RemoteRF: An Open-Source Platform to Democratize Access to Software-Defined Radios in Wireless Research and Education

Ethan Y. Ge and Ian P. Roberts
University of California, Los Angeles
www.remoterf.net

***Abstract*—Software-defined radios (SDRs) are powerful tools for research and education in wireless communications, but their cost and complexity put them out of reach for many universities and researchers worldwide. To address this, we introduce RemoteRF, a platform for creating large-scale testbeds of distributed SDRs that are centrally managed by a single server. These SDRs can be remotely accessed by users over the internet, allowing them to conduct wireless experiments at any time from virtually anywhere, as long as they have a network connection. When used in research, RemoteRF can be used to develop and experimentally evaluate new communication techniques or to collect real-world data to train and test machine learning models. When used in education, RemoteRF can allow students in virtually any sized class to share a handful of SDRs to complete active learning lab exercises that parallel course lectures. In an effort to democratize access to SDRs across the globe, the software powering RemoteRF has been made open-source and is extensively documented, allowing anyone to deploy their own instance today in a matter of minutes. Over the past year or so, RemoteRF has been used in both teaching and research at UCLA, where it has logged nearly 4,000 hours of use by more than 200 students and researchers to date.**

***Index Terms*—Software-defined radios, platforms, testbeds, prototyping, measurements, communications, sensing, education.**

I. Introduction

For decades, mathematical modeling and simulation have been the lone tools used by countless researchers to invent new wireless communication techniques. Similarly, engineering students have long been taught the fundamentals of wireless communication through blackboard lectures and problem-solving built on idealized theory. While this approach to both research and training has yielded the incredible wireless networks we have today, innovation is slowing by several measures and this has raised many questions about future advancements in wireless [1]. Machine learning (ML) and artificial intelligence (AI) have been a recent source of innovation but have largely been trained and tested on synthetic wireless data so far—again, often obtained using idealized mathematical models and simulation tools.

While sound models and simulation tools will forever play important roles in wireless, incorporating actual hardware platforms into research and education stands to revolutionize the future of the field [2]. Instruments called *software-defined radios (SDRs)*, for instance, can be used in research to take measurements, experimentally evaluate new techniques, and collect real-world data to train and test ML/AI models [3]. In education, SDRs can be used by students to complete *active learning* lab exercises where they implement fundamental concepts on actual radios, preparing them for careers in industry where they are tasked with translating ideas from theory to practice [4].

Key Barriers to Entry: Cost, Complexity, and Coordination

Although SDRs are not perfect representations of real-world wireless deployments, their value has been realized by many researchers, educators, and practicing engineers worldwide [4], [5]. The vast majority of wireless researchers and engineering students across the globe, however, do not have access to such instruments, and this is largely due to cost. A single SDR transceiver can cost anywhere from a few hundred to tens of thousands of US dollars (USD), putting even the most affordable models out of reach for many, especially when many SDRs are needed for research or large class sizes. Beyond cost, setting up the necessary drivers/dependencies and writing code to actually use SDRs introduces additional hurdles, particularly for first-time users. This worsens when conducting experiments that require coordination across multiple SDRs at once, especially in large-scale testbeds where SDRs are geographically distributed. Altogether, these financial and logistical challenges present substantial barriers to entry in incorporating SDRs into research or education, stifling progress in the field of wireless.

Where Existing Solutions Fall Short

Several have sought to address this problem through the creation of shared platforms that allow users to *remotely* access SDRs over the internet. Among the most popular are large-scale testbeds supported through the US National Science Foundation's PAWR program [6]—such as POWDER [7], COSMOS [8], and AERPAW [9]—which have been created primarily for research use, not education. These platforms have adopted remote-execution workflows in which users develop and run code within the testbed's own infrastructure, accessing allocated resources over SSH or through provisioned virtual machines (VMs). Other university-scale testbeds have taken a similar approach, with the ORBIT radio grid at Rutgers [10] and the Arena testbed at Northeastern [11] both relying on server-side execution of users' code. In the educational domain, platforms such as WebLab-Deusto [12] and the RHLab

