Board 382: RHLab RELIA: A Remote Integrated Environment forEmbedded Computing and RF Communication Systems

Brian Chap, University of Washington

Brian Chap is a Ph.D. student and research assistant of the Remote Hub Lab (RHLab) in the department of Electrical and Computer Engineering at the University of Washington. Brian's research spans remote engineering, computer vision, human-computer interaction, and image processing and sensing.

Marcos Jose Inonan Moran, University of Washington

Marcos Inonan is a PhD student and research assistant in the Remote Hub Lab (RHLab) of the department of Electrical and Computer Engineering at the University of Washington in Seattle. His research is centered on developing remote laboratories with a lens of equitable access to engineering education, and driven by his commitment to promote diversity, equity and inclusion in STEM education. In addition to his research on remote laboratories, Marcos has expertise in digital communication theory, signal processing, radar technology, and firmware engineering. Additionally, he has extensive experience in teaching embedded systems and senior design courses.

Zhiyun Zhang, University of Washington

Zhiyun Zhang is an undergraduate research assistant in the Remote Hub Lab at the University of Washington. He is a graduating senior and an incoming MS student with a focus area on embedded systems and remote engineering. Zhiyun is the recipient of the outstanding academic excellence award from the United States President's education awards program in 2019.

Pablo Orduna, LabsLand
Payman Arabshahi, University of Washington
Dr. Rania Hussein, University of Washington

Dr. Rania Hussein is an Associate Teaching Professor in the Electrical and Computer Engineering department at the University of Washington, where she also serves as the founder, principal investigator, and director of the Remote Hub Lab (RHLab). With her research focus on embedded systems, medical image analysis, digital twinning, and remote engineering, Dr. Hussein is committed to developing innovative solutions that enhance equity and access in engineering education and telehealth practices. Her work in promoting diversity, equity, and inclusion in higher education led to the successful building and passing of the religious accommodation law in the State of Washington, which provides alternative exam testing accommodations for students due to religious observances. Dr. Hussein is the recipient of the 2021 Innovative Program Award from the Electrical and Computer Engineering Department Head Association (ECEDHA), for founding the RHLab, as well as the 2022 IEEE Region 6 Outstanding Engineering Educator, Mentor, and Facilitator in the Area of STEM Award, recognizing her contributions to advancing students' success, mentorship, empowering under-represented communities, and promoting equitable access to engineering education.

RHLab RELIA: An Integrated Remote Environment for Embedded Computing and RF Communication Systems

Abstract

The development of technologies designed for the virtualization of signal and information processing experimentation has been persistently constrained by cost-associated hardware limitations, the inability to scale for larger audiences, and the lack of a flexible framework which supports user-specific interaction. Given recent advances in cloud computing, we introduce an adaptable, open-source system harnessing field-programmable gate array (FPGA) and software-defined radio (SDR) platforms to streamline the process of analyzing communication patterns between various individual transmitter-receiver pairs. A real-time user environment designed to complement the GNU Radio toolkit is demonstrated with minimal signal interference, enabling remotely-controlled, real information transfer. User-defined configuration files are processed through various Firejail-secured runner engines and dynamically visualized with adjustable feedback and interaction. By integrating novel images into the ADALM-PLUTO vanilla architecture, improved processed signal resolution was accomplished, with additional potential for ARM processor or FPGA reprogramming.

The findings of this paper are targeted towards individuals of all communities, including those with insufficient access to requisite hardware. To remedy such issues, the integrated environment has been adopted and is currently accessible through LabsLand, a partner in this research, and its network of affiliated universities and institutions. Individual contributions to the constructed system are highly distributable by courtesy of the modular nature of the provided framework, encouraging collaboration and sharing of physical resources. Existing functionalities of LabsLand, including learning management systems, are anticipated to further contribute towards the fostering of a complete, visual environment for users, replicating the actual experience of end users in standard, on-site hardware experimentation without associated localized issues.

