Developing an SDR-Based LoRa Communication System with a Web-Based Monitoring and Control Interface for Performance Analysis

Nguyen Viet Hung
Faculty of Telecommunications 1
Posts and Telecommunications Institute of Technology
Hanoi, Vietnam
nvhung_vt1@ptit.edu.vn

Vi Minh Hieu
Faculty of Telecommunications 1
Posts and Telecommunications Institute of Technology
Hanoi, Vietnam
hieuvm.b22vt197@stu.ptit.edu.vn

Nguyen Quy Duong
Faculty of Telecommunications 1
Posts and Telecommunications Institute of Technology
Hanoi, Vietnam
duongnq.b22vt113@stu.ptit.edu.vn

Duong Thi Thanh Tu
Faculty of Telecommunications 1
Posts and Telecommunications Institute of Technology
Hanoi, Vietnam
tudtt@ptit.edu.vn

Nguyen Kim Khoa
Department of Electrical Engineering
Ecole de technologie superieure
University of Quebec
Quebec, Canada
kim-khoa.nguyen@etsmtl.ca

Le Van Hau
Department of Electrical Engineering
Ecole de technologie superieure
University of Quebec
Quebec, Canada
van-hau.le@etsmtl.ca

***Abstract*— LoRa communication is currently gaining significant research attention globally for various applications, such as the Internet of Things (IoT) owing to its long-range transmission and energy-efficient characteristics. Meanwhile, Software-Defined Radio (SDR) technology, with its flexible customization and rapid adaptability for diverse radio applications, fully meets the requirements of such communication systems. While simulation studies and commercial deployments exist, there is a lack of reports that integrate theoretical evaluation with experimental validation on SDR platforms, particularly those incorporating a real-time monitoring framework. This paper presents a complete LoRa testbed based on ADALM-PLUTO and GNU Radio, which integrates a real-time dashboard using Node-RED, MQTT, and Database Management System (DBMS). The research methodology comprises two parts: (i) Monte Carlo simulations over an AWGN channel to establish a theoretical Bit Error Rate (BER) vs. Signal-to-Noise Ratio (SNR) baseline for various Spreading Factors and Coding Rates, combined with Time-on-Air calculation; and (ii) experimental BER/Packet Error Rate (PER) measurements on the SDR platform to compare against the theoretical baseline and assess real-world performance. The real-time dashboard enables the monitoring and control of message transmission/reception, observation of signal strength, and real-time tracking of the packet error rate.**

Keywords—LoRa, Internet-of-Things, Software Defined Radio, GNU Radio

I. INTRODUCTION

The Internet of Things (IoT) is expanding at an unprecedented rate, generating significant demand for diverse connectivity technologies. To address varying requirements for bandwidth, range, and energy consumption, numerous communication technologies have been developed: 5G provides high bandwidth and low latency; WiFi and Bluetooth are suitable for local area networks; NB-IoT is optimized for stationary devices; whereas Low-Power Wide-Area Network (LPWAN) technologies, such as LoRa, are specifically designed for long-range communication with extremely low power consumption.These characteristics have led to LoRa's widespread adoption in diverse domains: smart agriculture, smart cities, industry, and scenarios involving complementary connectivity via LEO satellites to extend coverage to remote regions. Utilizing a star topology, end-devices communicate with LoRa gateways, which then relay data to a network server, creating a flexible ecosystem for large-scale IoT applications.

The LoRa technology stack is divided into two main components: The LoRa PHY (Physical Layer) and LoRaWAN (MAC Layer) [1]. The LoRa PHY employs CSS modulation, which is proven to be highly robust against in-band and out-of-band interference—a common issue in shared ISM bands. The LoRaWAN protocol operates at the MAC Layer, managing medium access control corresponding to the Data Link Layer of the OSI 7-layer model [2]. While the LoRaWAN MAC protocol is an open standard, the LoRa PHY is proprietary technology owned by Semtech. This proprietary nature obscures many specific implementation details from the research community, impeding the full exploration of its potential, hindering performance enhancements, and limiting the development of advanced LoRa-based applications. This information gap has motivated numerous reverse-engineering efforts to gain a comprehensive understanding of its underlying mechanisms, including packet modulation, demodulation, and preamble detection.