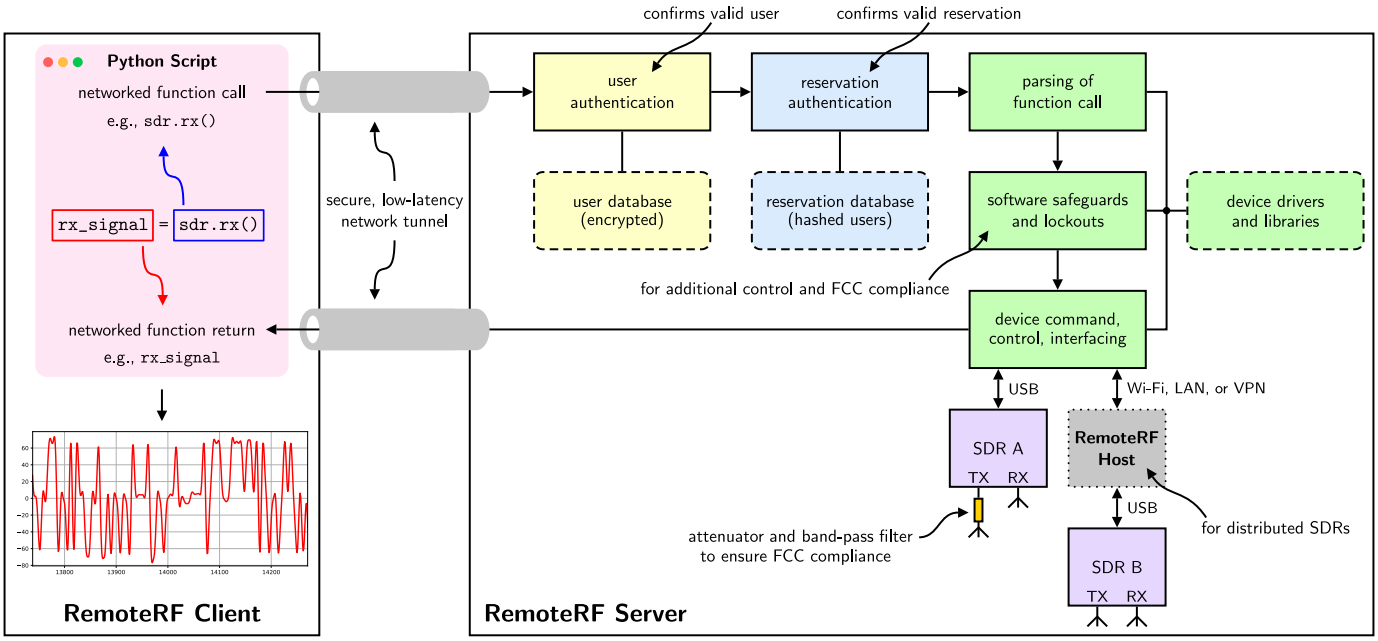

Fig. 1. Through simple Python scripting, a user can remotely access SDRs connected to the RemoteRF server deployed on their university campus or in their research lab. Most of this Python script executes locally on the user's personal computer, except for function calls that involve SDRs, which get routed over the network to the RemoteRF server. There, they are executed on the actual SDRs, and any return values (such as received signals) are routed back to the user's Python script, where they can be further processed locally.

remote SDR laboratory [13] similarly route user workflows through remote servers or VMs. A noteworthy drawback of all these existing approaches is that they rely on execution of code on the platform itself, which hinders users' ability to quickly iterate and inevitably shapes their experience around the constraints of the platform itself, rather than their own local tools. Further, all these platforms are either closed-source, fairly complex to deploy, or lack deployment documentation altogether, which prevents others from replicating them at their own institutions.

II. The RemoteRF Platform

This article introduces the RemoteRF platform, which aims to dramatically lower the barrier to entry for those wishing to incorporate SDRs into their research or instruction, especially at scale. RemoteRF accomplishes this by providing a systematic way to centrally manage and remotely access SDRs over a network connection. This allows researchers or university instructors to create large-scale testbeds of multiple SDRs that can be seamlessly shared across many users, such as members of a research group or students in an engineering course. The principal advantages of RemoteRF over existing alternatives are its low cost, simplicity, open-source software, and dual use in both research and education.

Overview of RemoteRF

The RemoteRF platform is comprised of three distinct components, each of which can be found in the functional block diagram of Fig. 1.

- **Server:** The RemoteRF server would usually be deployed on a university campus or in a research lab and would have one or more SDRs connected to it.
- **Client:** A RemoteRF client connects to the RemoteRF server and enables a user to remotely access the server's SDRs through Python via a network connection.
- **Host:** A RemoteRF host is an optional component that provides a virtual connection between one or more SDRs and a RemoteRF server over Wi-Fi, LAN, or virtual private network (VPN), rather than over a direct wired connection such as USB. This enables the creation of large-scale testbeds with *distributed* SDRs.

A typical RemoteRF deployment would consist of one server, many clients, and optionally one or more hosts. Setting up a RemoteRF server amounts to acquiring the desired SDRs and connecting them to a modest Linux-based computer with the server software installed. Directly connecting the SDRs to the server over a wired connection (such as USB) bypasses the need for RemoteRF hosts; for distributed testbeds, host software can be installed on lightweight, single-board computers—such as the Raspberry Pi—to provide a virtual connection between one or more SDRs and the RemoteRF server via Wi-Fi, LAN, or VPN. RemoteRF client software would be installed on users' personal machines, providing them with a command line interface (CLI) to connect to the server and a Python library to interface with its SDRs.

Preserving Local Experimentation

A key design goal of RemoteRF was to make remote SDR use feel as close as possible to *local* experimentation, i.e., as if SDRs were directly connected to the user's machine. To accomplish this, RemoteRF has been designed such that a user can remotely access SDRs over the network through Python scripts that closely resemble those they would write if the SDRs were directly connected to their personal computer.

