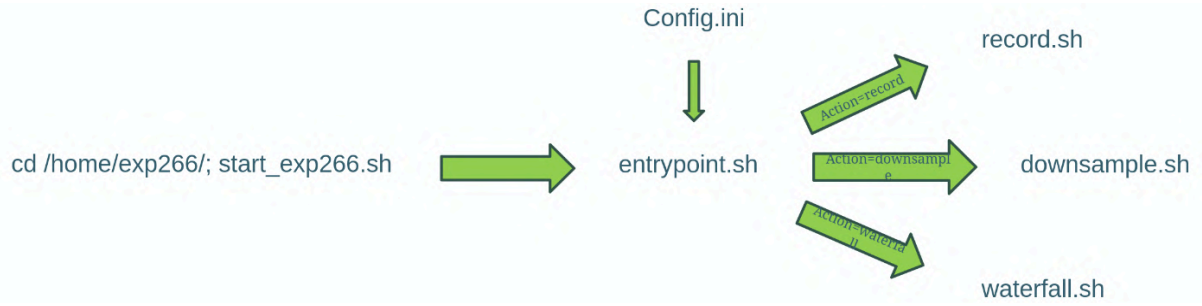

Fig. 34: EXP266 script structure. The actions can be triggered manually via SSH / Automax.

```
05_pre_pass_example.sh          14_s_set_dual_core.sh           30_s_unpack_tar.sh              87_g_downlink_filestore2.sh
'10____start of pass____'       16_s_exp_mngmnt.sh              50_g_uplink_filestore.sh        88_s_largestFolders.sh
12_g_post_sepp_recovery.sh      20_s_standard_activities.sh     70_g_uplink_filestore2.sh       89_s_list_tar_partitions.sh
13_g_config_sync.sh             21_s_memory_available.sh        73_s_storeFiles_NoFS_dry.sh     '90____end of pass____'
13_s_extra_check.sh             22_s_check_FMS.sh               74_s_update_exp172_HOPAS.sh     95_post_pass_example.sh
13_s_investigation.sh           22_s_check_NMFS.sh              80_g_downlink_filestore.sh      98_pass_summary.sh
13_s_reshape.sh                 23_s_check_spp.sh               81_s_e2fsck_dryrun_dmesg_P6.sh  bg_g_ping.f.sh
13_s_untar_fpga_exp_results.sh  25_s_check_overlay.sh           82_s_check_NMFS.sh              bg_g_uplink_filestore.sh
```

Fig. 35: View of Automax scripts to be executed during a pass. Highlighted scripts (with granted execution permission) are started in numeric order. Scripts containing “g” are executed on Ground Segment, and scripts containing “s” on board SEPP.

8. RESULTS OF THE EXPERIMENT

One of the first versions of the experiment, implementing waterfall plot generation action and automatic eMMC storage after recording, was tested in a real world scenario in December 2023. Experiment 266 has been scheduled during one of the evening passes over Europe. The spacecraft has been rebooted into the configuration including the FPGA SDR streaming interface from Sidloc. The recording session has been scheduled at the time of closest approach to the city of Hamburg, Germany. On ground, a ground station has been constructed using a Yaesu FT5DE handheld radio and Arrow II 10 element Yagi antenna [47] . The calculated link budget for the scenario is presented in Table 2.

Table 2: Link Budget for the December 2023 test.

Component	Description	Value	Unit
Transmitter	Transmit Power	37	dBm
	Yagi elements	10	-
	Antenna Gain	11.80	dBi
	Cable Loss	0.25	dB
Path	Distance	800000	m
	Frequency	433.625	MHz
	Wavelength	691.84	mm
	Free Space Loss	143.254	dB
Satellite	Antenna Gain	19.5	dBi
	Cable Loss	0.25	dB
	Receiver sensitivity	-98	dBm
System	Expected Signal Level	-75.454	dBm

The antenna used this time was slightly longer, including 3 additional elements over the November M17 test with SP5WWP. The theoretical transmit power was also slightly higher, with 5 W as opposed to 3 W max power output of the handheld radio. Despite this, due to the longer path to the satellite - about 800 km - the link margin is about the same in both tests. The longer distance comes from the fact that the satellite was overflying Hamburg at lower elevation, about 45^{o} above horizon.

