Implementation of a GNU Radio Based Search and Rescue Receiver on ESA's OPS-SAT Space Lab

Tom Mladenov[1]*, David Evans[1] and Vladimir Zelenevskiy[2]
[1]European Space Agency (ESA), European Space Operations Centre (ESOC), Robert-Bosch-Straße 5, 64293 Darmstadt, Germany
[2]Telespazio Germany GmbH, Europaplatz 5, 64293 Darmstadt, Germany
*Corresponding Author: tom.mladenov@ieee.org

***Abstract*—OPS-SAT is the first publicly accessible Hardware/Software innovation Lab in Low Earth Orbit. It was launched by the European Space Agency (ESA) on December 18th 2019 and is open to European academia and industry allowing new concepts to be tested in space with a range of interesting payloads. This paper discusses the implementation and results of an In-orbit Demonstration (IOD) using the Software Defined Radio (SDR) payload of OPS-SAT together with on-board RF Processing techniques. We demonstrate the design of a software configurable Search and Rescue receiver using GNU Radio, running on Linux in a 3U CubeSat. The system runs on the Satellite Experimental Processing Platform (SEPP) Cyclone V ARM System-on-chip and is able to autonomously detect and decode transmissions from terrestrial 406 MHz beacons from the global COSPAS-SARSAT search and rescue system. Decoded beacon information is logged on-board together with metadata and is downloaded to Mission Control at ESA/ESOC. The successful IOD paves the way for in-orbit RF experimentation and bridges the gap between satellite operations and the IoT era.**

***Index Terms*—Satellite applications, Signal processing, Emergency services**

I. Introduction

SOFTWARE Defined Radio or SDR is already widely adopted in ground applications and currently present in many consumer devices such as mobile transceivers and car radios. Currently also in aerospace applications, SDR is slowly finding its place, albeit at a much lower adoption and integration rate than is seen in consumer devices.

The OPS-SAT spacecraft is a 3U CubeSat launched by the European Space Agency (ESA) on December 18th 2019 and is an open hardware/software innovation platform in Low Earth Orbit (LEO). It was launched into a 515 km Sun-synchronous Orbit (SSO) together with CHEOPS from Kourou on a Soyuz launcher operated by Arianespace. The platform consists of several advanced payloads including a fine ADCS (Attitude Determination and Control System), X-band transmitter, optical receiver and Software Defined Radio payload. The heart of the satellite is the Satellite Experimental Processing Platform (SEPP), which consists of two Cyclone V System-on-Chip (SoC) units running Embedded Linux. A normal stack of embedded software applications is available on the SEPP, ranging from Java and Python 3, down to running low-level C applications together with an Application Programming Interface (API) to interact with the various payloads [1]. Experimenters coming from European academia and Industry write proposals for experiments on OPS-SAT and get selected after which the development process begins [2]. The purpose of OPS-SAT is to break the 'has not flown will not fly' boundary present in space operations. The spacecraft is illustrated in Figure 1.

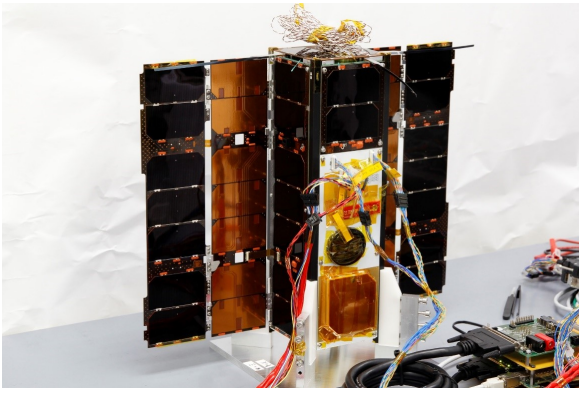

Fig. 1. OPS-SAT with deployed solar arrays in the cleanroom (TU Graz)

The spacecraft bus, payloads and structure are designed in a way such that experiments can be carried out safely. The latter includes the addition of power- and databus switches in order to completely isolate a unit from the satellite internally. In addition, the On-board Software (OBSW) used to run the satellite bus and process telecommands (TC) and send telemetry (TM) is running on a completely separate dedicated unit. OPS-SAT is controlled from ESOC (the European Space Operations Centre), from which also all the experiments are coordinated. Commissioning of the satellite was completed in late 2020 and the mission is currently in the operational phase having already realised various experiments for several partners including the French space agency CNES [3]. While the Mission is already using GNU Radio routinely in the ground segment [4], this paper covers the first experiment that uses the SDR-payload together with in-orbit RF signal processing using GNU Radio. The next paragraph introduces the specific use case of the experiment.

