LeoCommon - A Ground Station Observatory Network for LEO Satellite Research

Eric Jedermann
RPTU Kaiserslautern-Landau, Germany
jedermann@cs.uni-kl.de

Martin Böh
RPTU Kaiserslautern-Landau, Germany
m_boeh16@cs.uni-kl.de

Martin Strohmeier
armasuisse, Switzerland
martin.strohmeier@armasuisse.ch

Vincent Lenders
armasuisse, Switzerland
vincent.lenders@armasuisse.ch

Jens Schmitt
RPTU Kaiserslautern-Landau, Germany
jschmitt@cs.uni-kl.de

***Abstract*—Low Earth Orbit (LEO) satellites are becoming increasingly popular with private companies launching them to build vast networks that cover the globe. As these satellite systems expand, questions about their performance, security, and privacy are rising. To address these questions, researchers need to study these systems in real-world conditions. To support this kind of empirical research, we developed LeoCommon, an experimental network of ground stations. This network is designed to work with multiple satellite constellations such as Iridium, Globalstar, Starlink, and others. The LeoCommon system only uses open-source software and affordable hardware components that are easily accessible to academic researchers. We set up an initial network of ground stations in Central Europe, consisting of 10 stations. Using this setup, we have managed to collect over 500 synchronized recordings from the Iridium satellites, totaling more than 3,400 hours of data. This paper discusses the design of LeoCommon, our experiences in setting up the stations, and the initial results from testing the system with the Iridium network constellation.**

I. Introduction

The field of satellite communication received significantly growing public and academic attention over the past few years. The development of new satellite systems, particularly in low-Earth orbit (LEO) such as Starlink, OneWeb, and Kuiper, has injected new momentum into satellite communication. In addition to these new players, long-established LEO systems such as Iridium and Globalstar continue to be widely used.

As these LEO satellite systems and their applications are expanding, questions about performance, security, and privacy are rising. To address these questions, researchers have started experimenting with these systems from various angles. However, as LEO satellites are moving quickly relative to the Earth and their communication beams can be quite narrow, experiments with LEO satellites can be a complex endeavor.

To facilitate data collection, several crowdsourcing initiatives have emerged over the years. The SatNOGS network [2, 19] was founded in 2014 aiming to develop and push open technologies for satellite ground stations. Connected ground stations become a shared resource in a global network of amateur radio ground stations. In 2015, the Electrosense [15] network emerged as a distributed system to monitor the wide-band spectrum and decode arbitrary radio frequency signals. In 2019, the TinyGS[1] crowdsourcing network was created to collect LORA IoT signals from CubeSats.

While these systems provide useful satellite data collection capabilities, they lack features that are often necessary for experimental research with LEO systems. For example, these networks heavily rely on the RTL-SDR USB dongle as a radio receiver frontend, which severely limits the frequencies and the bandwidth of the signals that can be observed. Furthermore, these systems are not designed for high-quality measurements as needed in research but mainly aim to decode signals opportunistically. Finally, they do not allow for time-synchronized measurements across different ground stations, a feature that is often needed in research to compare recordings and for signal localization purposes.

To support the research in this field, we developed LeoCommon, a network of common ground stations. LeoCommon is designed to observe data communication from various satellite constellations such as Iridium, Globalstar, Starlink, and others, allowing researchers to run measurement campaigns in various frequency bands, using different modulations and encodings. LeoCommon supports distributed, synchronized, and high-quality measurements using open-source software and commercially available hardware components. This makes the system suitable and affordable for researchers.

Currently, we are running an instance of the network consisting of 10 ground stations distributed across several European countries. We have successfully tested the system for collecting data from the Iridium constellation. For instance, we can record all Iridium messages and collect raw samples of Iridium Ring Alert (IRA) headers. We are further extending our system to support a larger variety of hardware and cover more satellite communication systems.

[1] https://tinygs.com/

Workshop on Security of Space and Satellite Systems (SpaceSec) 2025
24 February 2025, San Diego, CA, USA
ISBN 979-8-9919276-1-1
https://dx.doi.org/10.14722/spacesec.2025.23027
www.ndss-symposium.org

As the research interest in the satellite domain is growing, we have decided to develop and provide a software solution solely based on open-source software and hope to grow the community around LeoCommon. The code for the project with setup manual and extensive documentation is available on GitHub: https://github.com/LeoCommon.

II. Goal and Resulting Requirements

The goal of LeoCommmon is to provide researchers with an easy-to-deploy and use platform to perform distributed LEO satellite communication research. For example, researchers leveraged empirical satellite data to devise and test a GNSS spoofing detection system [12], to fingerprint LEO satellites [13, 17, 21], and to assess the security as well as privacy threats of terminal users [5]. More research efforts are currently ongoing worldwide, and we expect a significant rise in studies in the future.

To support these research efforts, the goal of this work is to design a ground station observatory network with the following properties:

- Multi-constellation: We aim to support multiple interesting LEO constellations, shown in Table I. In this way, we can support the applicability of new approaches in research on a variety of satellite systems such as [4, 9].
- Distributed and synchronized measurements: LEO satellites are more dynamic compared to GEO satellites since they have a smaller antenna footprint and wander across the sky. So distributed measurements can provide valuable insights into a system, as shown by [7, 20]. It is also important to capture the time of a measurement, to recreate the conditions of the satellite constellation afterwards.
- Decoded and raw signals: Depending on the field of research, the recordings have to meet different requirements: for signal fingerprinting [13, 17, 21] raw recordings are used, while for traffic content focused research [5, 7, 12] decoded messages are utilized.
- High-quality data: In many research applications, reliable signal recordings with good SNR are necessary for their methods to work, as [5, 13, 17] explain.
- Open source and low cost: In setups with multiple ground stations, as [8, 20], a cost-efficient solution that can be adapted to the specific requirements is preferred.

