Exploring 5G with Open Source Solutions: Functional Testbed and User Equipment Evaluation

Bruno Tió, Facundo Pedreira, Leonardo Barboni and Claudina Rattaro
Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay
{brunot,facundo.pedreira,lbarboni,crattaro}@fing.edu.uy

***Abstract*—This work presents the successful implementation of a functional 5G testbed based on the open-source project srsRAN. The network was deployed in Standalone (SA) mode, integrating an operational Open5GS core and a working gNB, enabling full 5G connectivity. The gNB's radio interface was implemented using Software-Defined Radio (SDR) hardware, ensuring flexibility and reconfigurability of the RF components. Connections were established with multiple user equipment (UE) devices. The testbed successfully connected to a commercial off-the-shelf (COTS) device using a programmable physical SIM, allowing stable operation and real traffic exchange. A Sixfab 5G HAT, based on a Qualcomm modem, was also acquired and configured to validate connectivity from typical IoT platforms. Additionally, one of the UEs was implemented using SDR-based hardware, demonstrating the potential of open and flexible radio solutions for both infrastructure and user-side components.**

***Index Terms*—5G testbed, Software-Defined Radio (SDR), srsRAN, IoT.**

I. Introduction

The world of mobile networks is in constant evolution, with the deployment of fifth-generation (5G) systems emerging as a global focal point in recent years. With enhanced Mobile Broadband (eMBB), 5G enables high data transmission rates that support demanding applications such as virtual and augmented reality. Ultra-Reliable and Low-Latency Communications (URLLC) open the door to critical services, including autonomous vehicles and telemedicine, while massive Machine-Type Communications (mMTC) facilitate the deployment of Internet of Things (IoT) ecosystems with billions of simultaneously connected devices [1], [2].

Despite these advances, several technological challenges remain to fully realize these use cases—challenges that are expected to extend into the development of sixth-generation (6G) networks. These include the need for intelligent and efficient resource management to meet Quality of Service (QoS) requirements, the deployment of network slicing across all network segments, interoperability among heterogeneous systems, and robust approaches to cybersecurity threats, among others [3], [4].

Although recent progress has been made, many real-world mobile network deployments still depend on proprietary software and expensive hardware, creating a substantial barrier for research groups with limited resources. In response, open-source alternatives such as srsRAN and OpenAirInterface have emerged, offering robust tools for the local and autonomous implementation of both 5G NR and 4G LTE networks. These platforms democratize access to mobile network technologies, fostering innovation and enabling broader participation in research and education [5], [6].

OpenAirInterface (OAI) is maintained by the OpenAirInterface Software Alliance (OSA)—a global consortium of companies, universities, and institutions including major telecom players like Nokia, Vodafone, and Ericsson. The alliance actively collaborates with the 3rd Generation Partnership Project (3GPP), which defines global telecom standards. On the other hand, srsRAN is developed by Software Radio Systems (SRS), a private company that maintains both the open-source version and a commercial-grade edition with enterprise support. srsRAN 5G follows their earlier open-source LTE implementation.

Having an open, reconfigurable 5G testbed is essential for universities, research labs, and operators exploring advanced functionalities, interoperability, or emerging frameworks such as O-RAN. These environments make it possible to validate configurations, assess performance under varied conditions, and lower the entry barrier to technologies traditionally reserved for large vendors.

One of the main advantages of open-source testbeds like srsRAN lies in their ability to run on low-cost, low-power platforms—such as mini-PCs or embedded systems with virtualization support. This flexibility enables the design of portable and scalable networks deployable across diverse scenarios, from academic labs to production environments, covering multiple industry applications.

The main contribution of this work is the implementation of a fully functional end-to-end 5G Standalone (SA) testbed, entirely based on open-source solutions and general-purpose hardware. In addition, the performance of various user equipment (UE) was evaluated in practice, including both commercial off-the-shelf (COTS) devices and embedded platforms. Notably, the Sixfab 5G HAT—a recently released board based on a Qualcomm modem designed to add 5G connectivity to platforms like the Raspberry Pi—was successfully integrated and tested. To the best of our knowledge, no prior studies have documented the use of this device in experimental 5G SA environments. Its inclusion provides concrete evidence of the feasibility of using low-cost, low-power devices in 5G connectivity testing, particularly in IoT scenarios and portable deployments.