With the link budget looking favorable, the test transmission to the spacecraft was conducted during a live demonstration taking place at the 37th Chaos Communication Congress (37C3), one of the leading cybersecurity events in Europe. The test gathered a small audience of fellow amateur radio enthusiasts and members of the Libre Space Foundation, who have developed the Sidloc FPGA image used in this setup. The voice transmission started exactly on December 29th, 2023 at 19:03:38 UTC, the moment the experiment was scheduled on board OPS-SAT.

The message included greetings from earth and respective amateur radio callsigns of the operators present. Another transmission has been conducted in parallel including an QSL card transmitted via Simple Scan Television (SSTV) encoding.

Fig. 36: Ground antenna setup for the December test.

The recording lasted 33 seconds, still being limited by the SEPP memory constraint of 100 MB. After the recording, EXP266 automatically stored the recording into the eMMC. Then the SEPP was reloaded into the FPGA “standard image” enabling nominal operations using Automax. At the time, the outcome of the experiment was not known for about two more weeks, during which the downlink of samples was not a priority. After all, it was the beginning of the new year and many MCT members were unavailable. The spacecraft also experienced unforeseen reboots which needed to be addressed first. Additionally, with the altitude of the spacecraft decreasing ever more, it has entered a more turbulent period of operations.

When the spacecraft became stable and entered nominal operations again, the waterfall action of the experiment was executed, confirming the presence of the recording on board. The 8 MB waterfall image in JPG format could be downloaded over the remaining evening passes. Once the presence of the recording in memory was verified, the downlink of the entire binary file has been scheduled. This version of the experiment did not include the re-sampling part yet.

The downlink was completed over the S-Band channel in the evening, morning, and next day's evening satellite passes. Each pass another chunk of the recording was available and the transmission could be decoded, confirming the link budget margin was indeed enough to detect the signal on board. The decoded voice signal was quite faint, but enough to distinguish the transmitted message, especially with prior knowledge of its content. The SSTV transmission could not be decoded.