Currently, to mitigate this opacity, Software-Defined Radio (SDR) platforms have become essential tools for reverse-engineering and analyzing the LoRa PHY. SDR devices allow researchers to bypass fixed commercial transceivers in favor of flexible, software-controlled implementations [3], [4]. However, existing SDR-based testbeds often face challenges related to high hardware costs or incomplete open-source support.

Hereby, we present the design, development, and performance analysis of a comprehensive LoRa communication system implemented on SDR. The primary objective is to create a flexible, low-cost, and reproducible testbed that enables in-depth investigation of the LoRa physical layer, free from the constraints of proprietary commercial hardware. Our system utilizes the ADALM-PLUTO hardware, is implemented entirely using open-source tools, and this paper evaluates its resulting performance.

The rest of this paper is structured as follows. Section II describes the detailed architecture and implementation of the system. Section III evaluates its performance. Finally, we conclude the paper in Section IV.

II. System Design and Implementation

In this section, we present the implementation of the LoRa communication system and a web-based interface designed to monitor and control the message transmission and reception process between two ADALM-PLUTO devices which is called LoRaSDR Chat.

A. *Communication System*

The system is primarily implemented within the GNU Radio environment, a leading open-source software development framework specifically designed for Software-Defined Radio (SDR) applications. GNU Radio was selected as the development platform due to its high degree of modularity and visual development paradigm. This characteristic permits the rapid design and prototyping of complex signal processing flowgraphs by graphically interconnecting discrete functional blocks. We have extended the Web Application Programming Interface (API) to incorporate the MQTT protocol, and the source code including GNU Radio Python Scripts and Node-RED Flow Scripts is now publicly accessible on GitHub[1].

As the LoRa Physical Layer (PHY) processing blocks are not standard components within GNU Radio, the core of this SDR-based LoRa implementation utilizes a third-party, open-source module available on GitHub (tapparelj/gr-lora_sdr) [5]. This module provides the necessary LoRa modulator and demodulator blocks. On the hardware side, each ADALM-PLUTO (PlutoSDR) device is connected via USB to a Linux host computer executing the GNU Radio flowgraph. A typical GNU Radio flowgraph for a LoRa link comprises two main processing streams, operating on separate devices:

- Transmission (Tx) Flow: This processing stream begins with a Message Source block to generate the payload, such as a user-defined text string or binary data. This data is then fed into the LoRa Modulator block [5], which is the core component of the transmitter. This block is responsible for executing critical PHY layer tasks. These tasks include generating the preamble for packet detection and synchronization at the receiver, applying Forward Error Correction (FEC) to add redundancy for enhancing reliability against channel noise, and performing the characteristic Chirp Spread Spectrum (CSS) modulation using "Up-chirps" [6] (linearly frequency-increasing signals) to encode the data symbols. The resulting complex baseband (digital chirp) signal is then passed to the PlutoSDR Sink block. This block interfaces with the SDR hardware, performs the necessary Digital-to-Analog (D/A) conversion, upconverts to the target RF frequency, and facilitates wireless transmission via the antenna.

- Reception (Rx) Flow: Conversely, the signal is captured by the PlutoSDR Source block, which continuously samples the RF environment, performs Analog-to-Digital conversion and downconversion, and passes the resulting digital samples into the flowgraph. These samples are fed into the LoRa Decoder block [5]. This block executes the complex inverse operations: it first scans the incoming stream for preamble detection and performs packet synchronization. It then executes the CSS demodulation, typically by multiplying the received signal with a reference "Down-chirp" [6] (a linearly frequency-decreasing signal) to de-spread the desired signal. Finally, this block applies FEC decoding to correct potential bit errors and recover the original payload.