T. Mladenov was with the European Space Agency, European Space Operations Centre ESOC Darmstadt, Germany
Manuscript received July 12th, 2021

II. Method

To demonstrate the use of the SDR-payload of OPS-SAT with a real-life scenario, the COSPAS-SARSAT system was chosen. This is a global network of ground stations and transponder instruments on satellites which relay distress transmissions originating from emergency beacons. The system allows detection and relay of emergency transmissions via satellites which, depending on the orbit, can cover a very large area. Ships and aircraft larger than a certain size are required to carry such a beacon, also known as an Emergency Position Indicating Radio Beacon (EPIRB). The aviation equivalent is called an Emergency Location Transmitter (ELT), often mounted in the tail of the aircraft. Both beacon types are however identical in terms of operation. Distress beacons can be activated by a variety of means including prolonged contact with salt water for an EPIRB and very large G-forces for the ELTs. Manual activation is also always possible. Several protocols exist for the actual message encoding, but most schemes encode basic information such as a registration number and approximate location acquired via an internal GNSS receiver. Many satellites in Low, Medium and Geostationary orbits (LEO, MEO and GEO) carry repeater instruments which perform a frequency shift of the received beacon transmissions in UHF, and downlink it in the L-band (1.5 GHz) to the Local User Terminals (LUTs) for on-ground processing of the beacons. Once a beacon activation is verified, it is forwarded to the nearest Rescue Coordination Centre (RCC). If no GPS information is sent in the beacon, Time Difference of Arrival (TDOA) is used to determine a coarse position. The physical appearance of an ELT and EPIRB are illustrated in Figure 2.

Fig. 2. Artistic rendering of activated EPIRB and ELT beacons in a distress scenario [5]

As opposed to the on-ground processing that is employed in the operational COSPAS-SARSAT system, this IOD is concerned with the direct in-orbit processing and decoding of beacon transmissions. The SDR-payload of OPS-SAT together with the UHF monopole antenna are used to receive the terrestrial RF transmissions from the beacons in the 406 MHz band. In-phase and Quadrature (IQ) samples representing the real and imaginary components of the complex RF signal are acquired and stored in the on-board filesystem. The samples are then read by a C++ burst decoder based on GNU Radio. Two types of files are generated, json and IQ-files. The json-products contain decoded metadata from the transmissions such as encoded message, burst frequency and errors while the decimated IQ-files contain the raw transmission for offline analysis. The products are downloaded later via S-band when in contact with Mission Control at ESOC in Darmstadt, Germany. The raw RF input files that were originally captured are automatically deleted since they are no longer required. This process is illustrated in Figure 3.

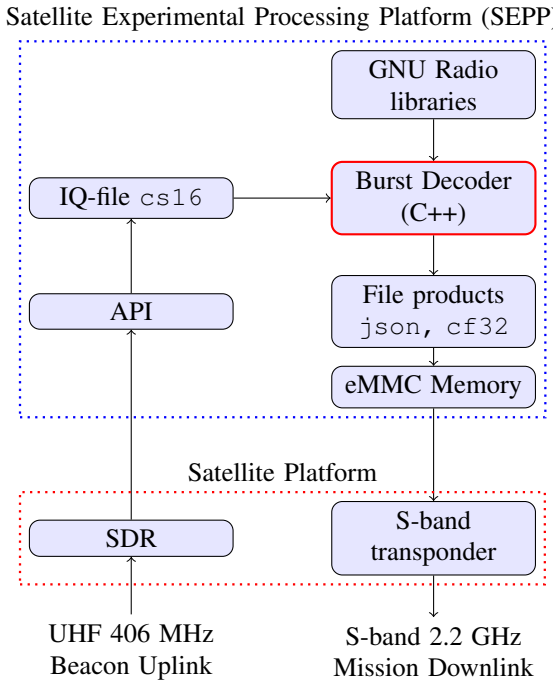

Fig. 3. On-board data flow overview

The next paragraphs will elaborate further on the technical details of the beacon transmitter standards and how these were used to test and design a prototype decoder that was deployed on the spacecraft.

