An Open Emulator for Smart Radio Environments

Abstract—**Reconfigurable Intelligent Surfaces (RIS) achieve Smart Radio Environments (SREs) when controlled appropriately. Multiple Testbeds, including the Standards-Compliant ones, are now available to control the RIS configuration in Real-Time for practical RIS deployment. However, practical performance analysis of SREs is challenging due to the large number of hardware components (including RIS elements, Software-Defined Radios (SDRs)/Universal Software Radio Peripheral (USRPs) with mobility, etc.) required, necessitating faithful and reliable computer simulations/Emulation. Available Simulators lack the capability of accurately capturing the Lab-prototype RIS characteristics and also are not capable of integrating the Traffic Signal Generators with closed loop RIS control.**

This paper presents Design and evaluation of an open-source near-real time Emulator that replicates a Wireless Environment using USRP-based SDRs, with each SDR connected to its own Traffic Signal Generator or sink, and supporting user mobility. RIS configurations can be dynamically controlled through integration with existing Testbeds.

We validate the achieved SRE performance against Laboratory measurements using a practical, in-house-fabricated RIS. The Emulator's utility is demonstrated in planning and dimensioning a RIS-assisted long-distance Broadcast System for rural coverage. Replacing USRPs with the Emulator in our Python-based, baseband-level Real-Time Wireless System results in no loss of functionality, including support for complex features such as carrier frequency offset introduction, etc.

Index Terms—**Emulator, RIS, SDR, USRP, CFO, SRE.**

I. Introduction

Sixth-generation (6G) Wireless Networks are expected to support highly complex communication scenarios defined under the IMT-2030 framework, which include requirements such as massive device connectivity (10^8 devices/km²), latency below 0.1 ms, and integration of intelligent surfaces and AI-native operations [1]. Achieving these goals requires a paradigm shift from traditional Network Design, which primarily focuses on optimizing the transmitter and receiver, to a more comprehensive strategy that involves manipulating the Wireless channel itself to improve signal transmission. This shift transforms the Wireless channel from a passive, uncontrollable entity into an intelligent part of the Network, creating a Smart Radio Environment (SRE) [2].

A key enabling technology for SREs is the Reconfigurable Intelligent Surface (RIS) [3]. An RIS consists of a planar array of low-cost, passive elements, each capable of applying a controllable phase shift to incident electromagnetic waves [4].

As Wireless technologies evolve, accurate modeling and simulation become increasingly vital. Simulation tools have been central to the Design and Optimization of each Cellular Generation [5], offering a cost-effective, flexible, and low-risk Environment for innovation [6]. While physical Testbeds provide higher realism, they are often resource-intensive and impractical in early stages. Consequently, simulation has progressed from a pre-deployment tool to an integral part of the Network lifecycle, culminating in the concept of the Network Digital Twin [7].

Recent global efforts also emphasize the urgency of standardizing RIS simulation and Emulation Frameworks. Leading organizations such as ETSI [8], TSDSI [9], RISTA [10], and initiatives like the 6GBricks [11] project are actively working toward defining unified architectures, reference implementations, and performance benchmarks for RIS-enabled Wireless environments. Several Simulators available from other researchers are summarized in Table I with respect to their relevant capabilities. We developed a RIS-assisted simulation framework tailored to indoor communication scenarios under user mobility [12], [13].

This paper presents Design and evaluation of an open-source near-real time Emulator that replicates a Wireless Environment using Software-Defined Radios (SDRs), specifically Universal Software Radio Peripheral (USRPs), with each SDR connected to its own Traffic Signal Generator or sink, and allows for user mobility. Further, the RIS configurations can be dynamically controlled by integrating the Emulator with existing Testbed. Using the Emulator in place of USRPs in our Python-based baseband level Real-Time Wireless communication System, we observe that there is no loss in functionality, including complex aspects like carrier frequency offset introduction, etc.

The paper is organized as follows: Section II describes the architecture of the Emulator. Section III presents performance Validation, including Real-Time Trafficintegration and limitations under Real-Time constraints, and compares Emulator results with actual measurements. Section IV demonstrates the utility of the Emulator in planning a SRE Broadcast System for Rural connectivity.