[1] github.com/Brauuwu/LoRa-SDR_Chat

In the direct communication model shown in Fig. 1, the signal flowgraph represents a unidirectional communication process, or Simplex Mode. Consequently, one Pluto device is assigned the fixed role of the transmitter (Tx) and the other Pluto device serves entirely as the receiver (Rx). The signal transmission path is configured to utilize a specific narrowband ISM (Industrial, Scientific, and Medical) frequency, adhering to regional spectrum regulations.

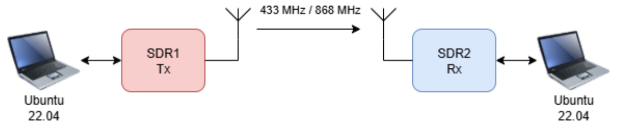

Fig. 1 LoRa SDR Communication System Model

Analogous to a fundamental digital communication system, this LoRa transceiver architecture incorporates input message processing blocks at the transmitter, such as encoding (FEC) and modulation (CSS), and corresponding output processing blocks at the receiver, namely demodulation and decoding. To ensure system validation beyond simple time-domain waveform and frequency-domain spectral analysis, we utilize a complete message transmission and reception framework that includes error control. This end-to-end validation approach is particularly relevant as the LoRa protocol is designed to operate effectively even when its signal strength is minimal, often significantly below the low noise floor [3]. Evaluating performance under such low Signal-to-Noise Ratio (SNR) conditions necessitates testing the integrity of the decoded data, not just the presence of a signal.

B. *Web-based Monitoring and Control Interface*

To move beyond simple command-line validation and enable robust interaction, we developed a real-time, web-based monitoring interface to visualize the transmission/reception process and key performance parameters. This system is based on a decoupled, four-component architecture, depicted in 3, which strategically separates the high-fidelity signal processing layer from the data transport and user-facing visualization layers. This design ensures modularity and scalability. The four main components are:

1) Data Collection in GNU Radio: At the core of the receiver's flowgraph, critical information is extracted in real-time. This metadata—such as the Received Signal Strength Indicator (RSSI) of the latest valid packet, a cumulative count of received packets, the recovered message payload, and other PHY-layer metrics like SNR—is accessed from the GNU Radio environment. This is achieved by interfacing with the flowgraph using custom Python blocks or probes, which sample the data streams and prepare the information as structured messages (e.g., JSON) for external data exchange.

2) Data Transmission via MQTT Protocol: We selected MQTT, a lightweight application-layer IoT protocol, for its efficient and reliable publish-subscribe model. This model fundamentally decouples the data producer (the GNU Radio flowgraph) from the data consumers (the dashboard and database). We integrated MQTT Publisher blocks directly into the receiver's GNU Radio flowgraph. The Publisher is responsible for sending the collected data packets to a central MQTT Broker, with each data type assigned a distinct topic. This topic-based segregation is crucial, as it allows different components of the dashboard to subscribe only to the data streams they need, facilitating an organized and efficient visualization of messages versus performance parameters.

3) Visualization in Node-RED: Node-RED serves as the low-code backend and frontend for our interface. As a flow-based programming tool, it enabled the rapid development and iteration of the data processing logic without complex server-side coding. We established flows within Node-RED that subscribe to the relevant MQTT topics from the Broker. As messages arrive, they are parsed, processed, and channeled to the appropriate elements on the web dashboard. This dashboard is constructed using various graphical components (nodes) from the Node-RED library, such as time-series charts for historical data, gauges for instantaneous values like RSSI, and text fields for logging received payloads.

4) Data Storage and Querying with DBSM: Recognizing that real-time monitoring alone is insufficient for rigorous subsequent post-analysis, we extended the processing flow in Node-RED. A parallel branch in the Node-RED flow is dedicated to data persistence. In addition to displaying data on the dashboard, this branch formats the received data (and its associated metadata like RSSI, SNR, and a precise timestamp) into a standardized SQL query. This query is then executed to insert the data into a Database Management System (DBMS). By storing each received packet as a distinct row, we create a comprehensive and permanent experimental log. This database is invaluable for offline analysis, allowing us to correlate performance metrics against experimental variables long after the experiment has concluded.