III. System Architecture

A high-level overview of the network architecture and its major components is shown in Figure 1. A central server is responsible for controlling the ground stations (shown as satellite dishes). The stations regularly pull tasks from the server, execute them, and upload the results. A researcher can use the web interface to schedule new tasks for the ground stations, view and download results from previous experiments, and configure the ground station.

In the following subsections, we describe the ground station and server software architecture.

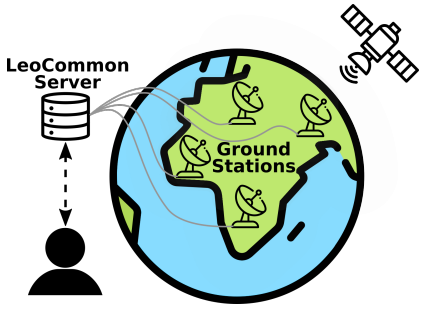


Fig. 1: High-level overview of the network.

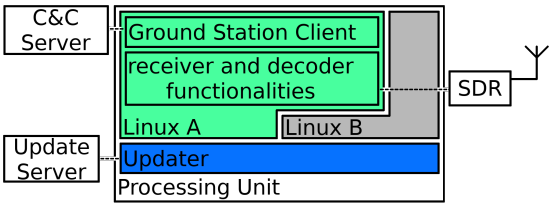


Fig. 2: Overview of the ground station systems architecture.

A. Ground Station Architecture

The operating system consists of three major parts shown in Figure 2: two independent Linux instances (Linux A and B, in green and gray) and an underlying updater (in blue). The two available Linux instances operate independently. During ground station operation, only one instance is active at a time. For example, as shown in Figure 2, Linux A is booted while Linux B remains inactive. This design allows the updater to perform updates on the inactive instance (Linux B) without disrupting the operation of the active system. Once the update is complete, the active instance (Linux A) is notified to schedule a reboot. Upon reboot, the bootloader switches to the updated instance (Linux B).

If the updated instance works correctly, it is marked as "healthy" and becomes the new default for subsequent boots. In the event of a permanent failure with the new instance, the system automatically reboots and reverts to the previously op-

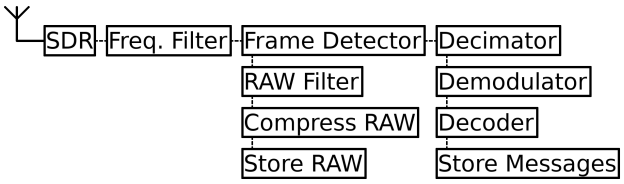


Fig. 3: Signal processing flow inside the ground station.

System	Satellites	Forward-Downlink Freq.	Return-Uplink Freq.
Iridium	66 LEO	1620 MHz	1620 MHz
Globalstar	48 LEO	2490 MHz	1615 MHz
Orbcomm	36 LEO	137 MHz	148 MHz
Starlink	+6,700 LEO	12 GHz	26 GHz
OneWeb	+600 LEO	12 GHz	26 GHz
Kuiper	3200 LEO	12 GHz	26 GHz

TABLE I: Interesting LEO communication systems.

Platform	CPU		RAM	CPU Mark		
	cores	Clock Rate [GHz]	[GB]	Score	Floating Point Math [MFLOPS]	ARM NEON Extended Instructions [10^6Matrices/s]
BeagleBone Black*	1x ARM Cortex-A8	1.0	0.5	-	73.7	-
Raspberry Pi 3B+	4x ARM Cortex-A53	1.4	1	198.6	1289.7	164.1
Rock Pi 4B+	2x ARM Cortex-A72 4x ARM Cortex-A53	1.8 1.4	4	997.5	3513.7	469.1
Raspberry Pi 4B	4x ARM Cortex-A72	1.5	4	515.6	3777.3	649.5
Odroid-XU4*	4x ARM Cortex-A15 4x ARM Cortex-A7	2.1 1.4	2	1274.6	2867.8	824.6
Odroid-N2+	4x ARM Cortex-A73 4x ARM Cortex-A53	2.4 2.0	4	1476.4	6437.7	953.1
Raspberry Pi 5B	4x ARM Cortex-A76	2.4	8	1813	10559	1609
Raspberry Pi 5B (w. Fan)	4x ARM Cortex-A76	2.4	8	2177	10470	2034

TABLE II: Results of PassMark PerformanceTest on ARMv8 (64b). Platforms marked with * only support ARMv7 (32b).

erational instance (Linux A) if three consequent boot or health check attempts fail. This approach ensures seamless updates while maintaining a highly reliable operational environment.

The update client communicates with a dedicated update server that operates independently of the command and control (C&C) server. This ensures the separation of concerns and enhances system robustness.

The components related to information flow and signal processing within the ground station are shown in Figure 3.

Once a signal is recorded, using the SDR, it is filtered to the frequency band of interest. A frame detector analyzes the incoming signal to identify parts that contain message transmissions. Depending on the requirements, the following signal processing chain can vary: One possibility is to store the raw signal. For this case, the raw frames are filtered for the important frames of interest (e.g., only header frames). After this, compression is applied to the frames of interest before they are stored [3, 11]. In case the message content is of importance, a decimator is used to trim the signal to the required sample rate after the frame detection stage. This is followed by demodulation and decoding which extract the transmitted symbols and bits. Different receivers, like an Iridium message receiver, an Iridium Ring Alert (IRA) header raw recorder, or a Globalstar message decoder, are implemented by using a combination of the provided functionalities. Once a receiving task has finished its execution, the locally stored result is uploaded to the C&C server.