II. Design of Emulator

Figure 1 provides a high-level working of the Emulator. The external entities integrated with the Emulator are shown in the **External Entities** block and include the baseband signal generators, the RIS controller that considers the channel estimates from these baseband entities and issues standard (TSDSI)-Compliant configuration commands to the hardware RIS driver. *These entities are not part of the Emulator and will be provided by third parties intending to use the Emulator exactly in the way they would use the SDRs.* The relevant interfaces of the Emulator with the external entities are explicitly depicted[1]. The practical passband processing

[1]One of the requirements was to ensure that the interfaces *Send_to_sim(Data, Operating Frequency, Sample Rate, Node ID)* and *Receive_from_sim(Samples Size, Operating Frequency, Sample Rate, Node ID)* should be similar to the ones provided by the relevant SDRs (for instance, the USRP provides the *usrp.send_waveform(samples, duration, center_freq, sample_rate, channel_ID, gain)* and *usrp.recv_num_samps(sample size, center_freq, sample_rate, channel_ID, gain)* interfaces, [24]) while performing the baseband level functionality.

TABLE I: Comparison of RIS Simulation Tools and Emulator

Simulator	**Mobility**	**Channel/Model Type**	**Open-source**	**RIS Parameters**	**Real-Time**	**Use Case / Scenario**
SimRIS [14]	No	Analytical	Yes	Phase response	No	Link-level modeling
QRIS [15]	Yes	Stochastic	Yes	RIS beam pattern	No	Multi-user RIS simulation
CoopeRIS [16]	Yes	Vehicular fading + Geometry	Yes	Dynamic phase shifts, mobile RIS	No	V2X and mobility-aware RIS
NYUSIM-RIS [17]	No	Ray-tracing + Extension	Yes	RIS reflection, frequency response	No	mmWave and THz propagation
SPM RIS [18]	Yes	Hybrid analytical-simulative	Yes	Element count, placement strategy	No	RIS placement and coverage
RIS_STAR [19]	No	Analytical	Yes	Transmission/reflection control	No	Channel hardening and coverage tests
Fraunhofer RT [20]	Yes	Ray-tracing	No	RIS orientation, material properties	Partial	Advanced ray-traced RIS visualization
CloudRT-RIS [21]	No	Ray-tracing	No	RIS capacity	No	Tunnel propagation
GoSimRIS [22]	Yes	Anlaytical	Yes	RIS controller config	Partial	O-RAN/xApp-based experiments
[23]	Yes	Analytical/Geometry	Yes	Phase profiles	No	End-to-end simulation
Emulator	**Yes**	**IQ sample + USRP interface Emulation**	**Yes**	**Real-Time control, support multiple RIS types**	**Yes**	**Support standard Compliant interfaces**

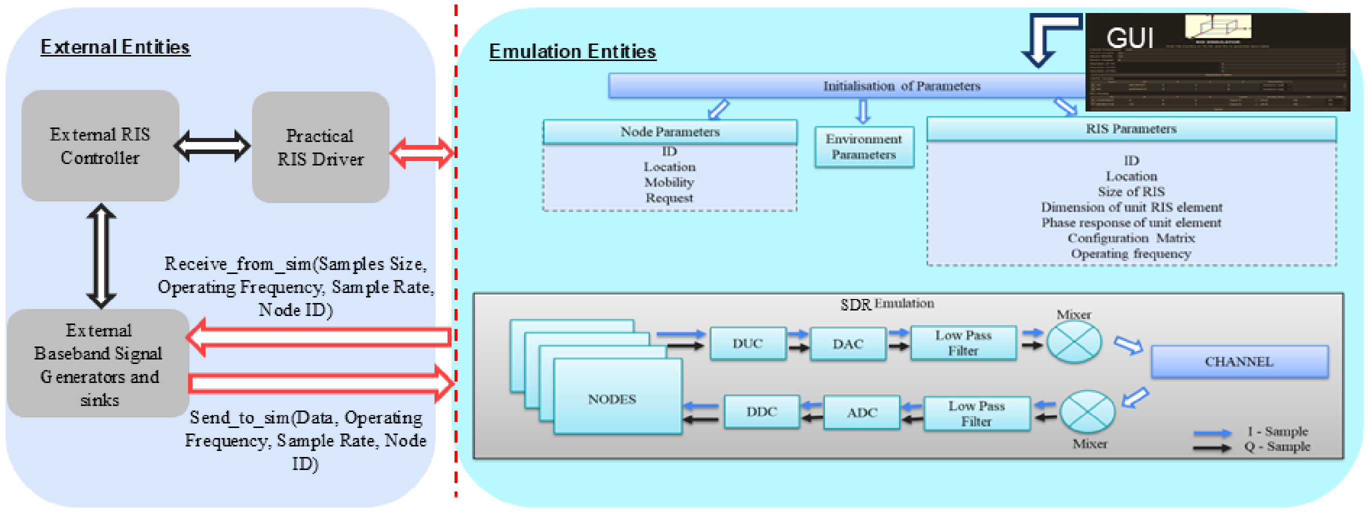


Fig. 1: System Architecture and Signal Flow of Emulator

while using these SDRs are also incorporated in the Emulator (**SDR Emulation** block of the **Emulation Entities** block). The Emulator operates in two distinct stages.