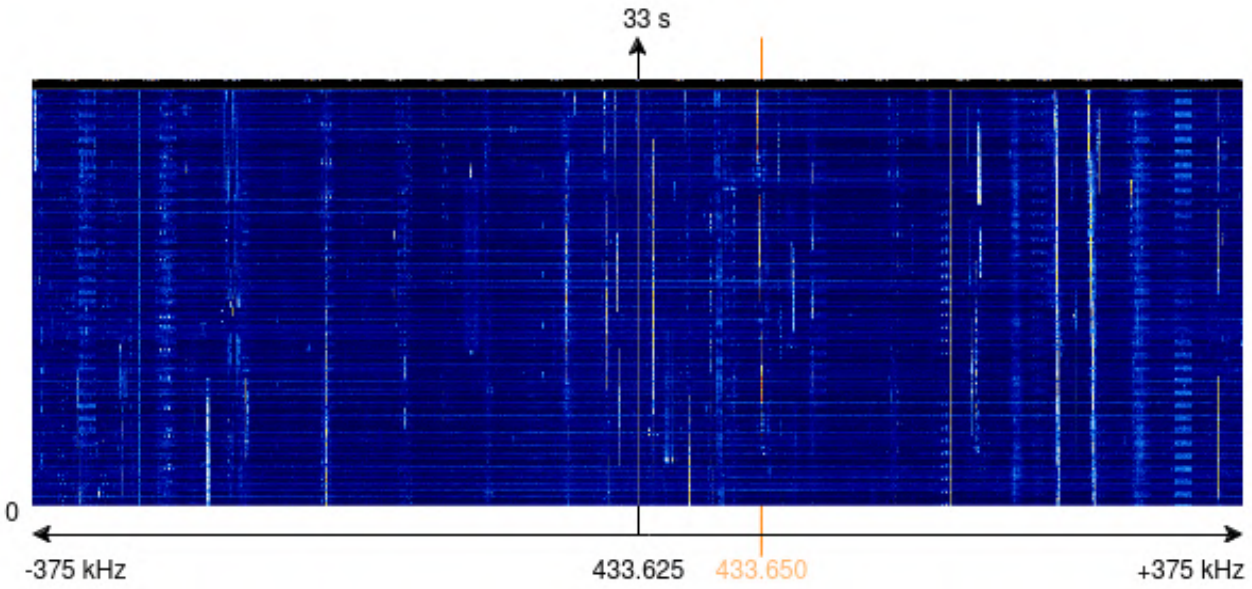

Fig. 37: Full recording downlinked to Earth. Center and expected signal frequency highlighted.

With the full recording on ground, later processing and development of the experiment was conducted on the Engineering Model of OPS-SAT. Just like on the real spacecraft, the recording was loaded into the eMMC in the same fashion as if it would be recorded via SDR. Then the down-sampling action of the experiment could be applied, extracting the transmission at given parameters. The resulting file weighed around 9 MB. If it were conducted on board, it could be downloaded in about the same ground-station time as the waterfall preview of the entire recording.

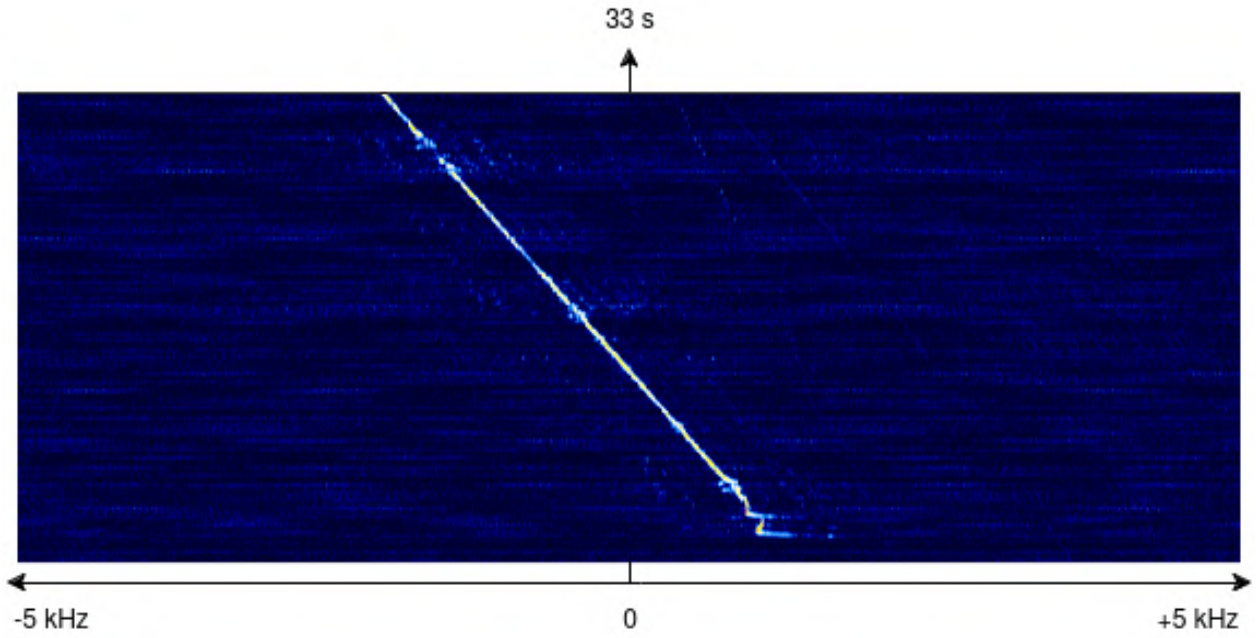

Fig. 38: The result of EXP266 down-sampling action - ±5 kHz bandwidth. Frequency shift due to the Doppler effect is visible.