B. C&C Server Software

As previously mentioned, there are two servers in our network. One update server and one central command and control server. The update server is a separate component, not interacting with the C&C server.

The central command and control server is the core of the network. It waits for requests from the ground stations and allows users to access the functionalities of the network. We utilize a database to store all the data received from the ground stations, user accounts, and access credentials. On top of this, a REST API provides access to the data. A web server serves the static content and load balances requests for dynamic content to the backend. For security reasons, we ensure that the server is only accessible over an encrypted and authenticated TLS connection.

IV. System Design

When designing the ground station network, we had two major goals in mind: providing a cost-efficient solution for researchers and creating a system with high reliability. The first point is achieved by selecting appropriate off-the-shelf hardware, while the second goal is met by setting up a redundant, robust, and modular distributed software system.

A. Ground Station Hardware

We use modular components with different requirements: The radio frontend must be able to receive a frequency in which, ideally, as many satellite constellations as possible operate. During the design phase, we specifically considered the satellite systems, shown in Table I. In our network, we focus on Iridium and Globalstar reception. Other satellite systems such as Orbcom, Starlink, OneWeb, and Kuiper are future candidates for potential integration. While open-source decoders are available for certain constellations, such as Orbcomm, others, including Starlink [6], lack publicly available decoders and necessitate additional integration efforts.

1) Receiver Frontend: As potential receiver options, we evaluate three primary software-defined radios (SDRs). The RTL-SDR family, while cost-effective and widely used in ground station networks such as OpenSky and Electrosense, is limited to frequencies up to 1700 MHz with a bandwidth of 3.2 MHz, rendering it insufficient for our requirements. In contrast, the ADALM-PLUTO SDR supports reception up to 3800 MHz with a bandwidth of 20 MHz, while the HackRF One extends up to 6000 MHz at the same bandwidth. Both devices were viable choices; however, we opted to begin with the HackRF One due to its widespread adoption within the research community. Future work may explore the integration of other commonly used SDRs.

2) Processing Unit: Seven single-board computers were evaluated as potential processing units for running a ground station, including three variants of the Raspberry Pi, the BeagleBone Black, the Rock Pi 4B+, and two Odroid variants. Each device has a price below 100 Euro, ensuring cost efficiency and supporting network scalability.

To assess the raw CPU and memory performance of these devices, we conducted benchmarks using the "PerformanceTest" suite from PassMark[2]. Technical specifications and benchmarking results for the tested devices are provided in Table II. The benchmarks represent the average of 10 CPU and memory test iterations executed on each device with standard settings `./pt_linux_arm64 -i 10 -r 1`. The Linux PerformanceTest version used was v11.0 Build 1002. Furthermore, all Raspberry Pi devices used the ARMv8 (64-bit) Linux kernel 6.12.4 LTS. BeagleBone Black and Odroid-XU4 were evaluated only with ARMv7 (32-bit) systems, as their CPUs do not support 64-bit instructions. Benchmarks were carried out in a controlled environment with an ambient temperature of 21^oC to take thermal throttling after consecutive iterations into account.

This approach ensures uniformity and repeatability and provides a reliable comparison of computational performance across all devices. The measured results drive the selection process for the ground station's processing unit.

ARM NEON Single Instruction Multiple Data (SIMD) performance is of particular importance, as these extended instructions allow for optimized mathematical functions during signal processing tasks performed by GNU Radio (Volk) [18] and other system software like NumPy. Accordingly, Table II is sorted by performance in this category. The results demonstrate that the Odroid-N2+ and Raspberry Pi 5 perform best, whereas the BeagleBone Black and Raspberry Pi 3B+ exhibit the lowest performance metrics. Due to the BeagleBone Black's significantly worse results and the absence of 64-bit instruction support in both the BeagleBone Black and Odroid-XU4, these devices have been excluded from further consideration.

3) Supplementary Hardware: To operate a ground station, an Iridium antenna, the `Taoglas IMA.01.105`[3], a combined GPS-LTE-module `SIM7600E-H 4G HAT`[4], a microSD card and a USB stick are used. The Iridium antenna is a professional weather-proof passive outdoor Iridium antenna, tuned to the appropriate frequency range with a peak gain of 3.9 dBi. The GPS LTE module provides reliable location information and flexibility in ground station placement. The combination of a microSD card and a USB stick is chosen as a cheap alternative to platform-dependent SSD extension boards. Also, this provides the benefit of a clear separation of concerns: the microSD card only stores the operating system and bootloader, while the USB stick holds the ground station configuration, credentials, and recorded data.

B. Ground Station Software

The ground station software is an embedded Linux system developed with the help of Buildroot[5], a versatile tool designed for generating embedded Linux systems. The primary objective was to create a reliable, easily updateable and secure system. To achieve this, a custom Linux distribution was developed, incorporating only the essential software components. This approach minimizes the system's codebase, thereby reducing potential vulnerabilities while enhancing performance and maintainability.