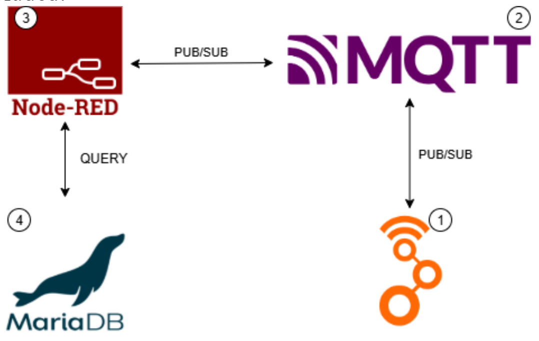

Fig. 2 GNU Radio to Node-RED interfacing via MQTT, with MariaDB backend for data persistence and management.

This architecture enables flexible and effective remote system monitoring via a standard web browser, offering an intuitive, consolidated overview of both the LoRa message reception status and the receiver's physical layer parameters. To complete this framework, a Graphical User Interface (GUI) was constructed using the nodes available in the Node-RED Dashboard. This interface provides a unified visual control and monitoring hub, enabling operators to interact with and evaluate the LoRa system's performance in real-time. The GUI is segmented into three primary, functionally distinct panels:

- *Control and Sending Panel:* Positioned on the left, this section provides a proactive mechanism for testing the communication link. It contains a text input field for message composition and a "SEND" button. When this button is pressed, the message payload is not handled locally; instead, it is published via MQTT to a separate topic. The transmitter's GNU Radio flowgraph, running on a different host, subscribes to this topic. This allows the operator to trigger on-demand transmissions directly from the web interface without requiring physical intervention or reconfiguration of the active flowgraph, thus creating a complete, closed-loop testing environment.
- *Monitoring and Feedback Panel:* The central area of the GUI serves as the primary real-time feedback log. It displays a timestamped, scrolling log of all successfully received messages, providing immediate confirmation of data integrity. Alongside this, it aggregates and displays critical performance statistics, such as the count of errored or missed packets, the calculated Bit Error Rate (BER), and other vital status information from the receiver.
- *Historical Charts Panel:* Located on the right, this panel is dedicated to visualizing trends over time. It features dynamic charts that track the fluctuation of the Received Signal Strength Indicator (RSSI) and the configured hardware gain at the receiver upon capturing LoRa signals. Displaying these two metrics together is crucial, as it allows the operator to distinguish whether a change in signal strength is due to channel fading or an adjustment in the receiver's own gain settings

III. Performance Evaluation

To comprehensively evaluate the system's performance and validate its operational capabilities within realistic contexts, we designed and conducted three distinct experimental scenarios. These scenarios were meticulously structured to methodically test the system under a gradient of environmental conditions and with varying core LoRa parameters, allowing us to isolate and understand different performance variables.

The first scenario focused on assessing the fundamental transmission range and link degradation within a controlled suburban environment. This setting, characterized by low-density buildings, open spaces, and some foliage, was chosen to represent a baseline "lightly obstructed" deployment. In this experiment, the key variable was distance; the transmission distance was incrementally increased from 100m to 400m at discrete waypoints. To simulate a challenging but common near-ground link, typical of many ground-based IoT sensor deployments, both transmitting and receiving antennas were positioned at a low altitude of approximately 1-2m above ground level. This configuration intentionally introduces propagation challenges from ground proximity and minor obstacles [7]. During this test, all core LoRa parameters were held constant to isolate the singular effect of path loss. The primary objective was to observe and quantify the degradation of link metrics, specifically the Received Signal Strength Indicator (RSSI) and the Packet Error Rate (PER), as a direct function of increasing distance. This allowed us to map the system's baseline path loss profile and identify the approximate range at which link reliability begins to collapse.