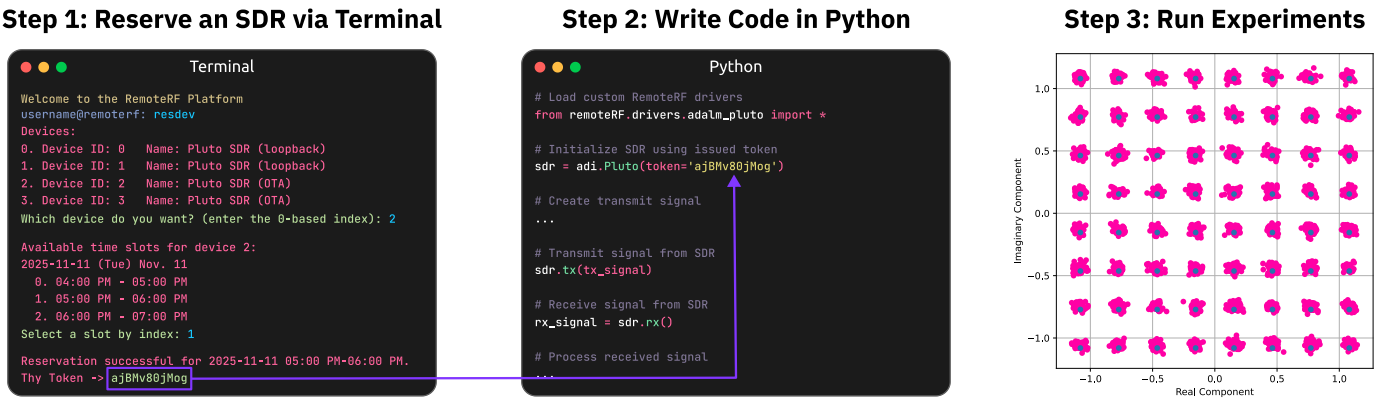

Fig. 2. Typical use of RemoteRF involves a user reserving access to a particular SDR, writing Python code, and then running the code. The majority of the Python code runs locally on the user's personal computer; only function calls that involve an SDR (e.g., to transmit/receive signals) are routed through the RemoteRF platform.

Upon running these Python scripts on their personal computers, most of the code executes locally, but any commands that involve SDRs are routed to the RemoteRF server for execution; any return values are passed back to the user for processing, plotting, and analysis. In stark contrast to existing platforms [7]–[13], this allows a user to develop and debug code on their own personal computer, rather than within some VM or over SSH, allowing them to take advantage of local software tools and compute resources (such as GPUs). This also has the added benefit of reducing the computational load of the RemoteRF server, allowing it to accommodate more users and SDRs.

Reservation-Based Access to SDRs

A central component of the RemoteRF platform is its reservation framework. Rather than allow users to access SDRs on a first-come, first-served basis, they must reserve SDRs during desired time slots via a CLI provided by the RemoteRF software package. Upon making a reservation, the server issues the user a unique token (a random string of characters) that provides them exclusive access to a specific SDR during the reservation interval, as shown in Fig. 2. This framework enables many users to systematically share limited hardware resources and plan experiments in advance. To further ensure fairness and control access within this framework, administrators can limit the reservation duration and maximum number of reservations each user may hold.

User Groups and Enrollment Codes

Given the shared nature of the RemoteRF platform, it is natural for an administrator to want to provide different levels of access for different groups of users. *User groups* can serve such a purpose, where each group can be given unique permissions and privileges, such as access to certain SDRs. When a user creates their RemoteRF account for the first time, they are asked to enter an *enrollment code*, which automatically assigns them to the user group attached to that enrollment code. In research, user groups can be used to restrict researchers' access to the particular SDRs needed to complete their research experiments. In education, user groups can give students across different courses unique permissions and SDR access. User groups and enrollment codes can both be made to expire after a certain duration, if desired, in order to further control access to RemoteRF.

Large-Scale Testbeds

To maximize its utility, RemoteRF supports the creation of large-scale testbeds through custom host software that can be run on lightweight, single-board Linux computers, such as the Raspberry Pi. A single RemoteRF host provides a virtual connection between the RemoteRF server and one or more SDRs over Wi-Fi, LAN, or VPN, meaning SDRs can be geographically distributed from the server yet still accessible, as long as they have a network connection and a power source. As depicted in Fig. 3, this enables the creation of testbeds that span entire rooms, buildings, or even campuses, with each node requiring minimal cost and setup to deploy. From a user's perspective, these distributed SDRs appear and function exactly as if they were directly connected to the RemoteRF server over a wired connection.

Custom Device Support and Software Lockouts

Although RemoteRF natively supports multiple commonly used SDRs out of the box, it is designed to also allow administrators to onboard SDRs that are not natively supported. To do so, administrators can write simple Python wrappers that define the functionality and properties of the SDRs they wish to onboard. These custom wrappers are to be written on the RemoteRF server by the administrator and would then be automatically pulled by the RemoteRF client behind the scenes, meaning no software updates are necessary nor changes on the end user's side. This makes it possible for administrators to expand their own RemoteRF deployment without any intervention from the RemoteRF development team. This also provides a systematic way for administrators to create high-level server-side commands that execute multiple functions or even entire scripts in order to simplify user-side commands. Furthermore, since actual control of SDRs happens server-side, custom lockouts can be implemented in software to limit SDR operation if desired. For instance, these can be