1) Updates: To implement the Updater as outlined in Figure 2, the lightweight Robust Auto-Update Controller[6] (RAUC) is used. It seamlessly integrates with Buildroot and is used in conjunction with Hawkbit[7], an update management framework, to check for and deploy over-the-air (OTA) system updates. The fact that RAUC only allows the installation of signed updates provides an added layer of security implemented across all ground stations. Together, Buildroot, RAUC, and Hawkbit form a cohesive ecosystem, ensuring reliability, security, and ease of maintenance.

2) Main System: Each Linux instance is configured with only the essential software components required for operation. These include GNU Radio 3.10[8], receiver and decoder functionalities, and the ground station client. Additionally, lightweight services such as the GPS daemon (GPSD) and the time synchronization daemon (Chrony) are included. Startup and service management is handled by the `systemd` init system, while wired and wireless network connectivity is managed through `NetworkManager`. For cellular backup connectivity (4G, 5G, etc.), `ModemManager` is used to configure and manage supported modems. This streamlined approach focusing on already available software packages minimizes complexity and reduces the attack surface by limiting the number of exploitable components. Notably, an SSH server is intentionally excluded from the ground station to enhance security. Any necessary reconfiguration of the ground station can only be performed through tasks issued by the central server or via software updates delivered through RAUC.

Another key aspect of the system's reliability is that both Linux instances are read-only and cannot be modified on the fly. This is achieved by using EROFS (Enhanced Read-Only File System)[9], which is inherently read-only by design. This approach prevents microSD card writes, reducing the risk of corruption, a common issue in long-running devices due to the wear caused by frequent writes and unstable power conditions. To enable data storage, essential data like the ground station configuration, logs from the different modules, and recording job results are saved on a writeable USB Stick.

3) Receiver & Decoder Modules: The Iridium message decoder module is a slightly modified version of the open-source Iridium receiver software "gr-iridium"[16]. It observes a configurable part of the Iridium frequency band and demodulates the received messages. To provide reliable timestamps and avoid accumulation of timing errors we modified the timing calculation by using system calls, utilizing the system time. The effect and reason for accumulating timing errors are discussed later in Section V.

[2]https://www.passmark.com/products/pt_linux/index.php
[3]https://www.taoglas.com
[4]https://www.waveshare.com/wiki/SIM7600E-H_4G_HAT
[5]https://buildroot.org/
[6]https://rauc.io/
[7]https://eclipse.dev/hawkbit/
[8]https://www.gnuradio.org/
[9]https://docs.kernel.org/filesystems/erofs.html

Sample Rate [MSPS]	2	3	4	5	6	7	8	9	10
Raspberry Pi 3B+	0.74	64.45	73.50	76.11	82.69	84.34	90.87	91.89	92.57
RockPi 4B+	0	0.91	0.54	0	0	8.74	16.66	22.1	31.83
Raspberry Pi 4B	0.1	0.04	0.05	0	0	0.02	2.7	27.14	38.97
Odroid-XU4	0	0	0	0	0	0	0	3.81	13.29
Odroid-N2+	0	0.03	0	0	0	0	0	0	0
Raspberry Pi 5B (w. Fan)	0	0	0	0	0	0	0	0	0

TABLE III: Testing different software by measuring the amount of dropped frames [in %] when receiving satellite signals at different sample rates.

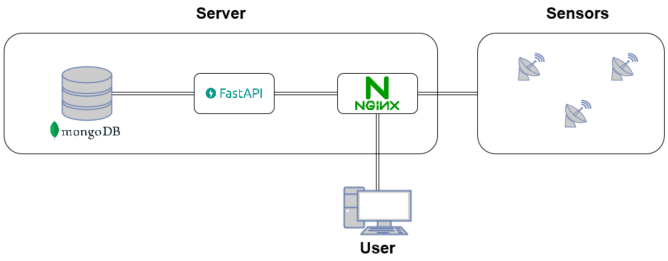


Fig. 4: Overview of the server design.

The raw IRA header recorder enables the capture of raw samples from Iridium Ring Alert headers. These messages, transmitted every 4.32 seconds by each Iridium satellite antenna, include the ID of the sending satellite and antenna. Such messages are frequently utilized in research on fingerprinting, as discussed in Section VII. The software implementing this functionality is derived from research conducted at the University of Oxford [17].

The Globalstar message decoder is currently under development and is further described in section VIII.

C. Server Software

For the update server, we utilize Hawkbit, which offers excellent integration with the RAUC updater of the ground station software, providing a reliable, secure, and user-friendly update mechanism. Since Hawkbit is a fully functional, out-of-the-box software module, further description is not necessary at this stage.

The central command and control server architecture is illustrated in Figure 4. MongoDB[10] is used for storing data and user accounts, selected for its flexibility and superior performance with large datasets, as highlighted by Parker et al. [14]. The REST API is implemented using FastAPI[11], chosen for its emphasis on scalability and high-performance connections. Additionally, a nginx[12] server runs on top, configured to accept only encrypted TLS connections.

User and ground station authentication is implemented using JSON Web Tokens[13], in compliance with the OAuth2.0 standard[14]. The system employs a combination of refresh and access tokens for authentication. The access token is used to authenticate communication with the server, and when it expires, the refresh token is used to obtain a new pair of access and refresh tokens. Each refresh token can be used only once, providing a balance between security and usability. Each ground station is configured with a unique access-refresh token pair stored on its USB stick.

Users initially log in with a username and password, and upon successful authentication, they are issued an access-refresh token pair. Once the access token expires, a background script automatically acquires a new access token using the refresh token. Consequently, all server requests, except for the initial login, are authenticated using the access token.