III. Beacon Transmission Characteristics

Modern EPIRB/ELT transmissions occur in the protected 406 MHz band and are phase modulated at 400 bits per second with a phase offset of 1.1 Radians. Manchester encoding is used to encode the bitstream, resulting in a modulation rate of 800 Baud. This encoding is used to ensure there are periodic transitions present in case of many identical bits. The periodic transitions in turn allow for a better symbol clock recovery. This is illustrated in Figure 4. Due to the Biphase nature of the linecode, a residual carrier will be present in the transmitted signal. A residual carrier in this case is desirable as it allows for precise frequency offset determination for coarse position estimation.

A typical transmission is radiated with a power of 37 dBm or 5 Watts into a monopole antenna and consists of several periods. Firstly an unmodulated carrier is sent for 160ms. A bit synchronisation pattern follows, consisting of 15 '1's,

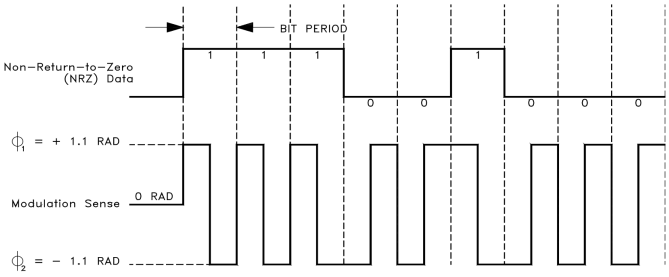

Fig. 4. Manchester encoding of the data bits, resulting in 2 possible phases of the carrier (-1.1 and +1.1 Rad) [6]

resulting in 30 Manchester symbols. The next 9 bits form the frame sync of the transmission and can have 2 values depending on the activation type. The frame sync will consist of the sequence `000101111` for a regular transmission and `011010000` for a transmission in self-test mode. When a user performs a monthly test of the beacon to check functionality, the transmission will use the self-test frame sync in order to not signal a genuine distress to the authorities. If the beacon however is activated in an emergency, the normal frame sync will be used. After the frame synchronisation, a format bit follows indicating if the message is a long or a short message. Finally the data field is transmitted, which has a length of 87 bits for a short message and 119 bits for a long message [6]. The full sequence is illustrated in Figure 5.

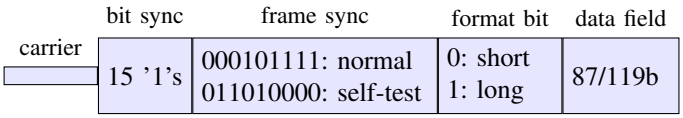

Fig. 5. COSPAS-SARSAT beacon transmission according to C/S T.001 [6]

IV. Burst Decoder Implementation

The burst decoder (highlighted red in Figure 3) processes the captured raw RF samples and performs demodulation of any beacons present in the recording. While the modulation and framing is known, the time of the transmission as well as the exact frequency are not. The following requirements are defined for the burst decoder:

- IQ-sample input in format cs16
- Logging of burst frequency
- Logging of time-of-burst (TOB)
- Preamble detection
- Manchester decoding and output in json format

The burst decoder is a Linux executable that can be invoked from commandline with arguments to perform the desired signal processing steps on an IQ input file. The executable uses GNU Radio shared libraries that are dynamically linked during compilation as well as runtime. The necessary GNU Radio signal processing blocks are imported via the C++ header files in accordance with the API [7] and connected together forming the final flowgraph. The burst decoder can be split up in 3 main steps:

1) Sample Preprocessor
2) Phase Lock & Demodulator
3) Symbol Timing Recovery & Decoding

The following paragraphs cover the details of each processing step as well as the design choices. The complete overview of the different stages in the burst decoder is given in Figure 6.