The second scenario transitioned to a far more challenging dense urban environment. This setting was selected to test the system's robustness under conditions of significant multipath fading, high ambient RF noise levels, and heavy signal attenuation from concrete structures. We established a fixed-distance link of 650m, specifically chosen to represent a typical urban NLOS scenario where no direct visual path existed between the transmitter and receiver. The low-altitude antenna configuration of 1-2m was deliberately maintained to emphasize these difficult NLOS conditions. In this harsh environment, the key variable under investigation was the Spreading Factor (SF), which was systematically varied across its operational range from SF7 to SF12. The core objective was to empirically quantify the critical trade-off between data rate (achieved with low SFs) and link reliability (provided by high SFs). By carefully measuring the Bit Error Rate (BER) and Packet Error Rate (PER) for each SF setting, we could pinpoint the exact performance gains of using slower, more robust modulation schemes. These empirical results were then compared against the theoretical performance baseline in an AWGN channel to evaluate the system's "implementation margin" and quantify the severe performance penalty imposed by the real-world urban multipath channel.

The third experimental scenario was rigorously designed to evaluate the system's long-range transmission capabilities at a distance of 1200m, with a specific focus on the critical role of antenna placement, marking a significant transition from the near-ground NLOS conditions of previous trials to a more favorable LOS or near-LOS propagation model. For this test, the antennas were elevated to a height of 10m, transitioning from a near-ground NLOS setup to a more favorable LOS or near-LOS condition. This elevation is crucial as it likely clears major ground-level obstructions and the first Fresnel zone, mitigating the severe multipath fading and signal absorption experienced in the 1-2m tests. This scenario had a dual purpose: first, to again evaluate the impact of the SF on decoded signal quality at this extended range, and second, to assess the influence of increased antenna height on decoding capability and overall coverage. Comparing the results of this test with the previous scenarios allows for a clear distinction between system limitations and channel-induced limitations.

Before presenting the experimental results, the performance of the LoRa-SDR system is first analyzed via AWGN channel simulation. To establish a theoretical baseline, we performed Monte Carlo simulations over an AWGN channel for SF7 through SF12, with a 125 kHz bandwidth, hard-decision decoding, and two distinct coding rates (CR): 4/5 and 4/8. Fig. 3 illustrates the resulting BER vs. SNR curves on a logarithmic scale. At a BER threshold of 10^{-3} with CR 4/5, the required SNR for each SF is approximately: SF7 ≈ -9 dB, SF8 ≈ -10 dB, SF9 ≈ -11.6 dB, SF10 ≈ -13 dB, SF11 ≈ -14.5 dB, and SF12 ≈ -16.2 dB. The use of CR 4/8 (4 check bits) provides improved BER performance compared to CR 4/5 (1 check bit) due to more robust FEC, enabling operation at a lower SNR for the same BER level. However, the primary trade-off is a significant increase in the Time-on-Air (ToA).

For a 50-byte payload, 125 kHz BW, and an 8-symbol preamble, the ToA increases by 43-47% when switching from CR 4/5 to CR 4/8. Specifically for SF12, the ToA is 2.3s for CR 4/5 and 3.3s for CR 4/8 (a 42.7% increase), resulting in an approximate 30% reduction in throughput and a ~43% increase in energy consumption per packet. Table I summarizes the ToA for all SFs.

TABLE I. TIME-ON-AIR COMPARISION FOR DIFFERENT SPREADING FACTOR AND CODING RATES.