Throughout January intermediary versions of the experiment were uplinked and scheduled in orbit, but due to other issues with the spacecraft, persistent storage was not used for the storage of results. At least two additional experiment executions have been attempted - over Ukraine and Yemen - with hopes to measure possible interference in the radio spectrum over active war zones. Their successful execution was confirmed via telemetry, but the results could not be downlinked due to reboots of the spacecraft. The downsampling action has been tested in space using the stored recording from the December test, but no end-to-end live recording, downsampling and downlink test was performed in the end.

The final implementation of the recording action of the experiment, including direct writes to eMMC, has been validated on the EM. To test it, several recordings using the EM SDR subsystem have been conducted, making use of the entire 196 MB eMMC partition. The EM recordings contained mostly noise and thus are not presented here. The final version was pending full deployment in space

Due to the unavailability of OPS-SAT as it was coming to the end of the mission the final version of the experiment has not been used in space to the full extent. Two weeks after finishing the internship, on February 18th, 2023, the execution of new experiments was suspended, with the MCT focusing on finalizing execution of the current backlog.

9. CONCLUSIONS

The OPS-SAT-1 mission came to an end around 22:30 UTC on May 22nd, 2024 [48], when it decayed into Earth's orbit. During its last days, the mission was followed by worldwide amateur radio using Open Source GNU Radio modules [49] decoding its telemetry to the very last moment.

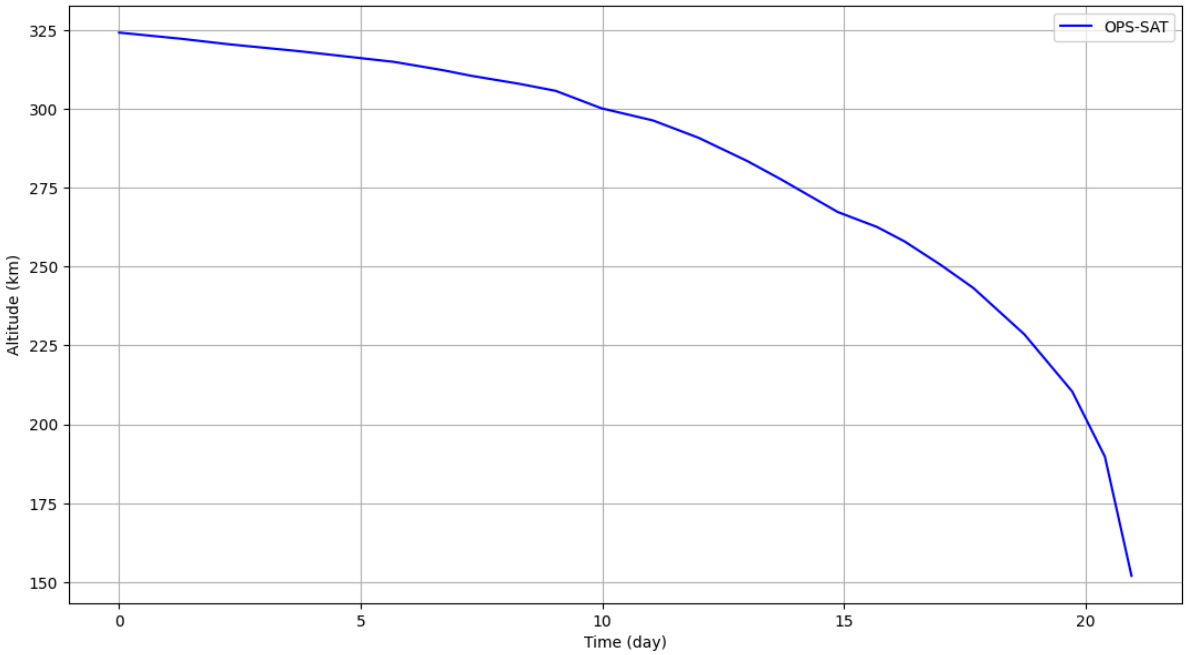

Fig. 39: OPS-SAT Mean altitude decay in May 2024 [48].