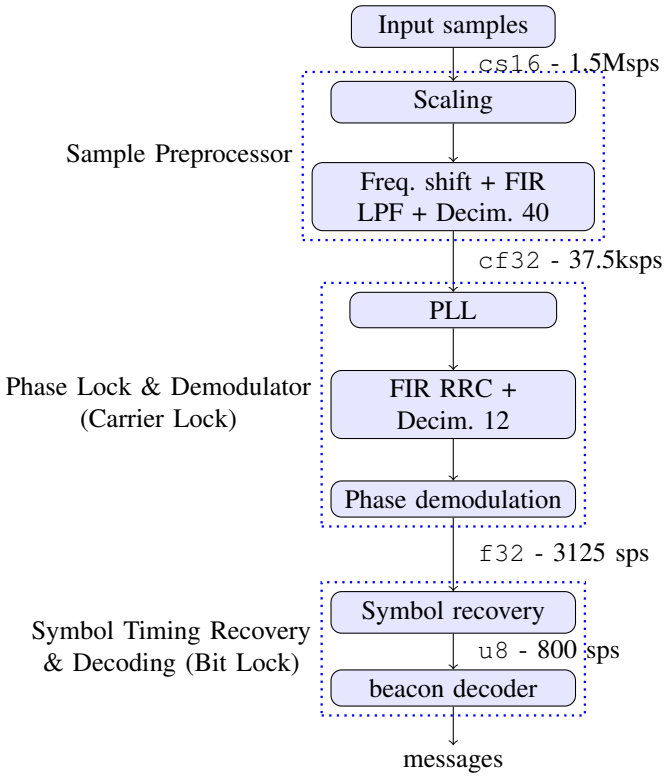

Fig. 6. Burst decoder flowgraph showing the 3 main signal processing stages

A. Sample Preprocessor

The sample preprocessor is used to filter out the frequency range of interest as well as lower the sampling rate. The SDR interface on OPS-SAT currently does not support streaming of IQ-samples, and therefore measurements have to be logged to disk and then read. The 12-bit I and Q samples acquired from the SDR are padded to 16 bits and stored as files with the GNU Radio `cs16` format. The latter indicates that the samples are stored as 16-bit signed integers. To use this format as source in the GNU Radio flowgraph, an `interleaved_short_to_complex` block is used to convert the samples to the `cf32` format. This format is commonly used in GNU Radio and stores the I and Q samples as 4 byte IEEE-754 floating point values. Finally, to adjust for scaling offsets, the I and Q samples are multiplied with a constant that ensures the full-scale 12 bit signed IQ input values map to -1.0 and 1.0 when converted to `cf32` format.

$$\vec{S}_{CF32} = \vec{S}_{CS16} * \frac{1}{\frac{2^{12}}{2} - 1} = \vec{S}_{CS16} * \frac{1}{2047} \quad (1)$$

An IQ imbalance is present in the RF Front End causing a DC-offset as indicated in Figure 7 by a carrier at 0 Hz. To ensure the signals of interest appear in a spurious free part of the baseband, the SDR center frequency is tuned to a lower frequency. The Carrier Frequency Offset (CFO) is chosen such that the target passband still appears within the baseband so that it can be frequency shifted. This is shown in Figure 7a where the blue area indicates the target passband. A CFO of -125 kHz was selected such that the actual tuned SDR Center Frequency becomes 406.025 MHz - 125 kHz = 405.9 MHz. The GNU Radio block `freq_xlating_fir_filter_ccf` is used to revert the frequency shift of 125 kHz to bring the passband back to baseband which is illustrated in Figure 7b. The resulting signal is then low-pass filtered using a Finite Impulse Response (FIR) filter with a cutoff frequency equal to the Nyquist frequency of the new samplerate as a decimation is also performed afterwards to reduce the samplerate from F_{s1} to F_{s2}. The low-pass filter in this case is functioning as an anti-aliasing filter. The final settings of the preprocessor are given in Table I.

TABLE I
SAMPLE PREPROCESSOR SETTINGS

Name	Value
Input samplerate (F_{s1})	1.5 Msps
Output samplerate (F_{s2})	37.5 ksps
Carrier Frequency Offset	-125 kHz
LPF Cutoff	15 kHz
Decimation	40

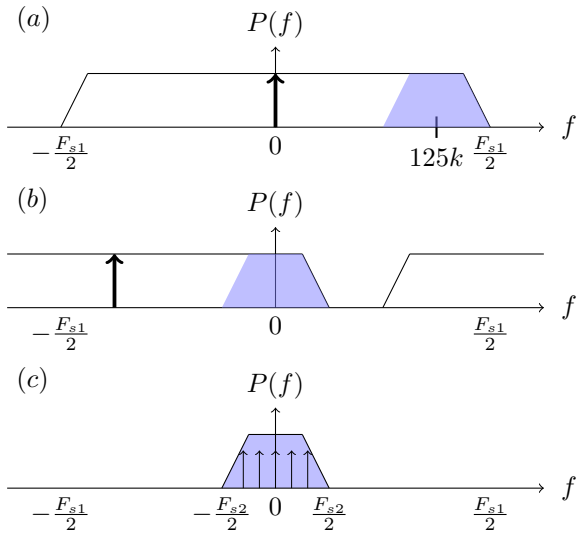

Fig. 7. The various sample processing stages showing the target passband highlighted in blue (a), the frequency shifting (b) and finally the lowpass filtering and decimation (c). The frequencies shown are baseband frequencies, to obtain the RF carrier frequency, the configured SDR center frequency of 405.9 MHz has to be added.