Introduction

Numerous lessons learned amidst the COVID-19 crisis have pushed engineers, educators, and other professionals to rethink lab work approaches post-pandemic era. Offering an equivalent to hands-on engineering labs virtually presented itself as a particular challenge during the emergency transition to work-from-home (WFH) and remote learning. This necessitated innovative strategies to create lab-based solutions efficiently and conveniently for all individuals, irrespective of geographic location. One such strategy involved the implementation of remote hardware systems for forming full-fledged remote lab experiences without compromising on positive aspects of physical hardware experimentation. While the implemented systems may have appeared temporary in nature, and were often inadequate in scale, construction, and integration, the potential effectiveness of using such technologies to replicate, and improve, testing and learning experiences for individuals was noticeable. Such experiences have inspired this work which seeks to design and distribute a new generation of environments offering an open-access solution to costly hardware platforms unobtainable to many under-served communities and institutions with limited resources. This project builds on the success of previously implemented remotely-accessible FPGA systems by expanding scope and incorporating hardware which integrates FPGAs and software-defined radios (SDRs), together with new software enablers, for interdisciplinary projects in scientific and engineering disciplines. The proposed toolkit is replicable across institutions and provides access to industry-grade hardware for all communities. While individual institutions may use this open-source toolkit to create a remote lab for their own purposes, our sustainability plan, in coordination with LabsLand, proposes a scalable solution allowing users to connect individual contributions together in order to decrease equipment purchase costs and cultivate further inter-institutional collaboration by sharing both physical resources and user-generated content.

In this paper, we share our approach in implementing an open-source, integrated remote environment for software defined radio applications using ADALM-PLUTO. In this system, individuals using GNU Radio may design a complete flow from one device emitting a particular signal to another device receiving that signal with a different, separate GNU Radio configured process. By adopting this technology, individuals will be able to analyze real radio waves without encountering small-scale device issues.

Related Work

GNU Radio

GNU Radio is a GPL-licensed, Linux-supporting software framework recognized for its low-level customizability. Signal and information processing units are arranged block-wise in C++, with each block corresponding to various sources/sinks (e.g. signal and noise generation, UDP and USRP I/O, file read/write, oscilloscope visualization, etc.) or operations (e.g. modulation/demodulation, interpolation, FFT, filtering, delays, gain control, etc.). Transceiver outputs are graphed for ease-of-use, although the process of tuning such outputs is arduous, due to the irreplicability of precise hardware tx/rx chains [1]. The system defined and discussed in this paper aims to address this issue via an interactive user environment which enables end users to tune results in real-time and with sufficient specificity.

Software defined radio (SDR) architectures

Since the release of Universal Software Radio Peripheral (USRP) in 2003 [2], Software Defined Radio (SDR) has gained significant traction for its versatility in the construction of high-quality communication prototypes. The ability to manage signal processing through FPGAs and other programmable devices, and the subsequent ability ”to turn hardware problems into software problems” [3] is a feature which has ensured continual growth in the field, from both developers and end users alike.

ADALM-PLUTO, or PlutoSDR, (as visualized in Figure 1) is one instance of an SDR module developed by Analog Devices, Inc. to help individuals self-learn wireless communication, software-defined radio (SDR), and radio frequency (RF) [4]. This module was selected for the design due to its reliability [5], cost-effectiveness, USB connectivity, high signal-to-noise (SNR) ratio, and full-duplex architecture. Additionally, the ADALM-PLUTO supports a wide range of tools (e.g. Matlab and GNU Radio). Its 12-bit ADC (analog-to-digital converter) and DAC (digital-to-analog converter) provide sufficient signal resolution for most experimentation expected for the design. A Raspberry Pi may be incorporated to control ADALM-PLUTO over high-speed data transfer.