The OPS-SAT mission has been testament to what is possible when access to space is democratized. The provided platform allowed hundreds of scientists, engineers, startups and well established institutions to test a great variety of new concepts, gaining space heritage and appreciation for the ways of space operations. Over the years, OPS-SAT Space Laboratory created a worldwide community of experts sharing knowledge and building the basis of a very new approach to space exploration for years to come. Work within the Space Laboratory is not done yet. New missions are planned with the upcoming OPS-SAT VOLT [50] launch as soon as in 2025. The next mission is aimed to introduce new opportunities for experimentation in the optical and quantum communication field.

In the context of this master thesis OPS-SAT has been a great learning opportunity. Over six months spent in Darmstadt, it's been a quick lesson in spacecraft operations. It allowed the author to use the gained experience to implement their own experiment and run it in space, organizing knowledge and boiling it down into the best version of the experiment. Experiment 266 is not so much as a "state of the art" implementation in itself, but rather a summary of all operational experience gained during the internship. Its purpose was to apply all of the practices that were used with other experimenters, creating a single, coherent way to operate the SDR subsystem. The achieved goal is the implementation of an operational concept integrating all subsystems of OPS-SAT, encompassing the entire mission architecture - from the SDR and SEPP, downlink/uplink chain, to the Ground Segment, Mission Control Systems and finishing at the operational practices of the Mission Control

Team itself. From a technical point of view, the experiment had to work around peculiar limitations resulting from component degradation, but also took advantage of previous developments which made it easier than ever to deploy new software in orbit.

This work would not be possible without the Mission Control Team support, the worldwide community and the Open Source nature of the OPS-SAT ecosystem. Due to a mix of Open Source licensed software being used in the implementation of the SDR experiment, the source code is publicly available in a repository [51] and included as attachment to this thesis.

9.1. Future work

Space missions to date have made use of limited computing resources, using slower, but more dependable hardware. Instead, the missions relied on sizable ground segments, powerful ground stations, and Mission Control Systems for communication, processing of data, and decision-making. Due to the high demand for Earth observation data used in a variety of branches of the economy, new, high-resolution sensors are being sent to orbit, generating more data than ever. The downlink of this data is often limited by the connectivity methods with space in terms of throughput and latency, especially when decision-making is involved. This highlights the usefulness of moving the processing of data to space, using flexible computing platforms that can be easily reconfigured in flight. This approach has been demonstrated by OPS-SAT and is already seeing applications in the private sector. Companies like KP Labs are developing a new generation of purpose-made data processing units to be placed on board a new generation of satellites [52].

The development of new operational concepts, including processing algorithms in space, is more important than ever before. This approach can take advantage of these powerful processing platforms to process data at the edge of space. Furthermore, for deep-space missions, higher computing power can result in more mission autonomy, with spacecraft making autonomous decisions based on sensor data. This approach also brings another, surprising opportunity for improvement. By bridging the gap between computational resources in space and on the ground, more expertise from Information Technology and Telecommunication fields can be applied to space. This allows years of Earth-based expertise among experts to be rapidly transferred to space systems. By democratizing access to space, the future of humanity in space exploration can be accelerated.

Although limited to an implementation in the constrained environment of OPS-SAT, the experimental work shown in this thesis highlights an important concept of operations for future space missions. Providing future opportunities to launch software experiments on satellites containing Software Defined Radios, the implementation could be expanded by using more tools in the Digital Signal Processing toolbelt to achieve efficient processing of many other kinds of signals directly on board. With more available computational resources, the same techniques could be applied in real-time processing. Automatic modulation classification would enable the satellite to identify and prioritize signals of interest, reducing the amount of data that needs to be downlinked and enhancing the mission's efficiency.