The resulting preprocessed stream of complex samples with reduced samplerate is also stored in a file on-board for later downlink in case a burst is detected.

B. Phase Lock & Demodulator

Due to the high relative velocity of the spacecraft towards the ground there will be a significant Doppler shift that needs to be taken into account. In order to synchronize to the burst transmissions, a Phase Locked Loop (PLL) is used to lock on the initial carrier of the beacon and dowconvert the burst transmission to baseband for further processing. More specifically, the GNU Radio block `pll_carriertracking_cc` is used to perform this operation. The phase-locked and downconverted burst transmission is filtered with a secondary FIR filter which further decimates the incoming stream to a rate that is closer to the actual symbol rate. An FIR filter with Root Raised Cosine taps is used. The majority of phase information will reside in the Quadrature (Q) component of the complex signal, while the In-phase (I) component mostly contains the power of the residual carrier due to the Biphase-L encoding of the bitstream. The contents of the I and Q branch after PLL lock and downconversion are illustrated in Figure 8. The PLL locking can clearly be seen as only the Real part of the signal (the I or in-phase part) contains power while the Imaginary part does not. Constant IQ-values also indicate a phasor with a fixed phase and amplitude, and hence is expected after the PLL obtains a lock. As soon as the phase modulation of the preamble begins, the power swings back and forth between the positive and negative Imaginary components following the 2 Manchester phase levels according to the Biphase-L encoding scheme. Only the samples from the PLL Q-branch are passed down further as this data contains the phase, and therefore the encoded message.

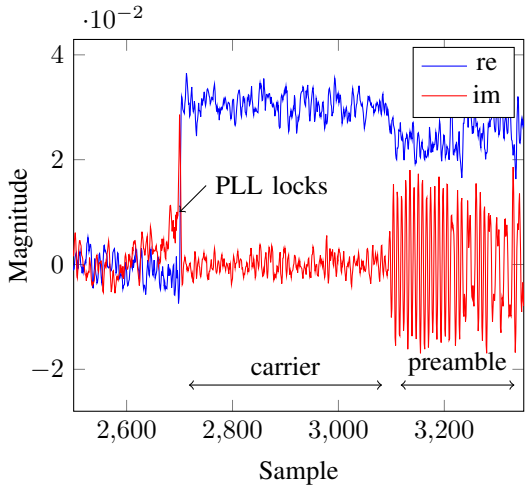

Fig. 8. I and Q samples of a detected burst, showing the initial carrier lock at sample 2700 and start of synchronisation symbols at sample 3100

C. Symbol Timing Recovery & Decoding

Symbol recovery is done by feeding phase symbols to a `clock_recovery_mm` block which performs symbol timing

recovery according to the Mueller and Müller Timing Synchronisation Algorithm [8]. This GNU Radio block is provisioned with the estimated symbolrate of 800 Symbols/second. The latter is an estimated rate since the sampling rate of the SDR is not exact. After symbol recovery, the output are also phase symbols in `f32` format, but resampled to 1 sample/symbol, hereafter called 'soft symbols'. The soft symbols are converted to hard symbols by means of a binary slicer. This block converts any value equal or less than 0 to a hard 0, and any positive value to a 1. The conversion from soft symbols to hard symbols in the preamble sequence is illustrated in figure 9.