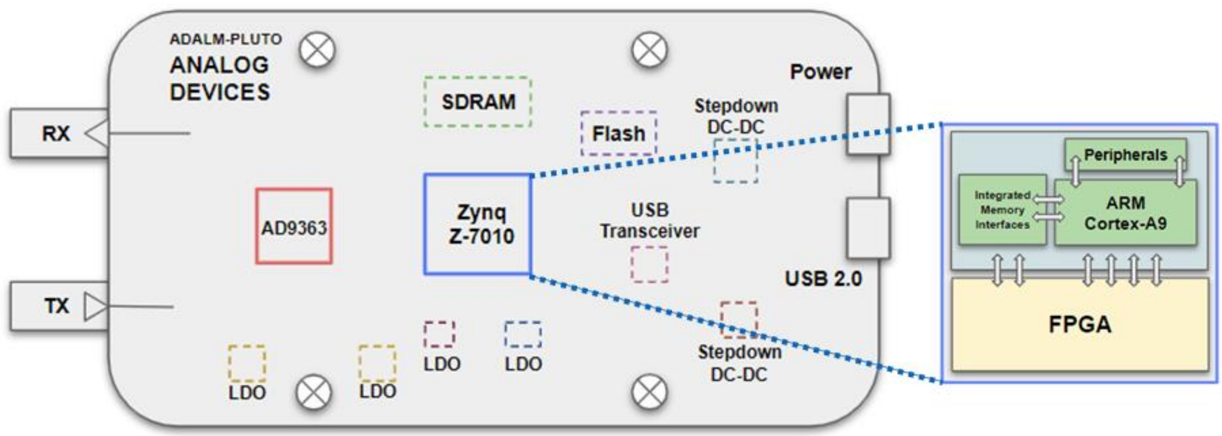


Figure 1: Simplified schematic of the internal structure of an ADALM-PLUTO module

Integrated Remote Environments

Previous publications have attempted to integrate SDR architectures into environments which enable remote hardware access. Xu et al. propose a communications-oriented environment requiring prior allocation of resources through a reservation system (reducing real-time applicability and scalability) and MATLAB. As noted by the authors, if an allocated resource were to behave unresponsively, a user must request a re-allocation of the previously assigned resources [6]. We instead propose a design which enables users to process tasks on any of a wide range of available devices and handles such failures through automatic time-outs, improving scalability and efficiency.

Mikroyannidis et al.'s FORGE design incorporates open-source GNU Radio, rather than MATLAB, as a framework [7]; however, they introduce virtual machines to run SDR, raising potential questions relating to scalability and security, and require a calendar-based system for the allocation of resources. Somashekar et al. similarly use GNU Radio, but on a more limited scope [8]. Somanaidu et al. also suggest an integrated SDR environment without a custom interface for analyzing frequency modulation (FM) signals using the USRP 2901 platform [9], a solution which does not consider scalability and is relatively less cost-effective than our suggested platform.

Methods

Hardware

To enable the delivery of higher resolution processed signals (rather than raw data, which is limited by transmission rate) to hosts, as well as permit reprogramming of the ARM processor and FPGA, customized Buildroot-based images were developed. BR2_EXTERNAL was selected for this purpose, as the manufacturer buildroot system is not actively maintained [10]. The ADALM-PLUTO firmware was also modified to further expand the existing frequency range, with new frequencies ranging from 70 MHz to 6 GHz. To address prior issues of scalability, as shown in Figure 2, all ADALM-PLUTOs are connected to a router via Ethernet, ensuring individual device access by IP address (in lieu of USB cable). This greatly reduces wiring between Raspberry Pis and ADALM-PLUTOs and expands the spatial scope at which Raspberry Pis and ADALM-PLUTOs may exchange information (providing the added benefit of improved long-term maintenance and reduced signal interference by selective placement).