Initialization stage:

All the System parameters—covering node configuration, RIS abstraction (see Section II), and Environmental settings—are loaded from a user-provided structured JSON file. These parameters define spatial location, mobility, carrier frequency, and the physical layout and properties of the node. Users are provided with a GUI to easily describe the scenario. The abstraction of RIS elements, as obtained from field results, can be provided to the Emulator as a csv file that provides tuples of the form $\{frequency, configuration\_id, phase\_response, gain\_response, angle\_incidence, angle\_reflection\}$ as detailed in Section II. Dimensions of the unit cells in a RIS, along with the overall dimension and orientation are also provided. The GUI also allows the user to select individual node's initial location as well as the mobility model from a set of built-in models - the Emulator currently supports Random Waypoint [25], Random Direction [26], Gauss Markov Model [27] and Random Walk [28]. The modular Design (Appendix A) allows seamless integration of new mobility models. RF impairments for each baseband node are also registered during initialization. Inter-process communications (IPCs) with external entities are then established, with each communication channel uniquely associated with the corresponding node or RIS element using its assigned identifier. These interfaces, illustrated in Figure 1, are implemented using Linux FIFO queues, named pipes, and files, enabling isolated and efficient message exchange.

Emulation stage:

The Emulator operates in discrete time steps of duration τ. For mobile nodes, τ is selected such that the positional change within a step does not exceed 5% of the signal wavelength at the operating carrier frequency. For static nodes, τ is chosen based on the coherence time of the corresponding channel models. The simulation uses the minimum of these values to maintain fidelity.

At each time step, the Core/Engine.py (see Appendix A) performs the following operations:

- **Request Handling:** It polls IPC interfaces for new requests, such as IQ sample transmission/reception or reconfiguration of RIS elements.
- **Mobility Update:** It updates the spatial coordinates of all nodes using mobility models (implemented in Mobility_functions.py, Appendix A).
- **Emulation:** The Emulator processes the current System state to generate received IQ samples for all nodes operating in receive mode. It aggregates signal contributions from all active transmitters and emulates the effects of the Wireless channel using the models defined in Simulator_functions.py (Appendix A). This stage jointly models the SRE and SDR functionalities, including realistic RF impairments. The Emulation abstracts key processes such as Digital Up-conversion/Down-conversion, Analog-to-Digital and Digital-to-Analog conversion, frequency mixing, and filtering (Channel_functions.py Appendix A). Signal routing, coordination, and interaction with node-level request queues are handled via Simulator_functions.py (Appendix A). Various stages of the signal chain can be selectively enabled or bypassed to meet Real-Time execution constraints.
- **RIS abstraction :** The RIS is modeled using the analytical formulation of path-loss-based model [29], in which the RIS is represented as a planar array of sub-wavelength elements. Each element applies a programmable phase shift to incident electromagnetic waves. Each element's configuration is specified through a user-provided CSV file (referenced in `RIS.json`, Appendix A), containing tuples of the form $\{frequency, configuration\_id, phase\_response, gain\_response, angle\_incidence, angle\_reflection\}$.
  These parameters are used to evaluate the received power from the n-th RIS element as:
  $$P_{R,n} = P_T G_T(\hat{r}_n^i) G_R(-\hat{r}_n^s) \left(\frac{\lambda}{4\pi}\right)^2 \frac{G_e(-\hat{r}_n^i) G_e(\hat{r}_n^s)}{r_{i,n}^2 r_{s,n}^2} \varepsilon_p, \tag{1}$$
  where P_T denotes the transmit power, $G_T(\hat{r}_n^i)$ is the gain of the transmit antenna in the direction of the n-th RIS element, and $G_R(-\hat{r}_n^s)$ is the gain of the receive antenna in the direction from the n-th element. λ represents the carrier wavelength, while $r_{i,n}$ and $r_{s,n}$ are the distances from the transmitter to the n-th RIS element and from the n-th element to the receiver, respectively. The function $G_e(\cdot)$ denotes the element gain pattern, modeled as $G_e(\psi) = \gamma \cos^{2q}(\psi)$ for $0 \le \psi < \pi/2$ and zero otherwise, where ψ is the angle from the element broadside, γ is a normalization constant, and q controls the beamwidth; ε_p is the power efficiency factor accounting for reflection and conversion losses, satisfying $0 < \varepsilon_p \le 1$.
  At the receiver, the total signal contribution from all N RIS elements is computed as: $y = \sum_{n=1}^{N} b_n \sqrt{P_{R,n}} e^{j\phi_n}$ with the phase term: $\phi_n = \frac{2\pi}{\lambda}(r_{i,n} + r_{s,n})$ and controllable complex coefficients b_n representing the per-element RIS response (see Appendix A modules channel_functions.py).
- **RIS Control:** The Emulator supports Real-Time dynamic control of each RIS through a dedicated control interface (through external RIS Controller see Figure 1). At runtime, control commands are issued specifying the RIS identifier, the per-element phase shift vector $\{b_n\}_{n=1}^{N}$, the τ at which the configuration is to be applied, and the intended receiving node. These commands are stored in a time-ordered queue and activated sequentially as the simulation clock progresses. Upon activation, each command updates the RIS response by setting the per-element complex reflection coefficients as b_n.