V. Capabilities and Receiver Quality

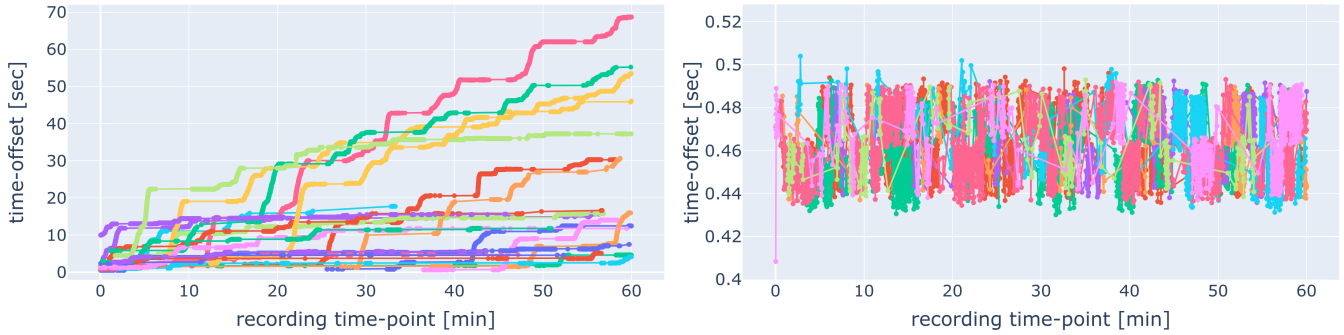
To assess the performance of different processing units, we evaluate their ability to receive Iridium messages. This involves using the Iridium message decoder connected to a HackRF One and measuring the percentage of dropped frames at varying sample rates. The results are presented in Table III, with values representing the average of three repetitions. To ensure consistency across measurements within each repetition, we pre-recorded the raw signals at the respective sample rates. These recorded signals were then replayed to each platform to maintain comparability in the measurements.

The evaluated sample rates begin at 2 MSPS, as the receiver software's minimum sample rate is 1.75 MSPS. Therefore, measurements below this threshold do not represent a realistic scenario.

When comparing the results, the Odroid-N2+ and Raspberry Pi 5 stand out, demonstrating excellent performance with no issues even at higher sample rates. The Odroid-XU4 and Raspberry Pi 4B follow in third and fourth place, respectively, performing well without any dropped frames up to 9 MSPS and 8 MSPS. These results offer valuable insight into the computational limits of the processing units within the scope of our analysis.

To assess the receiver performance in terms of timing, potential warnings for dropped frames are incorporated into the measurement results and uploaded to the server. This allows for a reliable evaluation of the quality of each reception.

[10]https://www.mongodb.com
[11]https://fastapi.tiangolo.com/
[12]http://nginx.org/
[13]https://jwt.io/
[14]https://oauth.net/2/



(a) Accumulating time offset with gr-iridium.

(b) Time offset of modified timing measurement, no accumulation.

Fig. 5: Timing offset of multiple 1 h recordings.

As shown in Table III, significant frame dropping occurs when using hardware close to its maximum computational capacity, but it can also happen sporadically at lower performance levels. In the original gr-iridium software, this causes issues in time calculations, as the receiving time of messages is derived from the frame counter. When frames are dropped, this leads to cumulative timing errors, as illustrated in Figure 5a. This figure displays the growing timing offsets over several 1-hour recordings, with the errors exceeding 60 seconds. The timing offset represents the difference between the timestamp of a recorded message from gr-iridium and a timestamp provided by the Iridium satellite through Iridium Broadcast messages. Such inaccurate receiving timestamps can pose challenges for research when the recorded messages are used. Given the bursty nature of Iridium signals, occasional frame dropping must be accounted for in reliable long-term observations. Our modified software addresses this issue by using system time for timing each individual frame, which is much more stable, as shown in Figure 5b. The remaining offsets are consistently within a 60 ms range and no longer accumulate. However, there is still a small amount of jitter to investigate in future work (Section VIII).

In addition to the timestamp, the data includes frequency and SNR estimations of the received message, along with meta-information such as dropped frames, the ground station's GPS location, and recording parameters like center frequency, sample rate, and receiver gain.

VI. Deployment and Coverage

We deployed a network of 10 ground stations across Europe, as shown in Figure 6. Since the network's launch in 2022, we have conducted over 500 recordings totaling more than 3400 hours, resulting in a data collection of 80 GB.

When deploying the ground stations, ensuring an unobstructed view of the sky is crucial to allow the reception of messages from a wide range of satellite positions.

The heatmap in Figure 7 illustrates the areas where an Iridium satellite can be received by a ground station, with brighter regions indicating more reliable reception. Figure 7a shows a sub-optimal ground station placement, located at the corner of a building and surrounded by tall trees. This