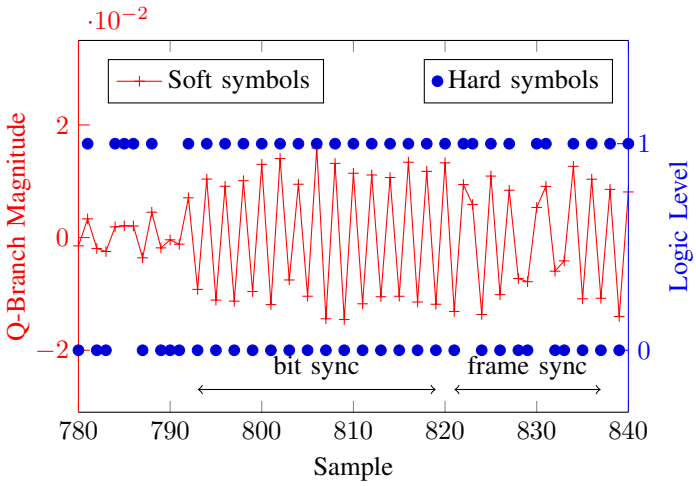

Fig. 9. Q-branch sample plot showing the resampled phase symbols at 1 sample/symbol and conversion from soft symbols (red) to hard symbols (blue)

The obtained logic levels of the hard symbols are still representing the Manchester symbols and not the actual bits. A differential decoder is used to map the Manchester symbol pairs to bits. The bit synchronisation pattern consisting of 15 '1's (30 Manchester symbols) can be seen in Figure 9. The decoding is performed in the C++ code and is implemented as a tag correlator where the hard symbols are shifted continuously through a synchronisation register. This register is checked at every bit shift how well it correlates with the expected preamble sequences. As the phase information is relative to the carrier that the PLL locks to, the absolute phase of the transmission is not known. Therefore it is also necessary to correlate against the inverted preamble, resulting in 4 possibilities:

- (15x `1`) + `000101111` (normal polarity, normal mode)
- (15x `0`) + `111010000` (inverted polarity, normal mode)
- (15x `1`) + `011010000` (normal polarity, self-test mode)
- (15x `0`) + `100101111` (inverted polarity, self-test mode)

If there is a match between the contents of the synchronisation register and one of the four possible preambles, then a transmission is detected after which the subsequent symbols are shifted into a separate message register. Once the message register is filled, it is differentially decoded and the output logged.

V. Deployment

Software uplink to OPS-SAT is handled via ipk-files and are managed on-board by the open source package manager `opkg`. This ensures easy tracking of installed files as well as version management of existing packages and upgrading them. An ipk-file is a compressed archive containing 2 further archives separating data from control information. The application files and folders are packaged in a `data.tar.gz` file while the control information such as version, dependencies and description are stored in a `control.tar.gz` file. GNU Radio v3.7.13.4 in a headless configuration is cross-compiled for the arm-linux-gnueabihf architecture and packaged in separate ipk-files. The archives containing the shared GNU Radio libraries are available to other future users as well since they are placed in `/usr/local/lib` after installation on the satellite. IOD specific shared libraries are packed together with the main ipk-file for the experiment.

VI. Orbit Test Campaign and Results

The COSPAS-SARSAT system comprises of a worldwide network of beacon reference stations. These ground stations transmit periodic bursts with a fixed frequency and repetition rate. The reference beacons are used to monitor the system performance since the geographical location of the emitters is known. The reference beacons are used to test the on-board implementation since the message they encode is known in advance and can be cross-correlated afterwards. For the orbital test campaign, four stations were selected due to their high transmission cadence of 30 seconds, i.e. a beacon is transmitted every 30 seconds. The live operational status and additional details of each station can be consulted on-line [9]. The chosen reference sites are given in table II and graphically represented in Figure 10.

TABLE II
Reference stations used in the test campaign

Site	Latitude (°)	Longitude (°)	Station/Beacon ID
Longyearbyen	78.228983	15.3955	A0234BF8A7335D0
Thule	77.46475	-69.217217	9B62197CA703500
Kerguelen	-49.3515	70.256	9C7FEC2AACD3590
McMurdo	-77.846033	166.71178	ADC389B8AA1B9D0

The ground station footprints for a satellite in LEO at 515 km orbital height are also indicated in light blue on the chart. If the satellite ground track is within this footprint, there is Line-of-sight (LOS) to the transmitter which is a requirement for communications in the UHF-band. The reference stations are located at extreme latitudes in order to increase the transmitter visibility to satellites in polar orbit, and hence will be in LOS with the beacon for most of the orbits. In coordination with Mission Planning, the SDR application is started when the satellite is about to enter the visibility footprint of the various stations. The experiment is started via previously uplinked time-tagged telecommands stored in the On-board queue (OBQ). Once started, the system will listen for beacon transmissions for 9 minutes and terminate afterwards. A total of 11 passes were scheduled of which 6 decoded 1 or more