III. Numerical Analysis

In this section, we evaluate the performance and fidelity of the proposed Emulator. First, we validate its integration with USRP-based SDRs by injecting Real-Time baseband signal generator and sink into the System (Subsection III-A). We then analyze the Emulator's ability to maintain Real-Time operation under various conditions, highlighting its performance boundaries (Subsection III-B). Finally, we assess the accuracy of the Emulator by comparing its outputs with Laboratory measurements obtained using a practical RIS prototype (Subsection III-C).

A. Demonstration of Real-Time Traffic Integration

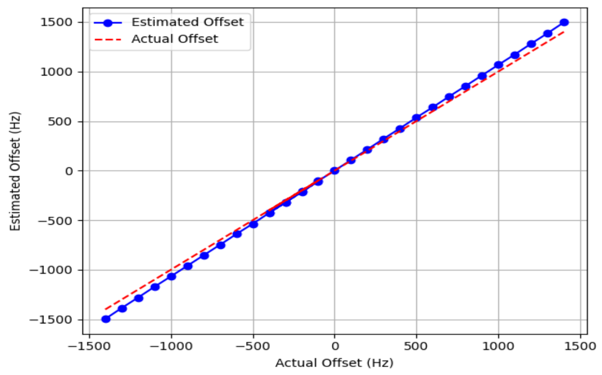


Fig. 2: Estimated vs. actual CFO values using BPSK receiver.

The Emulator is designed to replicate the Real-Time I/Q interface of a USRP to emulate hardware-like behavior of Wireless Systems entirely in software. By integrating an in-house Baseband BPSK-based Python communication System, we replicate USRP-like end-to-end communication.

To validate the fidelity of this Emulation, we focus on one of the most critical impairments in practical SDR Systems, Carrier Frequency Offset (CFO). CFO arises from mismatches between the local oscillators of the signal generator and sink, leading to cumulative phase rotation over time and resulting in constellation distortion and Bit Error Rate (BER) degradation [30].

The Emulator injects CFO values into the received waveform, while the BPSK sink performs Real-Time CFO estimation based on a predefined synchronization header. For the demonstration, we assume a Line-of-Sight (LOS) channel with fixed path loss and delay. *The existing BPSK signal generator and sink interfaced with the Emulator only by replacing their respective USRP APIs with our Emulator APIs.* Fig. 2 compares the actual CFO values injected by the Emulator with those estimated by the external BPSK sink. The results exhibit a strong linear correlation, confirming that the Emulator accurately reproduces CFO behavior typical of USRP-based Systems. Additionally, the sink's performance under these conditions validates the System's capability for Real-Time synchronization and demodulation within a software-only emulated Environment.