SF	CR 4/5 (ms)	CR 4/8 (ms)	Increase (%)
7	97.5	143.6	47.2
8	174.6	254.5	45.7
9	328.7	476.2	44.9
10	616.5	886.8	43.9
11	1314.8	1904.6	44.9
12	2302.0	3285.0	42.7

To rigorously evaluate the system's reliability in realistic operating conditions, the first experimental phase focused on quantifying the Bit Error Rate (BER) across transmission distances ranging from 130m to 410m. Both the transmitting and receiving antennas were positioned at a low height of 1m above ground level to simulate a worst-case scenario for ground-based IoT nodes. The system parameters were configured as follows: Spreading Factor (SF) = 7, Coding Rate (CR) = 4/5, Bandwidth (BW) = 125 kHz, and a sampling rate of 2 MHz. Fig. 4 illustrates the measured BER curve versus distance on a logarithmic scale. BER increases gradually from 1.15×10^{-4} at 130m to 4.90×10^{-3} at the maximum tested range of 410m. Despite this increase, the connection remained maintained, validating the effectiveness of the LoRa physical layer even with the lowest spreading factor.

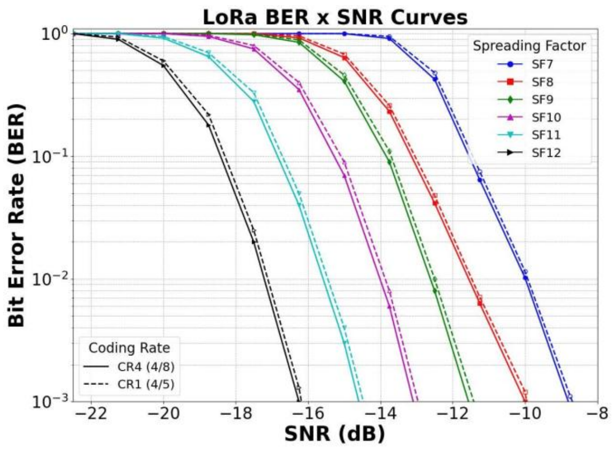

Fig. 3 BER vs. SNR Curves for different Spreading Factors (SFs)

Comparison with AWGN theoretical baseline: Fig. 3 presents the BER-SNR curve obtained from Monte Carlo simulation over an AWGN channel. For SF7, BER = 10^{-3} requires SNR ≈ -9 dB according to theory. Comparing with experimental results, BER = 10^{-3} is achieved at approximately 215-220m distance. Therefore, the SNR condition of ≈ -9 dB in AWGN simulation corresponds to an actual distance of ≈ 220m in a suburban environment with parameters SF7 and CR 4/5. This reflects the impact of multipath fading, shadowing from large obstacles, noise sources, and ambient noise in real-world environments.

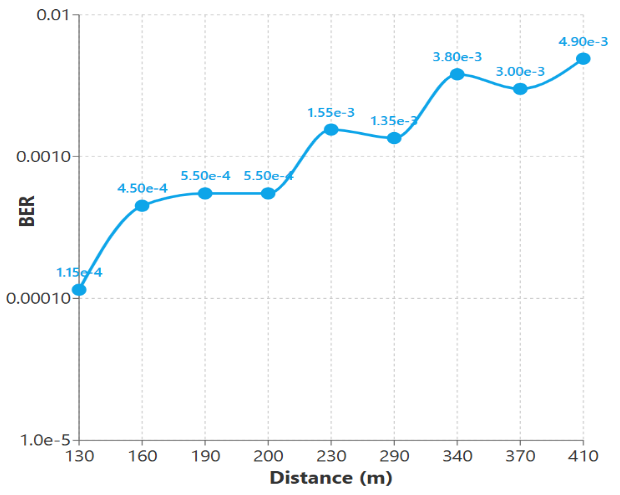

Fig. 4 BER Performance versus Transmission Distance for SF7

In the second experiment, we performed BER/PER measurements at a fixed distance of 650m in an interference-heavy urban environment, with both transmitting and receiving antennas placed 2m above ground level. The system parameters were set as follows: Bandwidth (BW) = 125 kHz, Coding Rate (CR) = 4/6, with Spreading Factors from SF7 to SF12. The corresponding sampling rates are provided in the TABLE II.