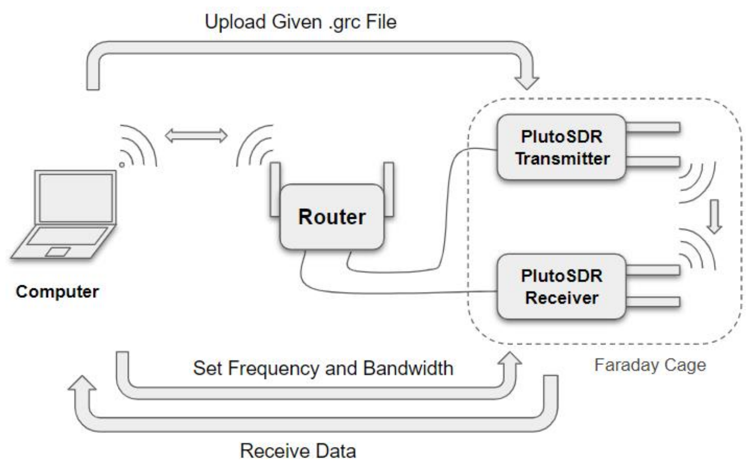


Figure 2: Dual ADALM-PLUTO setup for transmitting and receiving signals

Software

The software is composed of a central server handling the user interface (UI), scheduling, and data exchange, with authentication handled externally by weblablib [11]. A properly credentialed user may add a pair of receiver and transmitter GNU radio configuration (GRC) files to a Redis memory database (ideal for scaling and streaming large quantities of data), observe file progress (i.e. whether a file is queued, being processed, completed, etc.), or delete a pair of files when applicable. When a receiver (dubbed the “leading device”) is available, the scheduler identifies a receiver GRC file sent by the user based on priority and assigns it to the device. When a transmitter (dubbed the “lagging” device) is available, the scheduler assigns a configuration to it if and only if the leading device has either finished processing the associated receiver file or is actively processing the file. At assignment time, a GRC file is loaded onto a corresponding Raspberry Pi and QT components embedded within each file’s flowgraph are converted into individually-designed blocks before compilation (courtesy of GNU Radio Companion Compiler, or GRCC). Firejail sandboxing is utilized to prevent the execution of malicious contents, or contents which attempt to access restricted space. A thread responsible for checking file progress interrupts the process if a user requests file deletion or if execution is exceedingly time-exhaustive, ensuring optimal allocation of resources.

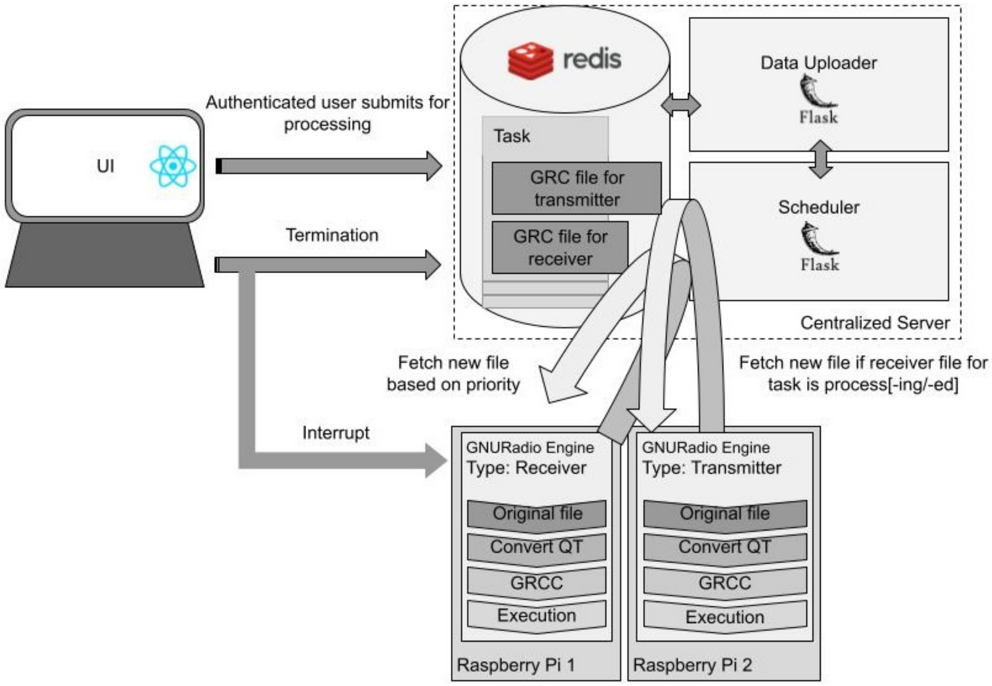


Figure 3: High-level software implementation schematic

Results

Hardware

A highly modular system comprised of multiple separate components (each consisting of a Raspberry Pi 4 Model B and ADALM-PLUTO) and a centralized server was developed. SDR libraries provided by the GNU Radio package enable the controlled timing of data acquisition and transmission for each ADALM-PLUTO through HTTP and web sockets. Specifications for the Raspberry Pi and ADALM-PLUTO are included in Table 1 of this paper.