B. Near-Real-Time Profiling

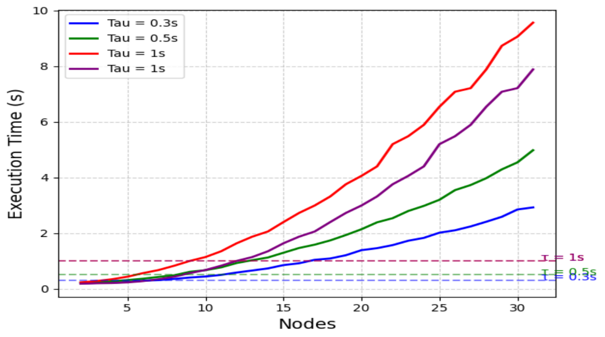


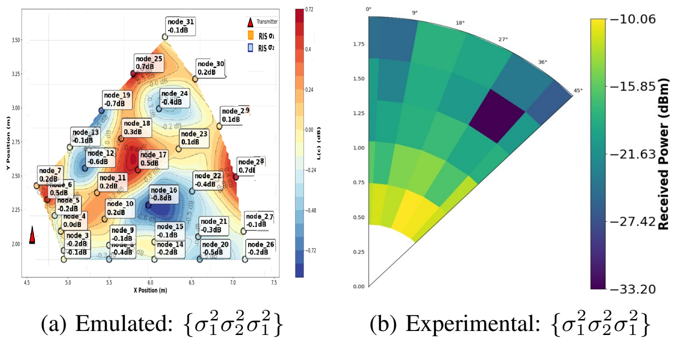
Fig. 3: Emulation Time vs. Number of nodes on Tyrone Server

To validate the Emulator's near-Real-Time performance, we conduct profiling on a high-performance Tyrone server to determine the maximum number of simultaneous nodes pairs it can support while satisfying Real-Time constraints. The System specifications are: **CPU:** $2\times$Intel Xeon Gold $6338(32$ cores each), **RAM:** 192GB, **OS:** Ubuntu 22.04 LTS.

To assess scalability, we incrementally increase the number of nodes in the Emulation Environment and measure the wall-clock time T_{sim} required to process a fixed Emulation window of duration τ seconds. For each configuration, a pure tone cosine transceiver is simulated per node request. The Emulator is then executed for a single frame of duration τ, and T_{sim} is recorded. Real-Time operation is deemed feasible if $T_{\text{sim}} \leq \tau$. The number of node pairs is increased until this condition is no longer met, marking the upper limit of Real-Time performance.

As shown in Figure 3, the Emulator maintains Real-Time performance up to **6**, **8**, and **10** nodes for $\tau = 0.3$, 0.5, and 1 seconds (red curve), respectively. By disabling selective pass-band functionalities, the Emulator achieves near-Real-Time performance for up to **12** nodes at $\tau = 1$ second (purple curve). The τ can be selected based on the desired trade-off between near-Real-Time performance and System complexity. While a fixed value of τ is used in the current implementation, various strategies for optimizing its selection can further enhance performance and will be explored in future work. These results demonstrate the Emulator's scalability and cost efficiency. By replacing expensive hardware SDRs with software-based Emulation, the platform significantly reduces the cost and complexity of large-scale Wireless experimentation.

C. Validation with Simulation and Experimental Results



(a) Emulated: $\{\sigma_1^2\sigma_2^2\sigma_1^2\}$ (b) Experimental: $\{\sigma_1^2\sigma_2^2\sigma_1^2\}$

Fig. 4: Comparison of received power distributions.

To validate the efficacy of the Emulator in replicating a practical RIS behaviour in the emulated Wireless Environment, we compare the received power distribution generated by the Emulator with that obtained from experimental measurements. The aim is to validate that the Emulator captures the spatial characteristics introduced by different RIS phase configurations.

The experimental setup consists of a Transmitter-Receiver (Tx-Rx) pair (B200 USRPs). The Rx is moved across a polar grid sector spanning $45°$, with a radial spacing of $30\,\text{cm}$ and an angular resolution of $10°$. Two types of fixed-configured passive RISs are considered, each comprising a uniform 5×8 element subarray. All 40 elements within a given RIS are assigned the same fixed phase value, chosen from the set $\{\sigma_1, \sigma_2\}$, where $\sigma_1 = 49.64°$ and $\sigma_2 = -153.77°$. Thus, each RIS is configured with either σ_1 or σ_2 uniformly across all elements.

The Emulation scenario consists of one Tx and 30 Rxs positioned according to the same polar grid used for the Rx movement in the experimental setup. The Tx transmits a single tone cosine signal at a carrier frequency of $f_c = 3.5\,\text{GHz}$, consistent with the experimental setup. The received power at each Rx location is calculated by aggregating contributions from the LOS path and non-LOS path from the RIS.

Figure 4 illustrates the power distribution generated by the Emulator for a configuration $\{\sigma_1^2\sigma_2^2\sigma_1^2\}$. The Emulator outputs gain of the total received signal relative to the LOS component, while the measurements report absolute received power.

Despite variations in absolute power levels, the Emulator accurately replicates key directional trends observed in the experiments. For instance, the configuration $\{\sigma_1^2\sigma_2^2\sigma_1^2\}$ produces peak gain near $25°$, with constructive interference concentrated in mid-sector angles and attenuation at the boundaries. These observations confirm that the Emulator captures the spatial behavior induced by RIS configurations, making it a practical and low-cost alternative to full-scale hardware Validation. By eliminating the need for anechoic chambers or multiple USRP devices, the Emulator provides a scalable solution for early-stage RIS prototyping and performance analysis.