II. Deployment of a 5G Testbed Architecture - Hardware and Software Setup

This section defines the main components of the testbed architecture, highlighting the most relevant aspects related to software, hardware, and configuration. The section is organized into three parts: the core network, the base station (gNB), and the user equipment (UE). Special attention is given to the latter, where the different evaluated variants are described in detail.

A. Core Network

Since srsRAN does not include its own core network implementation, this testbed uses Open5GS for deploying the 5G core. Open5GS can be installed directly on the host, within virtual machines, or via containerized deployments. In this case, a container-based deployment was selected, using Docker Compose to efficiently orchestrate and manage all core services. In particular, authors of [7] present a 5G SA platform based on Open5GS and srsRAN, which served as one of the key references used to configure and implement our testbed. Among the most important configurations to be made is the definition of the UE list, which is specified in a CSV file. This file includes essential parameters such as the IMSI, Key/OPC pair, and a static IP address for each UE. This configuration enables proper authentication and traffic control during network access.

B. Base Station Implementation and Frequency Configuration

To deploy the base station (gNB), we used the implementation provided by the srsRAN Project. This solution adheres to the principles of the O-RAN architecture and supports various functional split options as defined by the 3GPP [8]. In particular, it enables Option 8, which allows the gNB components—Central Unit (CU) and Distributed Unit (DU)—to be deployed either jointly or separately, while the Radio Unit (RU) functionality is delegated to the Software-Defined Radio (SDR, [9]) device. Band n3 from Frequency Range 1 (FR1) was configured for use, specifying both the band number and the Absolute Radio Frequency Channel Number (ARFCN). This band was selected due to its compatibility with the gNB and UE implementations provided by srsRAN. The core network and gNB were hosted on an Intel i7-12700 server with 16 GB of RAM, running Ubuntu 22.04 LTS. For the gNB's radio interface, two Software-Defined Radio (SDR) devices were used: the high-end USRP X310 and the mid-range USRP B200. In both cases, WA5VJB log-periodic antennas were employed. Although the X310 supports more configurations than the B200, due to limitations of srsUE in our setup, there is no practical difference between using one or the other in this case.

Up to this point, we have defined the core and the gNB; now it's time to focus on the user equipment (UE).

C. User Equipment (UE)

The srsRAN Project does not provide a dedicated 5G UE implementation. However, its predecessor, srsRAN 4G, includes srsUE, a software-based UE emulator originally designed for LTE networks. It is compatible with Frequency Division Duplex (FDD) configurations, bandwidths from 1.4 to 20 MHz, supports up to 2x2 MIMO, and downlink modulation up to 256-QAM. A key limitation is its fixed 15 kHz subcarrier spacing, whereas 5G supports variable spacings up to 120 kHz. Although originally developed for LTE, srsUE includes the basic functionalities required to connect to a 5G SA network. However, it does not support the full set of 5G NR features defined in the standard. To configure this type of UE, network identification values such as the IMSI and Key/OPC pair can be specified directly within the configuration file used to launch the UE. Instead of requiring a physical SIM card, srsUE simulates SIM behavior using the provided identifiers. This UE was implemented using a PC with the same specifications as those used for the core and gNB. For the radio interface, a USRP B200 was employed.

Fig. 1. Sixfab 5G Development Kit installed on Raspberry Pi 4.

In addition, the project explored the use of Commercial Off-The-Shelf (COTS) UE devices. srsRAN documentation provides a list of tested 5G-compatible UEs, enabling real-world use cases with native 5G capabilities. In particular, a Samsung Galaxy A33 5G smartphone was used as the commercial UE. To integrate this device into the network, a non-commercial SIM card was used—specifically obtained and programmed for this purpose. The SIM was provided by the Open Cells project https://open-cells.com, which offers programmable SIMs along with the UICC/SIM Programming tool that enables configuration via a USB dongle.