Software

Interactivity, and the ability of an end user to adjust features dynamically, are crucial towards realizing a full environment for individuals utilizing the system. Users may control parameters in real-time (e.g. amplitude, frequency, power, sampling rate, offset) and customize the handling of incoming data (e.g. arrangement, zoom, pausing, averaging, noise) on a graphical user interface (GUI) associated with the GNU Radio signal processing framework. The GUI is flexible in incorporation and may be integrated into additional SDR hardware devices besides ADALM-PLUTO, as well as any browser.

The link to the associated GitLab repository for this paper is here: `https://gitlab.com/relia-project/`.

Table 1: Raspberry Pi 4 & ADALM-PLUTO specifications

Specification	**Description**
Connectivity	USB 2.0
Maximum data rate	480 Mbps
Bandwidth range	200KHz - 20 MHz
Number of channels	2 (1 Tx - 1 Rx)
Type of Antenna	JCG401 - omnidirectional
Processor resolution	64-bits
Operative System	Linux
Programming Language	Python
GRC version	3.10

Discussion

Signal Interference

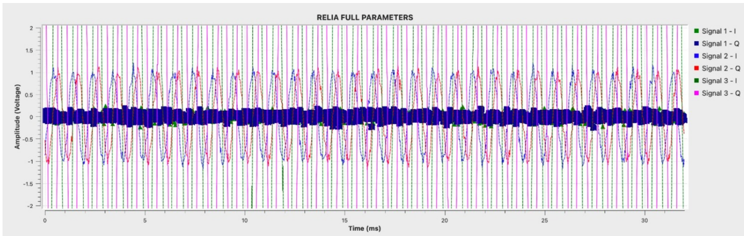
Signal interference among multiple ADALM-PLUTO modules was minimized through isolation in nickel and copper sheets, acting as a form of a Faraday cage. Packet Reception Ratio (PRR) for the transmission of 100K sequential ASCII characters under binary phase-shift keying (BPSK) modulation was measured and determined to be approximately 99% at distances of both 10 centimeters and 1 meter, implying minimal signal interference [12] [13].

Graphical User Interface

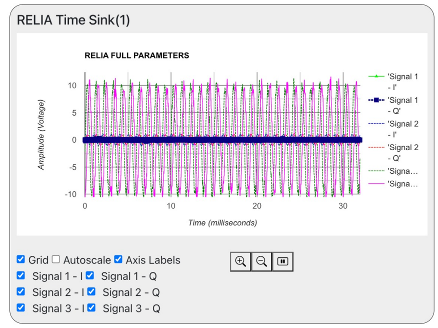
Our UI and GRC (GNU Radio Companion) are both visual interfaces for GNU Radio projects; both take user input from a YAML file containing all parameters and use Python libraries to read configuration files, as well as GNU Radio libraries to canalize and process the streaming data. Differences lie in the visualization framework; while our UI is based on Google Charts Gallery for creating independent interactive data in web browsers, GRC is a graphical interface for GNU Radio, which is a software-defined radio (SDR) toolkit.

Examples of tools available to users on the GUI are shown in Figure 4. A comparison of multichannel data from different domains: time, frequency, and scatter (constellation), is plotted. Both provide the user the same functionalities (pause, on/off, autoscale, grid on/off, etc.). However, the design of the system's independent windows allow users to monitor and control the streaming data seamlessly, as all controls from a plot are inside every window. It should be noted that, in contrast to GRC, multiple tool windows are available to a given user at a single time, allowing an individual to analyze multiple features of submitted data simultaneously.