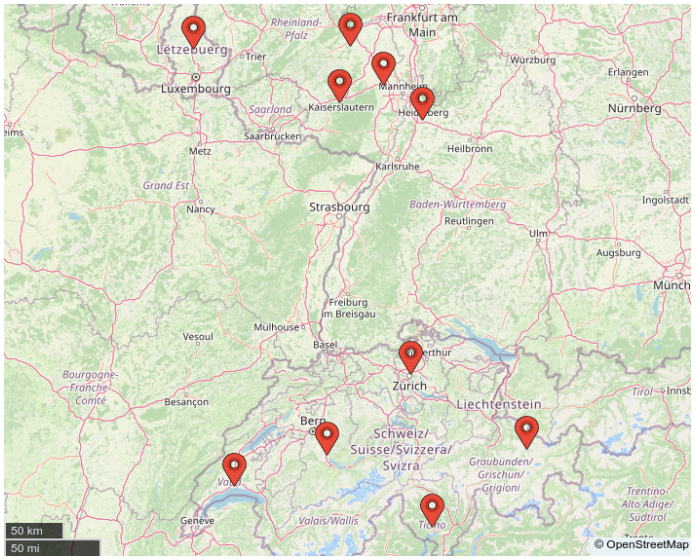
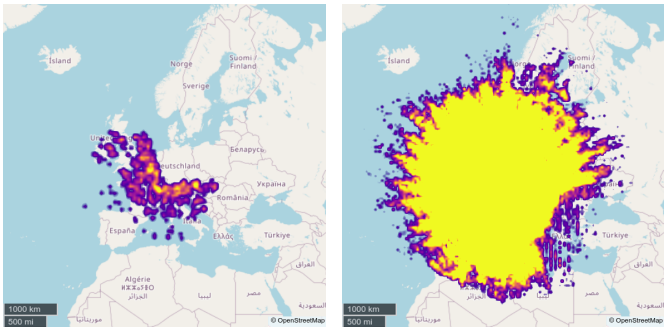


Fig. 6: Map of the currently placed ground stations.



(a) Sub-optimal ground station placement.

(b) Near-optimal ground station placement.

Fig. 7: Heatmap of receivable Iridium satellite locations.

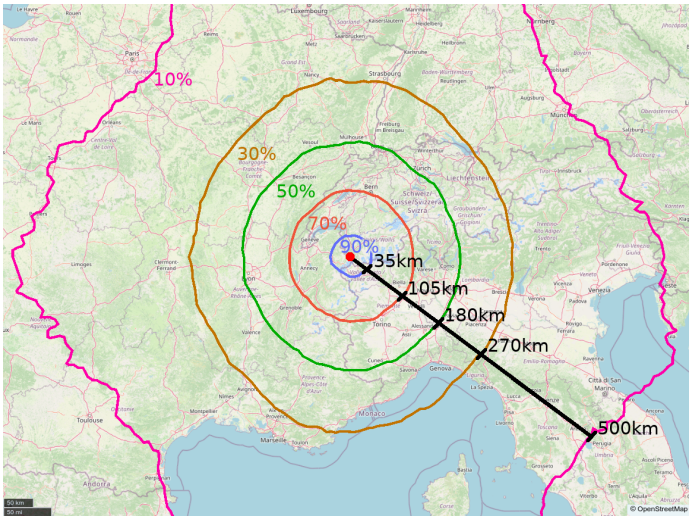


Fig. 8: Map of the Iridium downlink observation area as seen by a ground station. As the distance between the terminal and the ground station increases, the ability to observe messages decreases.

station has limited coverage, and during one hour of typical Iridium traffic, it will only receive about one or two thousand messages, equating to roughly 2 MB of demodulated data. Figure 7b (on the right) shows the coverage area of a well-placed ground station, capable of receiving messages from satellites across central Europe. It can capture up to 100 MB of demodulated Iridium messages in one hour of typical traffic. The data is compressed to about 15 MB before being uploaded to the server. After post-processing, this results in approximately 120,000 Iridium messages, including 1500 Iridium Ring Alert messages. This underscores the importance of optimal receiver placement.

The highlighted areas in Figure 7 represent the satellite positions during the reception of an Iridium message. This is not the earth surface that is reliably observed by a receiver, meaning the area where a ground terminal is located and the forward-downlink traffic from the satellite to the terminal is reliably received by the ground station. This area is influenced by the footprint size of the satellite antennas. For Iridium, the average antenna footprint diameter is about 400 km. The observation area is illustrated in Figure 8. If an Iridium terminal is located within the inner blue circle (35 km radius), a ground station (located at the red dot in the center) can observe at least 90% of the forward-downlink traffic to the terminal. As the distance increases, observed traffic decreases: 70% within the red circle (105 km), 50% within the green circle (180 km), 30% within the orange circle (270 km), and 10% within the purple circle (500 km). Forward-downlink traffic is receivable when both the ground station and terminal are within the same satellite antenna beam, but as the distance grows, the time spent in the shared beam decreases.

VII. Use cases in Existing Work

A. Recreation of previous measurements

With its capabilities of receiving Iridium messages, LeoCommon can be used to recreate measurements of several publications: In the GNSS spoofing detection work by Oligeri et al. [12], Iridium Ring Alert (IRA) messages were extensively collected to develop a model for location estimation to detect GNSS spoofing. Our network can reliably record IRA messages at one or multiple locations simultaneously. Additionally, the integrated GPS receiver provides a location reference for the ground station. In previous work of ours [7], IRA messages were used to develop a model of satellite antenna beams, which, when combined with further traffic recordings, helps estimate the location of Iridium ground terminals. Our network is capable of receiving IRA messages alongside other Iridium traffic. It delivers reliable timing information for each message reception and is capable of receiving traffic on multiple locations in parallel. In the work of Gurren et al. [5], an Iridium ground terminal and its connections via WiFi and the Iridium satellite network were analyzed to identify security vulnerabilities. To support this application, our ground stations can receive the forward-downlink (from the satellite to the user terminal) and also the return-uplink (from the user terminal to the satellite), provided the terminal is within line of sight of a ground station. This capability allows us to apply the same tools used by Gurren et al. for further security analysis of the recorded Iridium traffic.