Additionally, as shown by other experiments on OPS-SAT, Machine Learning algorithms could be used to make autonomous decisions about the data. The inclusion of machine learning models on board could automate the identification of key patterns and anomalies in the data, enabling a higher level of decision-making autonomy. Models could be trained to detect known signal patterns or unusual events, such as interference or unexpected signals, and respond accordingly.

Moreover, future iterations of the SDR experiment could leverage use of the FPGA or specialized neural processing units (NPUs). These platforms offer a blend of high-performance computation and reconfigurability, making them ideal for processing complex algorithms, like neural networks, directly in orbit. Future work could explore integrating such hardware to complement or replace current SDR setups.

The field of satellite data processing holds a key to the future of humanity among the stars. Before we safely send humans to explore and colonize new worlds, we ought to first explore them using machines. For this, mission autonomy and data processing algorithms are essential for the exploration of the most distant worlds. The technology and methodology developed and validated in this thesis serve as a stepping stone toward achieving these goals. By further enabling autonomous data processing and decision-making on board, we could move one step closer to creating spacecraft that can independently survey, analyze, and respond to their environment. This ability will be critical as we expand to the most remote and uncharted regions of our solar system and beyond, paving the way for humanity's ultimate goal of becoming a multiplanetary species.

BIBLIOGRAPHY

[1] European Space Agency. “The ESRO Convention and 'juste retour.'” European Space Agency, 14 June 2014, https://www.esa.int/About_Us/ESA_history/The_ESRO_Convention_and_juste_retour. Accessed on 30 September 2024.

[2] Schäfer, Madeleine. “How To Survive in Space - A light-hearted chronicle of ESOC.”, 1997, https://www.esa.int/About_Us/ESOC/History_of_ESOC_How_to_survive_in_space. Accessed on 30 September 2024.

[3] European Space Agency. ESTEC-PHOTO-1968.06.349-002. 1968. ESA Space Heritage Image Project, https://ship.esa.int/. Accessed on 30 September 2024.

[4] Evans, David, et al. “OPS-SAT: Preparing for the Operations of ESA’s First NanoSat.” 14th International Conference on Space Operations, 2016, http://dx.doi.org/10.2514/6.2016-2490.

[5] European Space Agency. “ESA Operations: OPS-SAT.”, https://web.archive.org/web/20191218055849/https://www.esa.int/Enabling_Support/Operations/OPS-SAT. Accessed 18 December 2019.

[6] European Space Agency, ©ESA/Sentinel, https://sentinels.copernicus.eu. Accessed on 8 September 2023

[7] Evans, David, et al. “OPS-SAT: FDIR Design on a Mission that Expects Bugs - and Lots of Them.” 14th International Conference on Space Operations, 2016, https://arc.aiaa.org/doi/pdf/10.2514/6.2016-2481.

[8] European Space Agency, ©ESA/OPS-SAT, https://www.esa.int/Enabling_Support/Operations/OPS-SAT. Accessed on 30 September 2024.

[9] Labrèche, Georges, et al. “OPS-SAT LEOP and Commissioning: Running a Nanosatellite Project in a Space Agency Context.” Small Satellite Conference, 2019, https://digitalcommons.usu.edu/smallsat/2022/all2022/113/. Accessed on 30 September 2024

[10] Institute of Communication Networks and Satellite Communications, “OPS-SAT Experimenter Interface Control Document”, Graz University of Technology, 2014. https://graz.elsevierpure.com/en/publications/ops-sat-phase-ab1-experimenter-icd. Accessed on 30 September 2024

[11] Intel Corporation. “Introduction to the Hard Processor System.” Intel, https://www.intel.com/content/www/us/en/docs/programmable/683126/21-2/introduction-to-the-hard-processor-system-98309.html. Accessed on 30 September 2024.