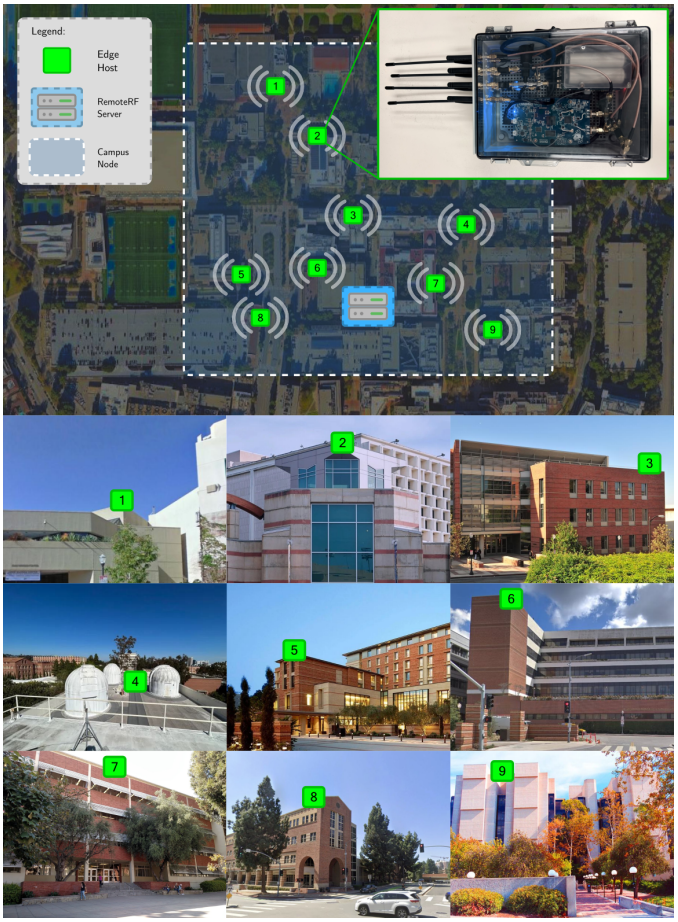

Fig. 3. RemoteRF enables the creation of large-scale testbeds of geographically distributed SDRs, such as the example deployment on the UCLA campus shown above. This is accomplished through lightweight host nodes like the Raspberry Pi, which provide a virtual connection between the RemoteRF server and distributed SDRs over Wi-Fi, LAN, or VPN.

used to limit the transmit power and carrier frequency of the SDRs to ensure that spectrum regulations are reliably met. Beyond these software safeguards, filters and attenuators may also be physically installed onto the SDRs to further guarantee regulatory compliance.

III. RemoteRF in Wireless Research and Engineering Education

RemoteRF has ripe applications in both wireless research and education, as both stand to benefit from incorporating SDRs. With this in mind, the platform was developed to be a single solution for both use cases, allowing a single RemoteRF deployment to fill both needs on a university campus.

Educational Use Case

With minimal cost and setup, a university instructor can deploy a RemoteRF server on their campus for use in their engineering courses, allowing students to remotely access its SDRs. In this context, RemoteRF would allow students in virtually any sized class to share access to a limited number of SDRs and complete lab exercises at any time from anywhere with an internet connection. In one class at UCLA, for instance, RemoteRF allowed sixty students to successfully share merely five SDRs to complete weekly lab exercises. This highlights the massive cost savings that RemoteRF can offer, compared to purchasing SDRs for each student in a course or holding in-person lab sessions, as illustrated in Fig. 4. It is also worth noting that the flexibility afforded by this anytime–anywhere access can be particularly valuable to non-traditional students, such as those with caregiving responsibilities, those that commute to campus, and those that take classes online.

Research Use Case

Similar to its role in education, a research group could deploy RemoteRF in their lab space to conduct experiments remotely and systematically share limited resources between lab members. Perhaps more exciting, though, is the potential to use RemoteRF to create large-scale testbeds which span buildings or even entire campuses that can be *centrally* controlled from a single RemoteRF server. This would allow researchers to develop and experimentally evaluate new techniques in a variety of real-world scenarios. Additionally, RemoteRF could provide a systematic way to automate the collection of measurements, e.g., to train and test ML/AI-based methods. In a similar vein, a large-scale RemoteRF testbed could be used to augment/validate ray-tracing datasets and digital twins, both of which are key ingredients in wireless research today [14].

IV. UCLA Case Study

The first RemoteRF prototype was deployed at UCLA in early 2025, where it has since been used in both education and research. This prototype began with five ADALM-PLUTO SDRs [15] directly connected to the RemoteRF server over USB and later grew to fourteen SDRs. Two of these SDRs are distributed around campus, each connected to a Raspberry Pi acting as a RemoteRF host. To date, the RemoteRF platform at UCLA has logged over 7,000 reservations by more than 200 students and researchers, totaling nearly 4,000 hours of use.