Using the capability to record raw IQ (in-phase & quadrature) samples, several research projects that trained neural networks for authenticating a satellite can be reproduced: Three notable works are from Oligeri et al. [13], Zhu et al. [21] and Smailes et al. [17]: Oligeri represents the IQ samples as 224x224 pixel grayscale images fed into a Convolutional Neural Network (CNN) for classification. Combined with an autoencoder, this achieves an accuracy between 0.8 and 1, depending on the scenario. Zhu enhances Oligeri's approach by using a 3D CNN with the temporal property as the third feature, reducing the required sample rate to 42% while maintaining an accuracy of 0.92. Smailes takes a different approach using a Siamese Neural Network that compares two signals to determine the likelihood they were sent from the same transmitter, achieving an accuracy of 0.95. LeoCommon supports this method by capturing raw IQ samples of IRA headers and uploading them to the server.

In addition to using raw samples of IRA messages for fingerprinting, they can also enhance the accuracy of location estimation for an Iridium terminal, as demonstrated by Liang et al. [10] and Kassas et al. [9]. Liang employs a combined delay-Doppler estimation algorithm, working on raw samples of IRA message headers to obtain accurate values to estimate the receiver's location with a precision of 200 m. By recording IRA message headers, these measurements can be directly reproduced, and the calculated location can be compared to the GPS-provided ground station position. Kassas uses signals from four LEO satellite constellations: Starlink, OneWeb, Orbcomm, and Iridium, to calculate the receiver's location using

Doppler measurements for extracting navigation information. In a stationary scenario, they achieved location accuracy within 5 meters, while in a dynamic scenario, the average error was 10 meters. This highlights the importance of combined receiving capabilities from multiple constellations for research.

B. Enhancing existing work

There are also research projects that can directly benefit from LeoCommon by supporting their work with additional real-world measurements:

In the work of Xiao et al. [20], the focus is on the secrecy capacity of the forward-downlink of a satellite in non-geostationary orbit. The satellite provides services to a fixed earth station, while a fixed eavesdropper attempts to intercept the communication. Their paper presents a theoretical analysis of secure communication performance, along with extensive simulations to validate their approach. LeoCommon could be used to set up such a scenario and evaluate the theoretical and simulated results under real-world conditions.

As a possible complement to this, Aigul et al.[1] demonstrated improvements in signal reception reliability through the use of Kalman filtering. Their simulations show that Kalman filters are effective even at a negative signal-to-noise ratio for identifying a signal. By incorporating their findings into our software, it would be possible to evaluate these results under real-world conditions across multiple satellite systems. This improvement could also enhance the reliability of signal reception in our ground stations.

Another previous work of ours [8] conducted simulations on using TDOA-based fingerprints of satellite messages to authenticate the sending satellite. LeoCommon could support this research by providing a measurement platform to receive Iridium messages simultaneously at different locations and providing accurate timestamps to each message.

The scenario of combining multiple LEO communication satellite constellations for precise location calculation is explored by Farhangian et al. [4]. They use "signals of opportunity" from various constellations to perform Doppler measurements, enhancing the location estimations of an inertial navigation system. By simulating signals from different satellite systems for their calculations, they highlight the need for a receiver system capable of reliably receiving signals from multiple satellite systems.

VIII. Future development

We began by implementing a cost-efficient solution for receiving Iridium signals and aim to expand to a platform supporting more constellations. Currently, the ground station software can receive Iridium satellite signals, demodulate all received messages, or process the raw headers of IRA messages. Moving forward, we plan to integrate more LEO systems, starting with the Globalstar message decoder. This decoder will function similarly to the Iridium message receiver, demodulating and collecting all messages received from this system. Next, we aim to integrate receiving capabilities for Orbcomm and evaluate the feasibility of receiving signals from Starlink and OneWeb.

It is also worth noting that supporting more SDRs beyond the HackRF One will enhance the platform's adaptability to specific tasks. Additionally, to accommodate different satellite systems, multiple antennas will be required. To streamline this process, we plan to connect all antennas to a switch that can be controlled via software, allowing the necessary antenna to be selected for each receiving task without requiring manual changes to the setup.

Currently, we are using Raspberry Pi 4B and 5B due to their widespread adoption and good support for hardware modules and software. However, with the increasing availability of affordable x64 mini-PCs, it is promising to explore their performance. If they prove suitable, they can be added to the list of supported devices. Additionally, finding alternatives for the combined GPS-LTE board would offer more flexibility, allowing the platform to be better adapted to specific requirements. For instance, a standard GPS-USB stick could suffice for locations with a reliable LAN connection, while an LTE-USB stick could provide connectivity as an alternative to the combined GPS-LTE board.

To further enhance the quality of recordings, we aim to investigate and eliminate the cause of the remaining jitter in time measurements. Recent theoretical improvements in signal reception reliability, as demonstrated by Aigul et al. [1], offer a potential method for improving signal detection, which still needs to be explored in a large-scale real-world environment.

IX. Conclusion

In this paper, we introduced LeoCommon, an open-source ground station observatory network designed for experimental research in LEO satellite communications. It is available on GitHub: https://github.com/LeoCommon. We outlined its architecture, design, and implementation, all driven by the goal of providing a low-cost, extensible, multi-constellation experimental platform with high-quality recorded data. LeoCommon focuses on minimizing data loss and ensuring time-synchronized measurements at distributed ground stations, both critical for empirical research in satellite networks. We presented the current capabilities of the platform, including coverage and observation areas, which demonstrate its favorable characteristics. Additionally, we explored the potential use cases of LeoCommon in existing satellite communications research and highlighted its potential for further development. We aim to extend LeoCommon's capabilities concerning additional constellations, supported hardware, and increased flexibility for conducting distributed measurement campaigns.