Finally, we identified the Sixfab 5G Development Kit for Raspberry Pi as an intermediate solution between srsUE and COTS devices. While srsUE offers full control but limited realism, and COTS offers realism but little configurability, the Sixfab kit provides a practical compromise. Based on a Qualcomm technology, it combines a modular hardware interface with user-level configuration options. It also allowed us to evaluate the technical and economic feasibility of using such devices as 5G UEs in IoT or portable deployment scenarios.

The Sixfab Kit features a 5G HAT based on the Quectel RM502Q-AE module, offering sub-6 GHz 5G connectivity,

USB 3.1 and PCIe interfaces, GNSS support, and compatibility with Raspberry Pi 4 and 5. Our setup included both the Sixfab 5G module and a Raspberry Pi 4, which ran Ubuntu 22.04. Figure 1 shows the assembled Sixfab 5G Development Kit, featuring a base with an integrated fan and antennas. These components connect to the 5G HAT, which also includes a slot for a programmable SIM card, configured similarly to the one used in the COTS UE. The HAT connects to the Raspberry Pi via its GPIO pins and an additional USB interface. In this case, UE configuration was carried out using Minicom, a terminal utility provided by Sixfab.

III. Results and Analysis

Once the entire testbed was assembled and end-to-end connectivity was verified, comparative performance tests were conducted between the different UEs (see Figure 2). The tests included in this section use the B200 device as the gNB's RU.

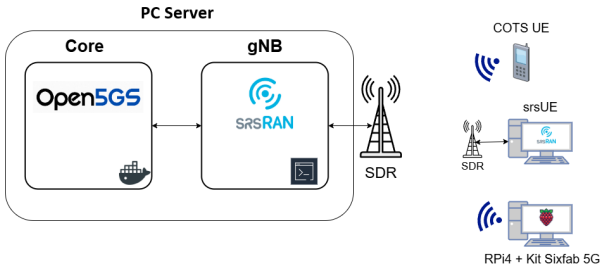

Fig. 2. 5G SA testbed.

A. Throughput Results by User Equipment

Each device was tested individually on the network, across the four available bandwidth configurations. The downlink recorded results are presented in Figure 3. The measurements were performed using iPerf with UDP traffic, running the iPerf server on the core network side and the client on each UE.

A clear performance gap was observed in favor of the commercial devices. The Samsung COTS UE consistently achieved the highest throughput across all scenarios. Nevertheless, the Sixfab 5G HAT for Raspberry Pi 4 delivered comparable results, with both significantly outperforming the srsUE-based solution. Moreover, achieving a stable connection between the srsUE and the gNB requires careful tuning of parameters such as gain settings, attenuation, and other radio configuration values.

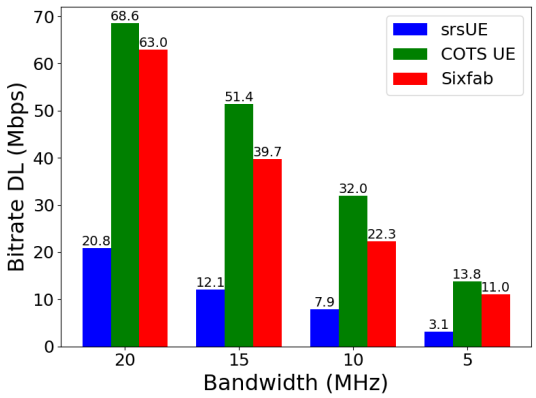

Fig. 3. Average downlink (DL) throughput for each UE at different bandwidths.

B. Throughput Results by Combination of User Equipments

After validating the functionality of each UE individually, additional tests were conducted with various combinations of UEs connected simultaneously to the network. This approach allowed us to evaluate the performance of the implemented gNB under multi-UE scenarios and to observe how each device behaved when sharing network resources. The results are presented in Tab. I.