Educational Use

In January 2025, RemoteRF made its debut in an undergraduate course on communications and has been used each academic quarter since then in multiple courses on communications. In this debut, sixty undergraduate students used RemoteRF to complete three lab exercises where they implemented digital modulation, pulse shaping, matched filtering, symbol detection, channel estimation, and equalization. At the time, only five SDRs were connected to RemoteRF, but all sixty students were able to successfully share those devices to complete their labs on time. Two of the five SDRs were configured in a loopback fashion, with a cable connecting their transmit and receive ports; this provided students with a controlled channel for initial development and testing before moving on to over-the-air communication. The other three SDRs had actual antennas connected to their transmit and receive ports, with a band-pass filter inserted at the transmitter to ensure students could only operate within the 915 MHz industrial, scientific, and medical (ISM) frequency band.

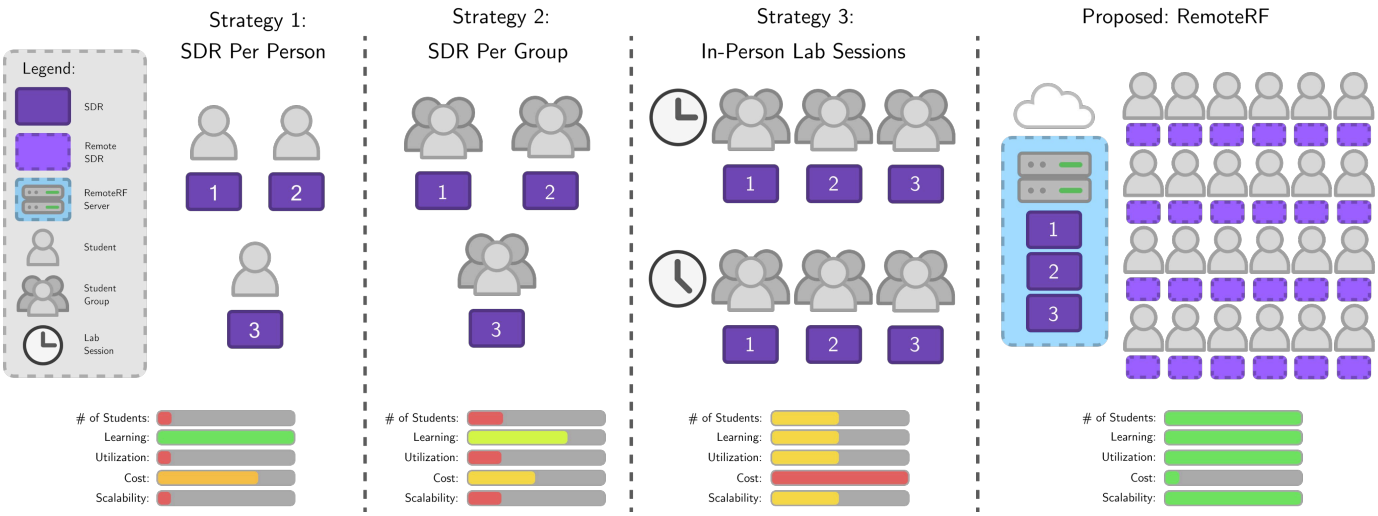

Fig. 4. Compared to purchasing an SDR per student, an SDR per group, or holding in-person lab sessions, the continuous availability and shared access offered by RemoteRF allows it to support larger classes with fewer SDRs, while also offering better learning outcomes, device utilization, and scalability.

In a subsequent graduate-level course on digital communications, students developed timing and frequency synchronization techniques in order to successfully transmit from one SDR to another over the air. Students were also tasked with implementing multi-tap channel equalization and orthogonal frequency-division multiplexing (OFDM) to combat frequency selectivity. For students to clearly observe frequency selectivity, multiple cables of different lengths were used to create a multi-path loopback channel that mimics multi-path propagation. This illustrates merely one unique way that RemoteRF can be used to devise controlled scenarios and test cases for students to experiment on. This particular graduate-level course is co-listed with UCLA's online master's degree program, and given RemoteRF can be accessed via the campus VPN, these remote learning students were able to complete all of the same lab exercises as on-campus students. In another graduate-level class on wireless communications, students used RemoteRF to complete quarter-long research projects where they implemented orthogonal time frequency space (OTFS) modulation, 5G initial access mechanisms, and ML-based channel coding techniques.

Research Use

As a research tool, RemoteRF is currently being used at UCLA to develop and experimentally evaluate new communication techniques. One such example is in developing ML-based radio frequency (RF) fingerprinting techniques, where subtle hardware signatures present in transmit signals are used to uniquely identify a device. In this context, RemoteRF is serving as a centralized platform to collect real-world data across several SDRs in order to train and test deep learning models. RemoteRF is being used in other work on communications-constrained robotics control and navigation. This is done by mounting an SDR on a mobile robot platform via the RemoteRF host software, allowing it to be remotely accessed during experiments. In addition to this, phased array SDRs connected to RemoteRF are being used to develop new beamforming methods for interference cancellation and integrated sensing and communication (ISAC). RemoteRF is also being used to develop and evaluate channel coding schemes that are robust to adversarial jamming. These diverse use cases illustrate how RemoteRF can be shaped and leveraged for a diverse range of research problems in wireless.