IV. SRE for RIS-Assisted FM Coverage

Frequency Modulation (FM), originally formalized by Van der Pol in 1946 [31], continues to be a widely used analog radio transmission standard due to its resilience to noise and superior audio quality compared to Amplitude Modulation (AM). Despite the projected global FM radio audience exceeding three billion by 2029 [32], coverage remains inconsistent in rural and obstructed areas where terrain and infrastructure limitations degrade signal strength [33]. Traditional solutions such as high-power transmitters or relay stations can mitigate these gaps but are often impractical due to cost, power demands, and maintenance complexity [34], [35]. Field studies in regions like Niger State, Nigeria [36], and Jharkhand, India [37], confirm that coverage remains poor in underserved areas, emphasizing the need for passive, energy-efficient alternatives.

Recent advances in low-frequency Antenna miniaturization, including the development of compact 100 MHz patch antennas using U-slots and shorting walls [38], demonstrate the feasibility of designing electrically small structures for FM-band applications [39]. These principles can be extended to the development of passive RIS, enabling FM signal redirection without the need for active amplification [4]. RIS technology offers a promising low-cost solution to extend FM coverage by intelligently reflecting and phase-aligning signals toward target regions.

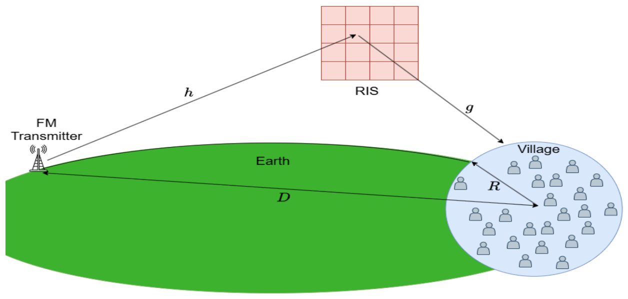


Fig. 5: SRE-based FM coverage scenario for a rural village.

We consider a rural village of radius R located at a distance D kilometers from a 100 MHz FM radio transmitter. Due to geographic obstructions such as hills and vegetation, direct signal propagation is heavily attenuated, resulting in degraded FM reception. To improve coverage, a passive RIS with static phase profile is deployed at an intermediate location between the transmitter and the village to reflect incoming FM signals toward the intended area. The RIS operates passively without dynamic reconfiguration during the Emulation.

The simulation is defined in a three-dimensional Cartesian coordinate System, where the FM transmitter is placed at coordinates $(0, 200, 100)$ meters. A planar RIS is positioned at a fixed location centered at $(2000, 34.9, 52014.9)$ meters—strategically positioned to serve a rural village area centered approximately at 52 km along the z-axis with $R = 2$ km. The RIS consists of a 30×30 array of unit cells, each measuring 0.25×0.25 meters. These parameters are selected to ensure sufficient aperture for effective reflection at FM frequencies while maintaining structural feasibility for large-area deployment in rural Environments.

Fig. 6 compares the Received Signal Strength under Line-Of-Sight (LOS) and total signal (combined LOS + non-line-of-sight (NLOS) conditions). While the LOS curve exhibits a smooth decay due to path loss, the NLOS-enhanced profile, which incorporates RIS-assisted reflections, demonstrates constructive total signal enhancements between 50 and 58 km. These fluctuations result in received power levels remaining above the -100 dBm sensitivity threshold, even at distances where LOS alone would fail to maintain connectivity. This confirms the RIS's ability to extend coverage in challenging Environments without active amplification, *and also demonstrates the versatility and applicability of the Emulator reported in this paper to solving Real-World Large-Scale Problems*.

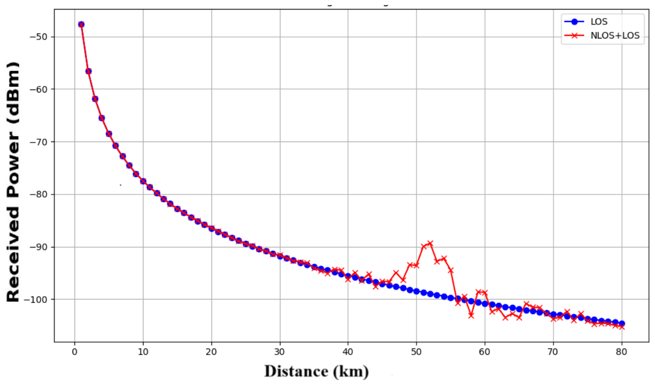


Fig. 6: Received signal power under LOS and total signal.