Acknowledgment

The authors would like to thank additional contributors to the network: Benjamin K. and Sarah S. for their work on the server and website, Elisabeth U. and Vanessa S. for their work on the ground stations.

References

[1] Kulakayeva Aigul, Aitmagambetov Altay, Daineko Yevgeniya, Medetov Bekbolat, and Ongenbayeva Zhadyra. Improvement of signal reception reliability at satellite spectrum monitoring system. *IEEE Access*, 10:101399–101407, 2022.

[2] Kevin Croissant, Dan White, Xabier Crespo Álvarez, Vasileios Adamopoulos, Cees Bassa, Redouane Boumghar, Hugh Brown, Fredy Damkalis, Ilias Daradimos, Patrick Dohmen, et al. An Updated Overview of the Satellite Networked Open Ground Stations (SatNOGS) Project. 2022.

[3] Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, et al. Language modeling is compression. *arXiv preprint arXiv:2309.10668*, 2023.

[4] Farzan Farhangian and Rene Jr Landry. High-order pseudorange rate measurement model for multi-constellation leo/ins integration: Case of iridium-next, orbcomm, and globalstar. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 237(4):925–939, 2023.

[5] Jordan Gurren, Avanthika Vineetha Harish, Kimberly Tam, and Kevin Jones. Security implications of a satellite communication device on wireless networks using pentesting. In *2023 19th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 292–298. IEEE, 2023.

[6] Todd E Humphreys, Peter A Iannucci, Zacharias M Komodromos, and Andrew M Graff. Signal structure of the Starlink Ku-band downlink. *IEEE Transactions on Aerospace and Electronic Systems*, 59(5), 2023.

[7] Eric Jedermann, Martin Strohmeier, Vincent Lenders, and Jens Schmitt. RECORD: A RECeption-Only region determination attack on LEO satellite users. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 6113–6130. USENIX Association, 2024.

[8] Eric Jedermann, Martin Strohmeier, Matthias Schäfer, Jens Schmitt, and Vincent Lenders. Orbit-based authentication using TDOA signatures in satellite networks. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2021.

[9] Zaher M Kassas, Sharbel Kozhaya, Haitham Kanj, Joe Saroufim, Samer W Hayek, Mohammad Neinavaie, Nadim Khairallah, and Joe Khalife. Navigation with multi-constellation leo satellite signals of opportunity: Starlink, oneweb, orbcomm, and iridium. In *2023 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 338–343. IEEE, 2023.

[10] Huaiyuan Liang, Honglei Qin, and Haotian Li. Doppler compensated pseudorange based signal-of-opportunity positioning using iridium satellite. *IEEE Transactions on Aerospace and Electronic Systems*, 2024.

[11] Aniol Martí, Jordi Portell, Jaume Riba, and Orestes Mas. Context-aware lossless and lossy compression of radio frequency signals. *Sensors*, 23(7):3552, 2023.

[12] Gabriele Oligeri, Savio Sciancalepore, and Roberto Di Pietro. Gnss spoofing detection via opportunistic iridium signals. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 42–52, 2020.

[13] Gabriele Oligeri, Savio Sciancalepore, Simone Raponi, and Roberto Di Pietro. Past-ai: Physical-layer authentication of satellite transmitters via deep learning. *IEEE Transactions on Information Forensics and Security*, 18:274–289, 2022.

[14] Zachary Parker, Scott Poe, and Susan V. Vrbsky. Comparing NoSQL MongoDB to an SQL DB. In *51st Annual ACM Southeast Conference*, ACMSE '13, 2013.

[15] Damian Pfammatter, Domenico Giustiniano, and Vincent Lenders. A software-defined sensor architecture for large-scale wideband spectrum monitoring. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, IPSN '15, 2015.

[16] Tobias Schneider and Stefan Zehl. gr-iridium: GNU Radio Iridium Out Of Tree Module. *Chaos Computer Club München*, 2022.

[17] Joshua Smailes, Sebastian Köhler, Simon Birnbach, Martin Strohmeier, and Ivan Martinovic. Watch this space: Securing satellite communication through resilient transmitter fingerprinting. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 608–621, 2023.

[18] Nathan West, Douglas Geiger, and George Scheets. Accelerating software radio on arm: Adding neon support to volk. In *2015 IEEE Radio and Wireless Symposium (RWS)*, pages 174–176. IEEE, 2015.

[19] Daniel J White, Ioannis Giannelos, Agisilaos Zissimatos, Eleytherios Kosmas, Dimitrios Papadeas, Pierros Papadeas, Matthaios Papamathaiou, Nikolaos Roussos, Vasileios Tsiligiannis, and Ioannis Charitopoulos. SatNOGS: satellite networked open ground station. 2015.

[20] Yeqiu Xiao, Jia Liu, Yulong Shen, Xiaohong Jiang, and Norio Shiratori. Secure communication in non-geostationary orbit satellite systems: A physical layer security perspective. *IEEE Access*, 7:3371–3382, 2018.

[21] Siyu Zhu, Yuanyu Zhang, Jinxiao Zhu, Yin Chen, Yulong Shen, and Xiaohong Jiang. 3d convolution-based radio frequency fingerprinting for satellite authentication. In *GLOBECOM 2023-2023 IEEE Global Communications Conference*, pages 7586–7591. IEEE, 2023.