[12] Zeif, Reinhard, et al. “The redundancy and fail-safe concept of the OPS-SAT payload processing platform.” International Astronautical Congress, 2018,

https://www.researchgate.net/publication/358621803_The_redundancy_and_fail-safe_concept_of_the_OPS-SAT_payload_processing_platform. Accessed on 30 September 2024

[13] Henkel, Maximilian, et al. “Commissioning a Fully-Reconfigurable Communication Chain in a CCSDS-Compliant Way on OPS-SAT Satellite”, International Conference on Broadband Communications for Next Generation Networks and Multimedia Applications, 2022 http://dx.doi.org/10.1109/CoBCom55489.2022.9880813.

[14] Løfaldli, André, et al. “ESOC Ground Station Architecture for OPS-SAT.” 8th International Workshop on Tracking, Telemetry and Command Systems for Space Applications (TTC), 2019, https://doi.org/10.1109/TTC.2019.8895329.

[15] Krebs, Gunter D. “OPS-SAT.” Gunter's Space Page, 14 January 2023, https://space.skyrocket.de/doc_sdat/ops-sat.htm. Accessed on 30 September 2024.

[16] Labrèche, Georges, et al. “Agile Development and Rapid Prototyping in a Flying Mission with Open Source Software Reuse On-Board the OPS-SAT Spacecraft.” AIAA SCITECH, 2022, http://dx.doi.org/10.2514/6.2022-0648.

[17] Pearson, Steve, et al. A Full End-to-end Automation Chain with MOIS, PLUTO, MATIS, SMF and SCOS-2000, vol. 13th International Conference on Space Operations, 2014, https://dx.doi.org/10.2514/6.2014-1833.

[18] Hessinger, Felix. “Automation of Operation and Testing for the European Space Agency's OPS-SAT Mission.” Erasmus Mundus SpaceMaster, Aalto University, 2019, https://aaltodoc.aalto.fi/items/514b64e1-3d29-4764-a169-de835f6df9ba. Accessed on 30 September 2024

[19] Henkel, Maximilian. “Using a Reconfigurable FPGA for Emerging Space Applications.” TU Graz Doctor Thesis, https://doi.org/10.3217/trsg7-9xv26.

[20] Henkel, Maximilian, et al. “Mitigating and Recovering from Radiation Induced Faults in Non-Hardened Spacecraft Flash Memory.” IEEE Aerospace Conference, 2024, http://dx.doi.org/10.1109/AERO58975.2024.10521125.

[21] Henkel, Maximilian, and Vladimir Zelenevskiy. “Recovery From File System Corruption on the OPS-SAT-1 Experimental Processor.” Small Satellite Conference, 2023, https://digitalcommons.usu.edu/smallsat/2023/all2023/263/. Accessed on 30 September 2024

[22] Free Software Foundation, “dd” GNU.org, https://www.gnu.org/software/coreutils/dd. Accessed on 30 September 2024.

[23] Free Software Foundation. “TAR” GNU.org, https://www.gnu.org/software/tar/. Accessed on 30 September 2024.

[24] Mladenov, Tom, et al. “Implementation of a GNU Radio-Based Search and Rescue Receiver on ESA's OPS-SAT Space Lab.” IEEE Aerospace and Electronic Systems Magazine, vol. 37, no. 5, 2022, https://doi.org/10.1109/MAES.2022.3143875.

[25] Libre Space Foundation. “Spacecraft Identification and Localization.” https://sidloc.org/. Accessed on 30 September 2024.

[26] The Chess-OPS Contributors. “CHESS-OPS.” https://chess-ops.space. Accessed on 30 September 2024.

[27] The Stockfish Contributors. Stockfish - Strong open-source chess engine, https://stockfishchess.org/. Accessed on 30 September 2024.

[28] Free Software Foundation. “GNU sed.” GNU.org, https://www.gnu.org/software/sed. Accessed on 30 September 2024.

[29] Labrèche, Georges. “SmartCam.” https://github.com/georgeslabreche/opssat-smartcam. Accessed on 30 September 2024.

[30] Waage, Ólafur. “opssat-doom.” GitHub, https://github.com/olafurw/opssat-doom. Accessed on 30 September 2024.