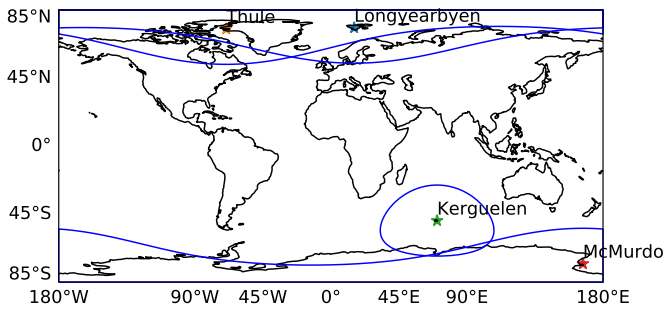

Fig. 10. Geographical location of the used test ground stations including the coverage area indicated in blue.

beacon transmissions. This success rate is mainly due to the dead time of the SDR in between sample capturing runs since the payload interface does not yet support sample streaming. All decoded beacons are given in Table III. Synchronisation error count indicates the number of incorrect symbols detected in the preamble. The decode error shows the number of occurrences where there are 3 or more subsequent identical symbols. In Manchester encoding only 2 subsequent identical symbols can occur. Possible causes for a good synchronisation but high decode error count is loss of PLL lock due to signal power fading.

The metadata for the highlighted beacon in table III as generated on board in a json product is illustrated in Listing 1. The full beacon message is contained in the `msg_bits` field. Since the most significant bit of the message indicates the message length, the hexadecimal sequence needs to be left shifted by 1 bit, after which the message matches the station ID for the transmitter at McMurdo given in table II.

Upon detection of a beacon, the decimated IQ output file after the preprocessor is stored on board for downlink to allow for further analysis on the ground. The filename of this file is also stored in the json product under the `filename` key to allow tracing of beacon detections across the various artifacts. The decimated IQ files are in the `cf32` format and can be viewed directly in GNU Radio as well. Figure 11 illustrates the Fast Fourier Transform (FFT) of the burst transmission

```
{
  "beacon": {
    "filename":
    ↪  "sdr_iq_20210615_043251_405900000_
    1500000_15_66_37500sps.cf32",
    "freq_hz": 406035424,
    "payload": {
      "errors_n": 0,
      "msg_bits": "56E1C4DC550DCE801D130F",
      "msg_len": 88,
      "symbols_s":
      ↪  "6669A956A565A6A5666655A6A5A9955556A
      6565A55AABB49C33D6B3691C3"
    },
    "sample_n": 846,
    "sync": {
      "error_n": 0,
      "inverted_b": true,
      "type_s": "beacon_normal_inverted"
    },
    "time_s": 1.0575000047683716
  }
}
```

Listing 1: Contents of the json product for the highlighted beacon in table III

contained in the IQ-file that was downloaded from the satellite showing the residual carrier and the phase modulation.

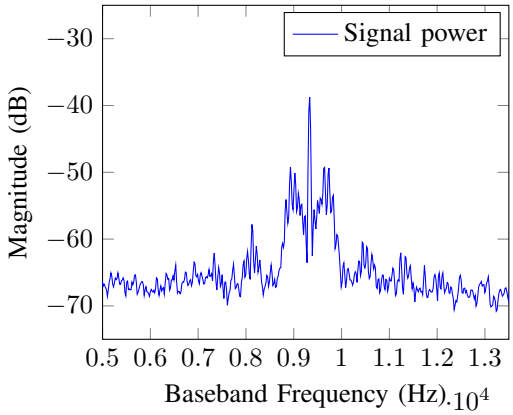

Fig. 11. Averaged FFT of the received beacon uplink, processed from the downloaded IQ-file from the satellite