TABLE I
Average DL and UL throughput per UE for different simultaneous connection scenarios.

UEs	Device	$\overline{Br_{DL}}$ (Mbps)	$\overline{Br_{UL}}$ (Mbps)
COTS + Sixfab	COTS UE	29.3	28.5
	Sixfab	27.9	27.5
srsUE + Sixfab	srsUE	14.4	14.1
	Sixfab	8.92	8.52
COTS + srsUE	COTS UE	15.6	14.9
	srsUE	35.1	34.8
All 3 UEs	Sixfab	9.43	9.12
	COTS UE	19.8	19.2
	srsUE	29.1	19.9

When multiple UEs were connected simultaneously, performance varied. For example, when the COTS UE and Sixfab transmitted in parallel, their individual bitrates dropped to nearly half compared to isolated tests—a predictable outcome given they share the same radio medium. However, results were more striking when the srsUE was involved: both the COTS UE and Sixfab experienced throughput drops exceeding 75%, performing noticeably worse than when used together without the srsUE. This behavior is attributed to the resource allocation mechanisms in srsRAN and the high RF gains required by the USRP B200 SDR used with the srsUE.

C. Internet Connectivity

Speed tests were conducted using Speedtest by Ookla, both via the web interface and the command-line tool for Linux (CLI). It is important to note that these results should be interpreted with caution, as the Internet access in this setup relied on a server PC (Core Network) connected to a public Wi-Fi network at the university. Since this connection is subject to congestion, bandwidth limitations, and latency variations, optimal performance was not expected.

Internet speed test results also revealed interesting variations. As shown in Figure 4, both the Sixfab and COTS UE showed consistent Internet performance. While the measured throughput does not increase strictly with bandwidth, it remains within expected variation margins. In contrast, the srsUE exhibited an unexpected peak at 10 MHz, followed by

a significant drop at higher bandwidths. This behavior lacked a clear explanation, although possible causes include USB 3.0 interface (fronthaul interface) saturation after repeated testing.

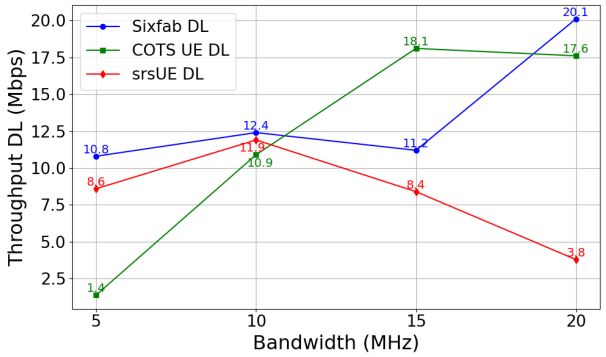

Fig. 4. Downlink throughput comparison from Speedtest measurements.

D. Qualitative Testing

We conducted real-world application tests to evaluate the practical usability of the testbed, focusing on three representative use cases: package downloads using `apt`, multimedia streaming (YouTube), and video calls between UEs. These cases cover distinct traffic patterns—bulk data transfer, adaptive streaming, and real-time bidirectional communication. The `apt` test involved downloading packages from the global repositories, offering a steady HTTP-based flow. YouTube streaming, with adaptive bitrate mechanisms, imposed more dynamic throughput demands. Video calling proved the most challenging, requiring low latency and sustained bidirectional flow, often revealing the testbed's performance limits.

To ensure consistent testing conditions, the same package was downloaded in each case, and the same video was played on the same platform during streaming tests. All experiments were performed on the same day. The COTS UE was unable to complete the `apt` test due to system restrictions imposed by the manufacturer. Although quantitative data was collected where possible, the following analysis is mostly qualitative due to the subjective nature of user experience and variability in real-time conditions. The focus was placed on user perception, identifying major difficulties, responsiveness, and connection stability.