Latency Comparison

While latency has not been a noticeable issue in UCLA's RemoteRF deployment, a simple test was run to benchmark the platform against the conventional case where SDRs are connected directly to a user's personal computer, e.g., over USB. This was done by running three commands 1,000 times each on an ADALM-PLUTO SDR [15] when connected locally (over USB), to the RemoteRF server directly, and to a host device (Raspberry Pi 4). The end-to-end latency associated with executing each command was recorded, and the resulting distributions are shown in Fig. 5. Naturally, there is higher latency when running these commands on RemoteRF, especially with host devices, since an additional hop is introduced between the user and the SDR. Still, the difference in latency is imperceptible for many use cases and is fairly consistent across runs. While certainly an area of potential improvement, these results and our first-hand accounts confirm that this increase in latency is dramatically outweighed by the convenience, cost savings, and opportunities the RemoteRF platform affords to both students and researchers.

V. Discussion and Outlook

Student Interest Before Using RemoteRF

Before developing and deploying RemoteRF, a survey was given to 50 undergraduate and graduate electrical engineering students at UCLA. Some noteworthy results of that survey are as follows:

- 88% of students reported being interested or very interested in using SDRs.

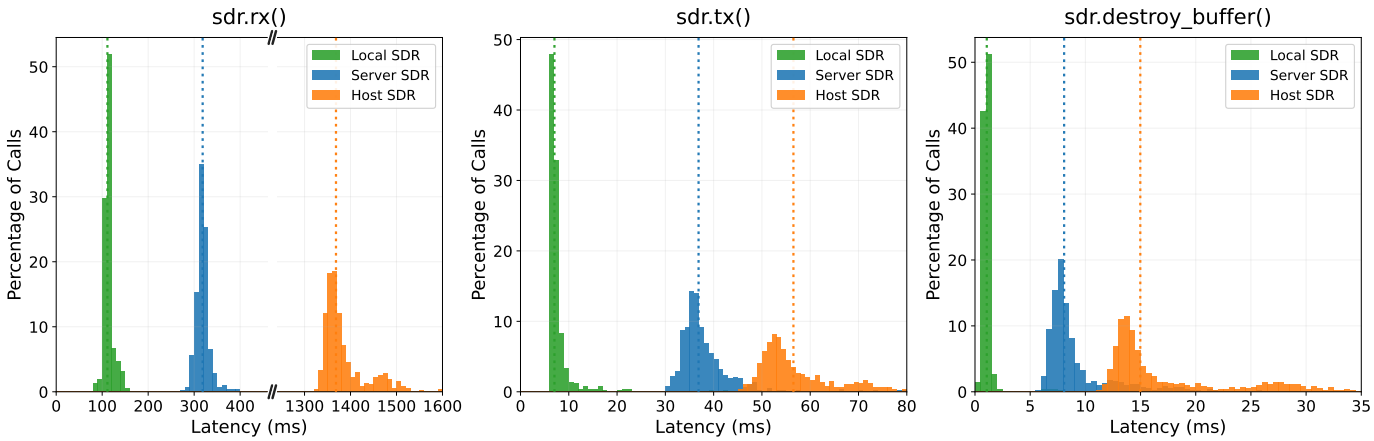

Fig. 5. End-to-end latency distributions for three ADALM-PLUTO SDR [15] function calls when the SDR is connected (i) locally over USB, (ii) directly to the RemoteRF server, and (iii) to a RemoteRF host. Functions were called 1,000 in each case, and the medians are shown as vertical dotted lines. `sdr.rx()` receives a signal of 100,000 samples. `sdr.tx()` transmits a signal of 10,000 samples. `sdr.destroy_buffer()` destroys the onboard transmit and receive buffers. RemoteRF naturally increases latency relative to a local USB connection, but the overhead is imperceptible or negligible in many applications.

- 86% of students agreed or strongly agreed that they wished courses incorporated more hands-on learning.
- 68% of students indicated they would use SDRs in their free time to explore wireless communications beyond course material if they were readily accessible.

These responses indicated a strong desire and need for increased hands-on learning in UCLA's communications engineering curriculum, prior to the deployment of RemoteRF.

Student Feedback After Using RemoteRF

Upon integrating RemoteRF into both undergraduate and graduate courses at UCLA, it was immediately clear that students both enjoy and benefit from using SDRs. After using RemoteRF in an undergraduate course in Winter 2026, an anonymous survey issued to another group of 50 students yielded the following responses:

- 86% of students said they had never used an SDR prior to using RemoteRF.
- 86% of students said RemoteRF helped reinforce prerequisite material on linear systems and signal processing.
- 92% of students said RemoteRF improved their understanding of concepts and techniques discussed in class.
- 83% of students said RemoteRF increased their interest in wireless communications and RF engineering.
- 72% of students said they often used the campus VPN to connect to RemoteRF from off campus.
- 61% of students said they were interested in continuing to use RemoteRF after the course concluded.

Anecdotally, many students also reported that implementation on the SDRs strengthened their understanding of course material, excited them about communications, and prepared them for job interviews. The resounding success of RemoteRF also highlighted that a relatively small pool of SDRs was sufficient to support even very large class sizes, thanks to its remote access and reservation framework.