TABLE II. SAMPLE RATE CONFIGURATION FOR DIFFERENT FOR SPREADING FACTORS

SF	7	8	9	10	11	12
SR	250k	250k	250k	250k	250k	125k

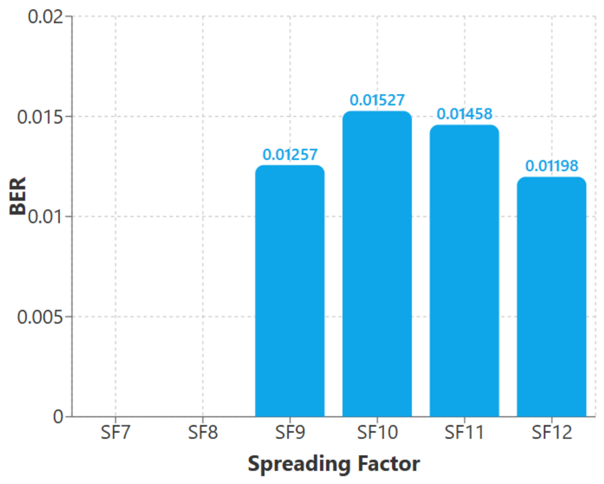

Fig. 5 BER Comparison for SF9-SF12 in Urban Environment (650m)

For SF = 7 and SF = 8, the receiver failed to decode any packets. This reflects a critical finding: at the 650m distance in the urban environment, and with a low antenna height (2m), the receiver sensitivity for SF7 and SF8 was insufficient to overcome the high interference and background noise, resulting in an SNR below the decoding threshold.

For SF = 9, a BER of 1.26x10^{-2} was recorded. This indicates that while the system began to successfully decode some packets, the bit error rate remained extremely high. This high BER was corroborated by observations on the web interface, which indicated that "a significant number of packets did not reach the destination."

For SF = 11 and SF = 12, the BER ranged from 1.46x10^{-2} to 1.53x10^{-2} Although higher SFs (SF10-SF12) enable the demodulator to operate at lower SNRs due to longer symbol durations, the BER remained high. This is attributed to the severe channel conditions within the urban environment, characterized by numerous obstructions and heavy interference.

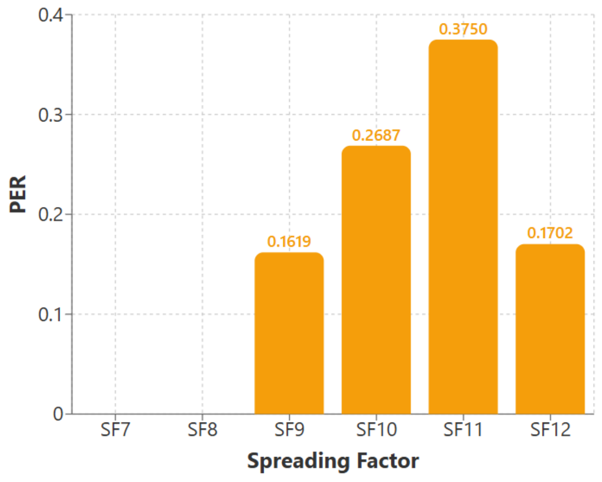

Fig. 6 PER Comparison for SF9-SF12 in Urban Environment (650m)

Furthermore, the Packet Error Rate (PER) results confirmed that only SF12 was sufficiently robust for transmission at 650m in this heavily obstructed, high-interference urban environment with the 2m antenna height.