[31] European Space Agency. “OPS-SAT Pictures Gallery.” Flickr, https://www.flickr.com/photos/esa_events/albums/72157716491073681/. Accessed on 30 September 2024.

[32] M17 Project Contributors. “M17 Project.” https://m17project.org/. Accessed on 30 September 2024.

[33] CNES - French Space Agency. “VTS Timeloop.” https://timeloop.fr/vts/. Accessed on 30 September 2024.

[34] Rothammel, Karl, et al. Rothammel's Antenna Book. 11 ed., p. 439, DARC Verlag, 2019.

[35] European Space Agency. “ESA's Java implementation of the CCSDS MO services.” GitHub, https://github.com/esa/mo-services-java. Accessed on 30 September 2024.

[36] European Space Agency. “OPS-SAT SEPP Continuous Integration Environment.” https://gitlab.com/esa/NMF/ops-sat-sepp-ci-docker/. Accessed on 30 September 2024.

[37] OpenWRT. “Opkg package manager.”, 12 April 2024, https://openwrt.org/docs/guide-user/additional-software/opkg. Accessed on 30 September 2024.

[38] Libre Space Foundation. “OPS-SAT SIDLOC FPGA Firmware.” https://gitlab.com/librespacefoundation/ops-sat-sidloc/ops-sat-sidloc-fpga. Accessed on 30 September 2024.

[39] Libre Space Foundation. “OPS-SAT SIDLOC SEPP HPS Driver.” https://gitlab.com/librespacefoundation/ops-sat-sidloc/ops-sat-sidloc-sw. Accessed on 30 September 2024.

[40] Free Software Foundation. “Gzip.” GNU.org, https://www.gnu.org/software/gzip/. Accessed on 30 September 2024.

[41] rxi. “Microtar: A lightweight tar library written in ANSI C.” GitHub, v. 0.1.0 https://github.com/rxi/microtar/. Accessed on 30 September 2024.

[42] Free Software Foundation, “Bash”. GNU.org, https://www.gnu.org/software/bash/. Accessed on 30 September 2024.

[43] Torborg, Scott. “renderfall: A tool to render spectrograms (a.k.a. waterfall diagrams) from raw RF or audio samples.” GitHub, https://github.com/storborg/renderfall. Accessed on 30 September 2024.

[44] emvivre@urdn.com.ua. “iq_toolbox: Toolbox for IQ signal processing.” GitHub, https://github.com/emvivre/iq_toolbox. Accessed on 30 September 2024.

[45] Raymond, Eric S. “Basics of the Unix Philosophy.” Catb.org, http://www.catb.org/~esr/writings/taoup/html/ch01s06.html. Accessed on 30 September 2024.

[46] Bell Labs. “Early Unix history and evolution: Pipes.” https://www.bell-labs.com/usr/dmr/www/hist.html#pipes. Accessed on 30 September 2024.

[47] Arrow Antennas. “Arrow II Portable Antennas: The Alaskan.” https://www.arrowantennas.com/arrowii/alaskanarrow.html. Accessed on 30 September 2024.

[48] European Space Agency. “OPS-SAT Reentry Dataset.” https://opssat.esa.int/ops-sat-1/reentry_dataset/. Accessed on 30 September 2024.

[49] European Space Agency. “gr-opssat: UHF specifications and example applications for demodulating and decoding the received OPS-SAT signal.” https://github.com/esa/gr-opssat. Accessed on 30 September 2024.

[50] European Space Agency. “OPS-SAT Space Lab - Volt.” https://opssat.esa.int/volt/. Accessed on 30 September 2024.

[51] Jasiukowicz, Marcin. “OPS-SAT SDR EXP266 implementation.” https://github.com/yasiupl/opssat-sdr. Accessed on 30 September 2024.

[52] KP Labs, “Leopard Data Processing Unit” https://kplabs.space/leopard/. Accessed on 30 September 2024.