RemoteRF's Usage Statistics

In addition to these survey results, RemoteRF's usage logs were parsed to provide quantitative insights on its use from March 2025 through June 2026. These findings are shown in Fig. 6 and summarized below.

- RemoteRF has been used for nearly 4,000 hours, with most of this by students to complete coursework where they implement core communication techniques on actual SDRs. This usage has accelerated as these lab exercises have become more integrated into course curricula.
- The majority of RemoteRF usage takes place outside normal business hours of 9 AM to 5 PM, both on weekdays and on weekends. On a typical weekday, for instance, 62% of usage happens outside of this range. This indicates that users often take advantage of the 24/7 access provided by RemoteRF.
- We also observe that users can reliably access SDRs when they need to, based on the *lead time* of their reservations. Nearly 60% of the time, users successfully reserve SDRs for the *ongoing* reservation window, i.e., they access SDRs instantly. Around 24% of the time, users reserve SDRs for a time slot that starts within 30 minutes. Although they rarely need to, students can also plan ahead by making reservations well in advance if they prefer. These results confirm that a handful of SDRs can support a far greater number of users without serious conflict, thanks to RemoteRF's continuous availability and its automated reservation manager. With RemoteRF, students are able to complete lab exercises at their own pace and indulge in their curiosity virtually anytime.

Research Lessons Learned

In using RemoteRF for research at UCLA, a few key lessons were learned. The need for extremely precise synchronization across SDRs is important for many experiments, but is not supported in the current version of RemoteRF, making it a high priority for future releases. In addition, interference across

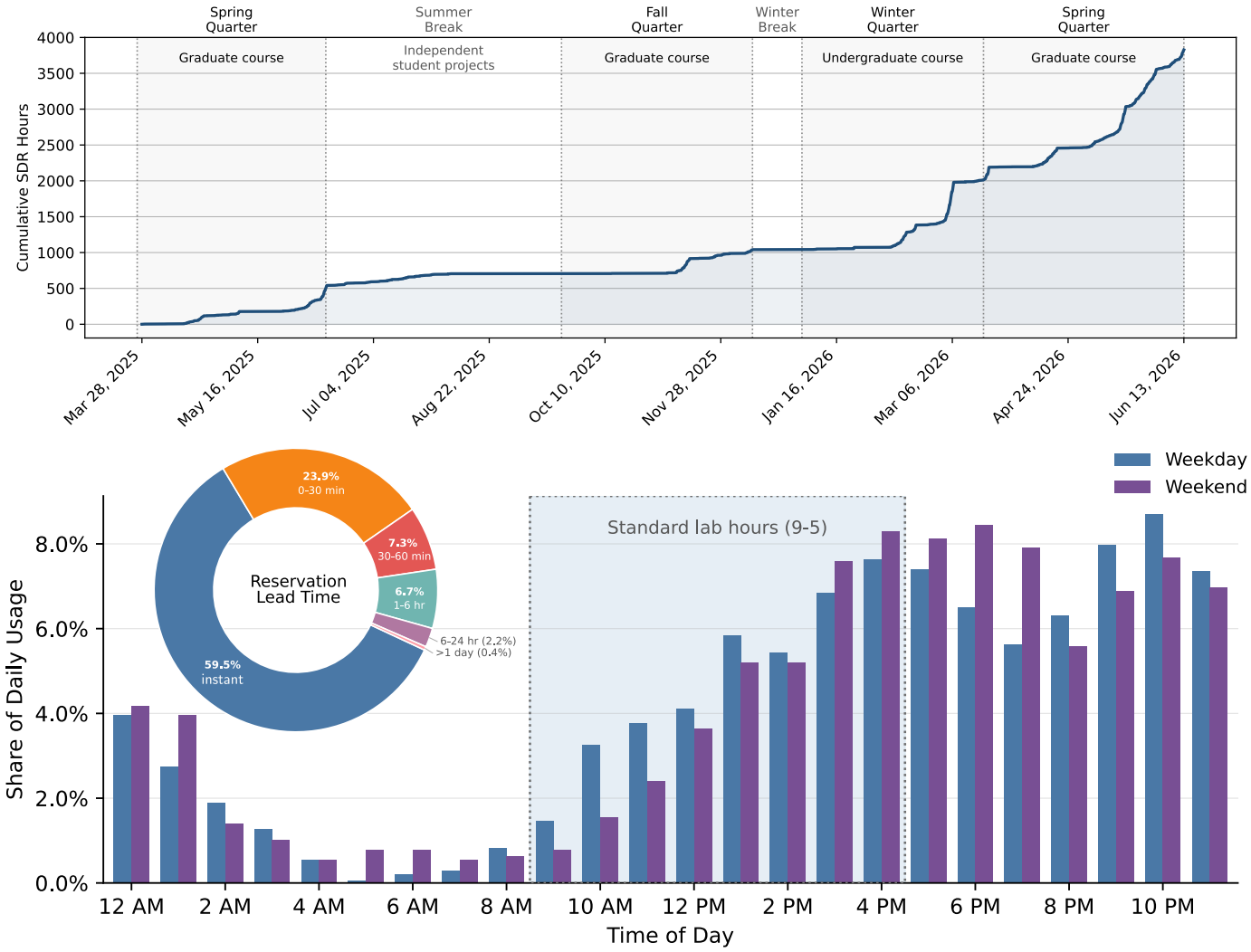

Fig. 6. (top) Cumulative usage on the UCLA RemoteRF deployment over time. (bottom) The share of daily usage on a given day during the work week and the weekend. (middle) Reservation lead time showing how far in advance users place reservations.