In the `apt` package download test, the reference server achieved a throughput of 160 kB/s. The COTS UE could not perform the download due to operating system restrictions. In contrast, the Sixfab and srsUE configurations achieved measurable performance, with download speeds of 39.9 kB/s and 24.3 kB/s, respectively. During video streaming, the COTS UE experienced no issues running the Android YouTube app, maintaining a consistent and smooth playback experience. However, the srsUE and Sixfab UE encountered different types of limitations. The srsUE suffered frequent disconnections during video loading via the YouTube web interface, requiring manual reconnection. In contrast, the Sixfab UE maintained a steady data stream but failed to properly render video in the browser, likely due to limitations of the Raspberry Pi. These issues persisted in the video call tests. The COTS UE provided the most stable experience. The Sixfab UE remained connected but failed to run Zoom effectively via the browser—video rendering delays caused by Firefox on Raspberry Pi with Ubuntu 22.04 led to noticeable quality issues. In contrast, the srsUE had no hardware limitations but continued to experience random disconnections during the calls.

IV. Conclusions

This work demonstrates the feasibility and potential of deploying a functional 5G SA testbed using open-source solutions and general-purpose hardware. By leveraging srsRAN and Open5GS, we successfully implemented a full 5G network supporting real data exchange with multiple types of user equipment. Our experiments show that commercial off-the-shelf (COTS) devices offer the highest performance, yet lower-cost and lower-power alternatives like the Sixfab 5G HAT also achieve stable operation and are viable for IoT and portable scenarios. Although the srsUE-based solution presented limitations in stability and performance, it still offers value in controlled experimental setups and educational environments. All configurations, test procedures, dependencies, and results are available in an open-access repository (https://gitlab.fing.edu.uy/portaran/portaran_srsran), ensuring transparency and reproducibility of the work.

Acknowledgment

This work was partially supported by CSIC R&D project "5/6G Optical Network Convergence: an holistic view".

References

[1] ITU-R, "Recommendation itu-r m.2083-0: Imt vision – framework and overall objectives of the future development of imt for 2020 and beyond," https://www.itu.int/rec/R-REC-M.2083, 2015, accessed: 2025-04-01.

[2] E. D. S. P. J. Skold, *5g NR: The Next Generation Wireless Access Technology*. Academic Press, 2018. [Online]. Available: libgen.li/file.php?md5=609d03b59709b6cd73adccb1e47a68e1

[3] S. Dang, O. Amin, B. Shihada, and M.-S. Alouini, "On challenges of sixth-generation (6g) wireless networks: A comprehensive survey of requirements, applications, and security issues," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 6–36, 2021.

[4] M. S. Akbar, Z. Hussain, M. Ikram, Q. Z. Sheng, and S. C. Mukhopadhyay, "On challenges of sixth-generation (6g) wireless networks: A comprehensive survey of requirements, applications, and security issues," *Journal of Network and Computer Applications*, vol. 233, p. 104040, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804524002170

[5] J. E. Håkegård, H. Lundkvist, A. Rauniyar, and P. Morris, "Performance evaluation of an open source implementation of a 5g standalone platform," *IEEE Access*, vol. 12, pp. 25 809–25 819, 2024.

[6] P. Vázquez, W. Peña, W. Piastri, L. Inglés, and C. Rattaro, "Maq5g: Deployment of a complete 5g standalone network testbed for testing and development," in *2024 XVI Congreso de Tecnología, Aprendizaje y Enseñanza de la Electrónica (TAEE)*, 2024, pp. 1–7.

[7] L. Mamushiane, A. Lysko, H. Kobo, and J. Mwangama, "Deploying a stable 5g sa testbed using srsran and open5gs: Ue integration and troubleshooting towards network slicing," in *2023 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, 2023, pp. 1–10.

[8] L. M. P. Larsen, A. Checko, and H. L. Christiansen, "A survey of the functional splits proposed for 5g mobile crosshaul networks," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 146–172, 2019.

[9] J. Mitola and G. Maguire, "Cognitive radio: making software radios more personal," *IEEE Personal Communications*, vol. 6, no. 4, pp. 13–18, 1999.