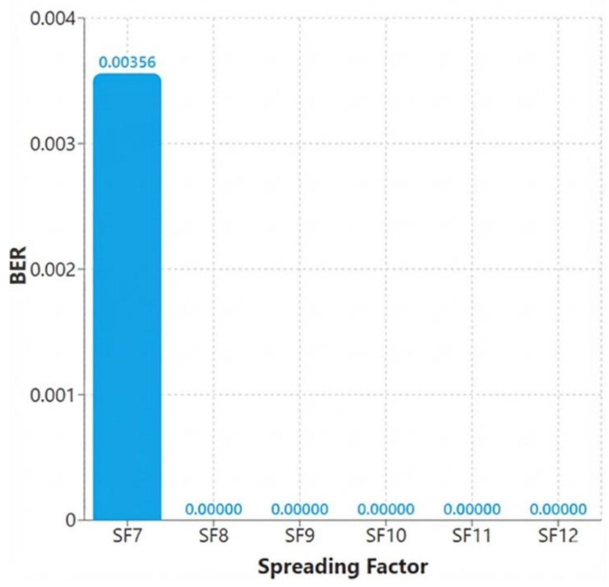

Fig. 7 BER versus Spreading Factor with Elevated Antenna (1200m)

Recognizing the pronounced impact of obstructions on channel quality, we relocated the transceiver system to an elevated height of 10m above ground. In this experiment, the system transmitted approximately 100 packets; the results were updated in real-time on the web interface and are summarized in the following table. SF7 yields a bit error rate of 0.00356, while from SF8 onwards, the system successfully decodes all received messages with zero errors. Notably, the results improve significantly despite the transmission distance doubling and the persistent presence of heavy noise in the environment.

IV. Conclusion

This research successfully presented a comprehensive and unified evaluation framework for LoRa communication, integrating an SDR platform (based on ADALM-PLUTO and GNU Radio) with an innovative web-based real-time monitoring dashboard and database logging capabilities. This framework addresses the critical need for end-to-end performance validation in practical deployment scenarios, supporting both theoretical analysis and experimental validation. The extensive experimental campaigns, conducted across urban, suburban, and varying antenna height environments, provided crucial benchmark data, clearly quantifying the trade-offs between range, environmental path loss, and link reliability (e.g., Packet Error Rate). These empirical results contribute vital insight into optimal LoRa configurations. In summary, this work advances a unified experimental methodology and establishes a solid foundation for the optimal configuration of both IoT and LEO satellite systems utilizing the LoRa physical layer. Future work will focus on extending experimental coverage, increasing the sample size, evaluating more complex propagation conditions, and developing adaptive parameter optimization mechanisms.

References

[1] A. A. Abuarqoub, M. A. Ferrer, J. F. Timmons, and R. V. Prasad, "On the feasibility of open-source LoRa PHY implementations: A comprehensive study," Computer Networks, vol. 240, p. 110083, Apr. 2024. doi: 10.1016/j.comnet.2024.110083.

[2] V. Carvalho, M. Feldman, I. Müller, and M. Götz, "Comparison of simulation, SDR implementation and commercial device on LoRa protocol," in Proc. 2024 8th Int. Symp. Instrum. Syst., Circuits Transducers (INSCIT), 2024, pp. 1–6. doi: 10.1109/INSCIT62583.2024.10693394.

[3] J. P. de Omena Simas, D. G. Riviello, and R. Garello, "Software-defined radio implementation of a LoRa transceiver," Sensors, vol. 24, no. 15, p. 4825, Jul. 2024. doi: 10.3390/s24154825.

[4] R. M. Colombo, A. Mahmood, E. Sisinni, P. Ferrari, and M. Gidlund, "Low-cost SDR-based tool for evaluating LoRa satellite communications," in Proc. 2022 IEEE Int. Symp. Meas. Netw. (M&N), 2022, pp. 1–6. doi: 10.1109/MN55117.2022.9887761.

[5] J. Tapparel and A. Burg, "Design and implementation of LoRa physical layer in GNU radio," in Proc. 14th GNU Radio Conf., 2024.

[6] Semtech, "LoRa and LoRaWAN: A Technical Overview," Semtech White Paper, 2020.

[7] M. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "Do LoRa low-power wide-area networks scale?" in Proc. ACM Int. Conf. Model., Anal. Simul. Wireless Mobile Syst. (MSWiM '16), 2016, pp. 59-67. doi: 10.1145/2988287.2989163.