SDRs can be problematic when devices are co-located; this motivates the need for a mechanism that informs users of the transmit frequency and bandwidth of other users' SDRs—another feature to be added in future releases of RemoteRF. Another useful feature would be the ability to remotely power cycle devices, to aid users in debugging and to protect devices that may overheat if run continuously for too long. Finally, adding native support of more SDR models to RemoteRF remains a high priority to expand its utility to a broader range of research problems.

VI. Concluding Thoughts and Looking Ahead

This article has unveiled RemoteRF, an open-source platform for constructing large-scale SDR testbeds that can be remotely accessed over the internet. The value of RemoteRF in both research and education has been substantiated by nearly 4,000 hours of use by more than 200 students and researchers at UCLA to date. As an educational tool, RemoteRF has been used in five course offerings at UCLA, providing undergraduate and graduate students the opportunity to implement core communication concepts on SDRs through lab exercises that parallel course lectures. In research, RemoteRF has been used by UCLA researchers on RF fingerprinting, communication-constrained robotics, interference cancellation, and ISAC. RemoteRF is fully open-source and extensively documented at `remoterf.net`, allowing anyone across the globe to deploy their own instance of RemoteRF for research or education within a matter of minutes.

Acknowledgments

This work has been supported by the Educational Innovation Grants program and the Internet Research Initiative at UCLA.

References

[1] M. Dohler, R. W. Heath, Jr., A. Lozano, C. Papadias, and R. Valenzuela, "Is the PHY layer dead?" *IEEE Communications Magazine*, vol. 49, no. 4, pp. 159–165, Apr. 2011.

[2] E. Björnson, M. Dohler, J. Hoydis, and R. W. Heath, Jr., "Automating wireless research and development: What remains human?" *Preprints*, April 2026. [Online]. Available: https://doi.org/10.20944/preprints202604.1172.v1

[3] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168–179, Feb. 2018.

[4] A. M. Wyglinski, D. P. Orofino, M. N. Ettus, and T. W. Rondeau, "Revolutionizing software defined radio: case studies in hardware, software, and education," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 68–75, Jan. 2016.

[5] T. F. Collins, R. Getz, D. Pu, and A. M. Wyglinski, *Software-Defined Radio for Engineers*. Boston, MA: Artech House, 2018.

[6] National Science Foundation, "Platforms for advanced wireless research (PAWR)," https://www.advancedwireless.org/, accessed: June 1, 2026.

[7] J. Breen *et al.*, "POWDER: Platform for open wireless data-driven experimental research," in *Proc. International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*, 2020, pp. 17–24.

[8] D. Raychaudhuri *et al.*, "Challenge: COSMOS: A city-scale programmable testbed for experimentation with advanced wireless," in *Proc. International Conference on Mobile Computing and Networking*. New York, NY, USA: Association for Computing Machinery, 2020.

[9] V. Marojevic, I. Guvenc, R. Dutta, M. L. Sichitiu, and B. A. Floyd, "Advanced wireless for unmanned aerial systems: 5G standardization, research challenges, and AERPAW architecture," *IEEE Vehicular Technology Magazine*, vol. 15, no. 2, pp. 22–30, 2020.

[10] D. Raychaudhuri *et al.*, "Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols," in *Proc. IEEE Wireless Communications and Networking Conference*, vol. 3, 2005, pp. 1664–1669.

[11] L. Bertizzolo, L. Bonati, E. Demirors, and T. Melodia, "Arena: A 64-antenna SDR-based ceiling grid testbed for sub-6 GHz radio spectrum research," in *Proc. International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*, 2019, pp. 5–12.

[12] P. Orduña *et al.*, "The WebLab-Deusto remote laboratory management system architecture: achieving scalability, interoperability, and federation of remote experimentation," in *Cyber-Physical Laboratories in Engineering and Science Education*. Springer, 2018, pp. 17–42.

[13] M. Inonan, B. Chap, P. Orduña, R. Hussein, and P. Arabshahi, "RHLab scalable software defined radio (SDR) remote laboratory," in *Proc. International Conference on Remote Engineering and Virtual Instrumentation*. Springer, 2023, pp. 237–248.

[14] A. Alkhateeb, S. Jiang, and G. Charan, "Real-time digital twins: Vision and research directions for 6G and beyond," *IEEE Communications Magazine*, vol. 61, no. 11, pp. 128–134, Nov. 2023.

[15] Analog Devices, "ADALM-PLUTO software-defined radio active learning module," Online: https://www.analog.com/en/resources/evaluation-hardware-and-software/evaluation-boards-kits/adalm-pluto.html, 2024, accessed: 2026-04-06.