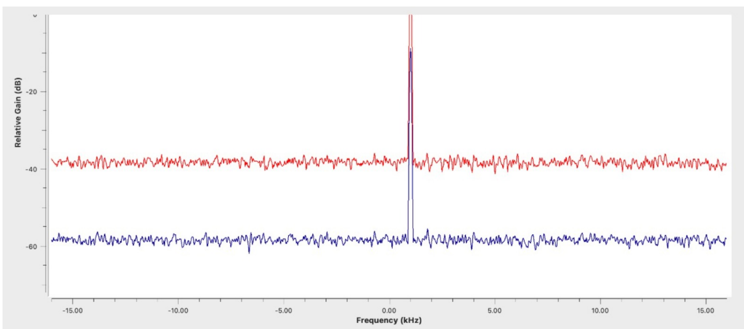
As previously noted in the Related Work section of this paper, if a resource were to behave unresponsively, the system scheduler would purge the task on the resource, providing an error message for user guidance (alongside additional messages if a GRC file were to fail at compile-time, runtime, etc.) and enabling re-submission to an alternate resource.



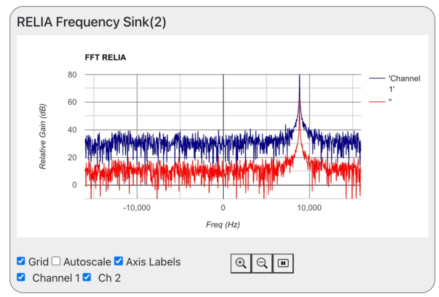
(a) Three-channel time sink I/Q data - GRC



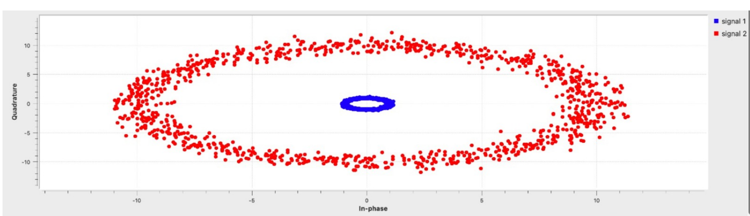
(b) Three-channel time sink I/Q data - Our system



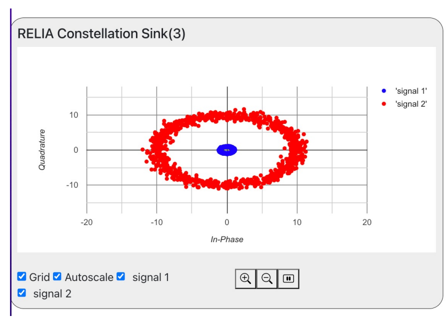
(c) Two-channel frequency sink I/Q data - GRC



(d) Two-channel frequency sink I/Q data - Our system



(e) Two-channel constellation sink I/Q Data - GRC



(f) Two-channel constellation sink I/Q Data - Our system

Figure 4: GRC (left) vs. Our system (right) GUI comparison

Digital Design Extensions

The system may be applied towards digital design (in addition to communication and signal processing) by harnessing remote access of the FPGA or ARM cores which control a SDR device (e.g. ADALM-PLUTO). Initial configuration tests on ADALM-PLUTO by BR2_EXTERNAL suggest that reprogramming of the Zynq platform for this end purpose is feasible.

Conclusion

This paper discusses the construction and implementation of a fully-integrated environment for the remote analysis of signal patterns. Through the incorporation of flexible and scalable features, our real-time system is demonstrated to be practical for larger audiences needing to replicate in-person hardware experimentation virtually. The environment achieves improved signal resolution and is resistant towards external interference, ensuring necessary robustness for real-life use.

The system outlined in this paper is presently available via LabsLand, a partner in this research. The remote laboratories at our research group have been used, through the LabsLand[1] network[14], by 3,700 students from 16 countries over more than 100,000 past laboratory sessions. Learning management systems (e.g. Canvas, Moodle, Sakai, etc.) are integrated and expected to further encourage the use of our system by individuals globally.