V. Conclusion

This paper presented the Design and evaluation of an open-source, near-Real-Time Emulator for SREs with RIS. The Emulator accurately replicates Wireless Environments using USRP-based SDRs with Real-Time Traffic and mobility support, and allows Dynamic RIS configuration through Integration with existing Testbeds. Validation against Laboratory measurements with a practical in-house-fabricated RIS demonstrated the Emulator's fidelity. Furthermore, its application in planning and dimensioning a RIS-assisted long-distance FM Broadcast System for rural coverage confirmed its utility. The Emulator preserves essential signal characteristics, including carrier frequency offset, and provides a flexible, scalable platform for realistic and reproducible RIS experimentation. Future work will focus on enhancing scalability.

Appendix

The simulator is organized into directories as shown below:

```
/
├─Core/
│  └─Engine.py
└─Config/
   ├─Nodes.json
   └─Output.json
```

```
|
|   |
|   |__RIS.json
|   |__Environment.json
|__External_traffic/
|__Modules/
    |__Channel_functions.py
    |__Mobility_functions.py
    |__Simulator_functions.py
```

- **Core/Engine.py:** Implements the main simulation loop operating at discrete time steps τ, updating mobility states, processing I/O requests, and handling signal propagation using channel models and RIS configurations.
- **Config/Nodes.json:** Specifies node configurations including node ID, location, mobility model, and center frequency. Used to initialize and update simulation state with IQ data and operational modes.
- **Config/Output.json:** Stores the runtime output of the Simulator, including received IQ samples, node states, and event logs.
- **Config/RIS.json:** Specifies RIS parameters including operating frequency, configuration ID, phase and gain response maps, angle of incidence and reflection. Supports programmable phase profiles and Real-Time reconfiguration via standardized interfaces.
- **Config/Environment.json:** Defines the physical layout of the simulation Environment.
- **External_traffic:** Emulates SDR APIs at the baseband level. The *Send_to_sim(Data, Operating Frequency, Sample Rate, Node ID)* and *Receive_from_sim(Sample Size, Operating Frequency, Sample Rate, Node ID)* functions enable nodes to issue transmission and reception requests, which are logged and updated in `Config/Nodes.json` for simulation processing.
- **Modules/ Channel_functions.py:** Implements LOS and RIS-assisted NLOS propagation models, as referenced in Sec. II. The Emulation abstracts key processes including Digital Up/Down-conversion, Analog-to-Digital and Digital-to-Analog conversion, frequency mixing, filtering, and path loss, along with noise addition.
- **Modules/Mobility_functions.py:** Defines multiple mobility models, including static, random walk, random waypoint, and Gauss-Markov model. Updates node positions in `Config/Nodes.json` and enforces spatial constraints using the simulation boundaries defined in `Config/Environment.json`.
- **Modules/Simulator_functions.py:** Provides utility functions for Simulator I/O via JSON state files, supporting external baseband signal generation and reception requests from `External_traffic`, node state querying, and signal interpolation.

References

[1] "Recommendation ITU-R M.2160: Framework & overall objectives of the future development of IMT for 2030 & beyond."

[2] M. Di Renzo, A. Zappone, M.-S. Debbah, Merouane Alouini, and J. T. S. Yuen, Chau de Rosny, "SRE empowered by RIS: How it works, state of research, & the road ahead," *Selected Areas in Communications*, 2020.

[3] Q. Wu and R. Zhang, "Towards smart and reconfigurable environment: IRS aided wireless network," *IEEE Communications Magazine*, 2020.

[4] E. Basar, M. Di Renzo, J. De Rosny, M. Debbah, M.-S. Alouini, and R. Zhang, "Wireless communications through RIS," *IEEE Access*, 2019.

[5] P. K. Gkonis, P. T. Trakadas, and D. I. Kaklamani, "A comprehensive study on simulation techniques for 5G networks: State of the art results, analysis, and future challenges," *Electronics*.

[6] A. S. Toor and A. Jain, "A survey on wireless network simulators," *Bulletin of Electrical Engineering*, 2017.

[7] J. Gomez, E. F. Kfoury, J. Crichigno, and G. Srivastava, "A survey on network simulators, emulators, and testbeds used for research and education," *Computer Networks*, 2023.

[8] ETSI, "Reconfigurable intelligent surfaces (RIS); use cases, deployment scenarios and requirements," Tech. Rep. RIS 001 V1.2.1, February 2025.