TABLE III
DECODED SAR MESSAGES DURING ORBIT TEST CAMPAIGN

UTC Time	Burst Frequency (Hz)	Message (HEX)	Sync err	Decode err	Station LOS
2021-06-15 00:42:25	406013952	1411A5FC5399AE857D5D8002002C01	1	21	Thule + Longyearbyen
2021-06-15 01:37:50	406030144	4E3FF6155669AC86E79580	0	6	Kerguelen
2021-06-15 01:39:20	406027552	4E3FF6155669AC86E79580	0	0	Kerguelen
2021-06-15 02:13:11	406029632	1411A9FC5381AE857D5D80	2	5	Thule + Longyearbyen
2021-06-15 02:14:42	406021984	5011A5FC5399AE857D5D80	0	0	Thule + Longyearbyen
2021-06-15 02:15:21	406028288	10215F14B1878C81A4204E5A2ECAF0	0	5	Thule + Longyearbyen
2021-06-15 02:17:18	406028384	4DB10CBF08FC07F800223E	0	22	Thule + Longyearbyen
2021-06-15 03:12:49	406023424	4EBFF61556692C86E795A0	0	1	Kerguelen
2021-06-15 04:32:53	**406035424**	**56E1C4DC550DCE801D130F**	**0**	**0**	**McMurdo**
2021-06-15 09:19:53	406028576	56E104DC550D0E801D130F	0	5	McMurdo
2021-06-15 09:24:25	406029248	D66134FC3068C086A0478000000000	2	12	McMurdo

VII. Conclusion

We demonstrate the development of a GNU Radio application and the porting of it to an operational environment for on-board deployment on OPS-SAT. In-orbit processing of the RF-samples significantly reduced the amount of data that needed to be downlinked, in turn increasing the Mission efficiency. While the major limitation is currently the interface to the SDR, resulting in non-continuous sample capturing, the results are nevertheless encouraging. The successful IOD activity indicated that the RF frontend on OPS-SAT is suitable for experiments in the 406 MHz Search and Rescue band. This experiment has also shown the capability of OPS-SAT to serve as a flexible platform for SDR experimentation for applications ranging from Remote Sensing to LoRa and other low-power internet of things (IoT) applications, bridging the gap between satellite operations and the IoT era. A multitude of experiments have already been submitted that aim to utilise this new in-orbit infrastructure for digital signal processing and RF experimentation in Space.

References

[1] Marszk, D., Feiteirinha, J.L., Fischer, B., Taubert, D., Graber, T., Lofaldli, A., Sarkarati, M., Evans, D., and Merri, M., "OPS-SAT – opening a satellite to the internet." in *Proc. of The 12th IAA Symposium on Small Satellites.* International Academy of Astronautics (IAA), 2019.

[2] Evans, D., Marszk, D., Mladenov, T., Labrèche, G., Zelenevskiy, V. and Shiradhonkar, V., "OPS-SAT – The World's First Satellite Accessible by the Public Over the Internet," in *Proc. of the 16th International Conference on Space Operations, Cape Town, South Africa – 3 - 5 May 2021.* International Astronautical Federation (IAF), 2021.

[3] European Space Agency. (2021) Interplanetary internet & cameras in space: ESA's OPS-SAT first results. [Online]. Available: https://www.esa.int/Enabling_Support/Operations/Interplanetary_internet_cameras_in_space_ESA_s_OPS-SAT_first_results

[4] Mladenov, T., Fischer, B. and Evans, D., "ESA's OPS-SAT Mission: Powered by GNU Radio," *Proceedings of the GNU Radio Conference*, vol. 5, no. 1, 2020. [Online]. Available: https://pubs.gnuradio.org/index.php/grcon/article/view/65

[5] International Cospas-Sarsat Programme. (2021) Introduction to the Cospas-Sarsat System. [Online]. Available: https://cospas-sarsat.int/images/stories/SystemDocs/Current/G003-NOV-2019.pdf

[6] International Cospas-Sarsat Programme. (2021) Specification for COSPAS-SARSAT 406 MHz Distress Beacons, C/S T.001 Issue 4 – Revision 8, June 2021. [Online]. Available: https://cospas-sarsat.int/images/stories/SystemDocs/Current/T001-JUN-7-2021.pdf

[7] GNU Radio project. (2021) GNU Radio Manual and C++ API Reference v3.7.13.4. [Online]. Available: https://www.gnuradio.org/doc/doxygen-v3.7.13.4/

[8] K. Mueller and M. Muller, "Timing recovery in digital synchronous data receivers," *IEEE Transactions on Communications*, vol. 24, no. 5, pp. 516–531, 1976.

[9] International Cospas-Sarsat Programme. (2021) MEOSAR Reference Beacon Status. [Online]. Available: https://cospas-sarsat.int/en/system/meosar-system-status/meosar-reference-beacon-status