Acknowledgements

This work was supported by the National Science Foundation's Division Of Undergraduate Education under Grant 2141798.

References

[1] Danilo Valerio. Open source software-defined radio: A survey on gnuradio and its applications. In *ftw. Technical Report*, 2008.

[2] Matt Ettus and Brian Bloom. Universal software radio peripheral (usrp). In *Proceedings of the 4th Workshop on Software Radio*, pages 1–8. IEEE, 2003.

[3] Eric Blossom. Gnu radio: Tools for exploring the radio frequency spectrum. In *Linux Journal*, 2004.

[4] Analog Devices Inc. Adalm-pluto: A wideband transceiver for software defined radio, 2018. URL `https://wiki.analog.com/university/tools/pluto`.

[5] Yonghan Kwon, Mingyu Park, and Jeongyeup Paek. A measurement study of adalm-pluto software defined radio with ieee 802.15. 4. In *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*, pages 865–867. IEEE, 2022.

[6] Zhengguang Xu, Wan Chen, Daiming Qu, Xiaojun Hei, and Wei Li. Developing a massive open online lab course for learning principles of communications. In *TALE*, pages 586–590. IEEE, 2020.

[7] Alexander Mikroyannidis, Diarmuid Collins, Christos Tranoris, Spyros Denazis, Daan Pareit, Jono Vanhie-Van Gerwen, Ingrid Moerman, Guillaume Jourjon, Olivier Fourmaux, John Domingue, and Johann M. Marquez-Barja. Forge: An elearning framework for remote laboratory experimentation on fire testbed infrastructure. In *hal-01656701f*, pages 521–559, 2017.

[8] Manjunath Somashekar, Preethi Biradar, Kalyan Ram Bhimavaram, Panchaksharayya S. Hiremath, and S. Arun Kumar. Remote labs for communications. In *Online Engineering and Society 4.0: Proceedings of the 18th International Conference on Remote Engineering and Virtual Instrumentation*, pages 47–54. Springer, 2021.

[1] `https://labsland.com`

[9] Utlapalli Somanaidu, Nagarjuna Telagam, Nehru Kandasamy, and Menakadevi Nanjundan. Usrp 2901 based fm transceiver with large file capabilities in virtual and remote laboratory. In *International Journal of Online Engineering*, pages 193–200. iJOE, 2018.

[10] Gwenhael Goavec-Merou, Pierre-Yves Bourgeois, and Jean-Michel Friedt. Embedded gnu radio running on zynq/plutosdr. In *Proceedings of the GNU Radio Conference*, 2021.

[11] Pablo Orduña, Jaime Irurzun, Luis Rodriguez-Gil, Javier Garcia-Zubia, Fabricio Gazzola, and Diego López-de Ipiña. Adding new features to new and existing remote experiments through their integration in weblab-deusto. In *International Journal of Online Engineering*, pages 33–39. iJOE, 2011.

[12] Marcos Inonan, Brian Chap, Pablo Ordña, Rania Hussein, and Payman Arabshahi. Rhlab scalable software defined radio (sdr) remote laboratory. In *20th Annual International Conference on Remote Engineering and Virtual Instrumentation (REV)*, 2023.

[13] Rania Hussein, Brian Chap, Marcos Inonan, Matt Guo, Francisco Luquin Monroy, Riley Maloney, Stefhany Alves, and Sai Jayanth Kalisi. Remote hub lab – rhl: Broadly accessible technologies for education and telehealth. In *20th Annual International Conference on Remote Engineering and Virtual Instrumentation (REV)*, 2023.

[14] Pablo Orduña, Luis Rodriguez-Gil, Javier Garcia-Zubia, Ignacio Angulo, Unai Hernandez-Jayo, and Esteban Azcuenaga. Increasing the value of remote laboratory federations through an open sharing platform: Labsland. In *Online Engineering & Internet of Things*, pages 859–873. Springer, 2018.