[9] TSDSI, "Methods and interface design for RIS-assisted communication systems," Technical Specification TSDSI STD 5003 V1.0.0, 2023.

[10] Y. Z. J. He, "RISTA – RIS technology white paper," Tech. Rep., 2023.

[11] "D3.2 initial report on the RIS controller and paas abstraction enablers," 6G-BRICKS Project, Tech. Rep., 2024.

[12] R. suppressed for double blind review.

[13] Reference suppressed for double blind review.

[14] E. Basar and I. Yildirim, "SimRIS channel simulator for RIS-empowered communication systems," in *2020 IEEE (LATINCOM)*.

[15] I. Burtakov, A. Kureev, A. Tyarin, and E. Khorov, "QRIS: A quadriga-based simulation platform for RIS," *IEEE Access*, 2023.

[16] M. Segata, P. Casari, M. Lestas, A. Papadopoulos, D. Tyrovolas, T. Saeed, G. Karagiannidis, and C. Liaskos, "CoopeRIS: A framework for the simulation of RIS in cooperative driving environments," *Computer Networks*.

[17] A. Habib, I. Khaled, A. El Falou, and C. Langlais, "Extended NYUSIM-based mmwave channel model and simulator for RIS-assisted systems," in *2023 (EuCNC/6G Summit)*.

[18] E. Björnson, H. Wymeersch, B. Matthiesen, P. Popovski, L. Sanguinetti, and E. de Carvalho, "RIS: A signal processing perspective with wireless applications," *IEEE Signal Processing Magazine*, 2022.

[19] J. Xu, Y. Liu, X. Mu, and O. A. Dobre, "STAR-RISs: Simultaneous transmitting and reflecting RISs," *IEEE Communications Letters*, 2021.

[20] F. HHI, "Fraunhofer ray tracing tool for RIS," 2022.

[21] A. Habib, C. Langlais, A. El Falou, S. Kangoute, Y. Wang, and M. Berbineau, "Integration of RIS into CloudRT simulator for railway tunnel scenarios," in *2025 19th (EuCAP)*.

[22] S. E. Ghalbzouri, K. Boutiba, A. Ksentini, and M. Benjillali, "Neural-driven control of RIS in 6G networks: A GoSimRIS and xApp-based framework," *IEEE Networking Letters*, 2025.

[23] H. Jiang, L. Dai, M. Hao, and R. MacKenzie, "End-to-end learning for RIS-aided communication systems," *IEEE Transactions on Vehicular Technology*, 2022.

[24] M. W. Lichtman, *PySDR: A Guide to SDR and DSP using Python*, 2020.

[25] C. Bettstetter, G. Resta, and P. Santi, "The node distribution of the random waypoint mobility model for wireless ad hoc networks," *IEEE Transactions on Mobile Computing*, 2003.

[26] P. Nain, D. Towsley, B. Liu, and Z. Liu, "Properties of random direction models," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, 2005.

[27] J. Ariyakhajorn, P. Wannawilai, and C. Sathitwiriyawong, "A comparative study of random waypoint and gauss-markov mobility models in the performance evaluation of manet," in *2006 International Symposium on Communications and Information Technologies*.

[28] K.-H. Chiang and N. Shenoy, "A 2-d random-walk mobility model for location-management studies in wireless networks," *IEEE Transactions on Vehicular Technology*, 2004.

[29] S. W. Ellingson, "Path loss in RIS-enabled channels," in *Proc. IEEE PIMRC*, 2021.

[30] G. Xing, M. Shen, and H. Liu, "Frequency offset and I/Q imbalance compensation for direct-conversion receivers," *IEEE Transactions on Wireless Communications*, 2005.

[31] B. Van der Pol, "The fundamental principles of FM," *Journal of the Institution of EE-Part III: Radio and Communication Engineering*, 1946.

[32] "Radio industry statistics (toneisland)."

[33] Wikipedia contributors, "Fm broadcasting – reception distance," 2024.

[34] "Lpfm economic considerations."

[35] "Broadcast relay station."

[36] O. Onuike, O. Oladipo, A. Ahmed, and A. Lawal, "Spatial coverage of fm radio transmitters in niger state, nigeria," *International Journal of Engineering and Technology*, 2013.

[37] V. Pavarala, "Building solidarities: A case of community radio in jharkhand," *Economic and Political Weekly*, 2003.

[38] M. Ali and I. Khan, "Wearable antenna design for fm radio," *Arabian Journal for Science and Engineering*, 2014.

[39] A. Tatarenko, R. Petrov, and G. Srinivasan, "A 100 mhz antenna based on magnetoelectric composite materials," 2008.