IEEE Robotics and Automation Practice (RA-P) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-P paper.

Open-Source Web Lab for Remote and On-Site Robotics Practice in a Realistic Zero-Setup ROS 2 Development Environment

Dāvis Krūmiņš, Veiko Vunder, Heiki Kasemägi, Alvo Aabloo, and Karl Kruusamäe

***Abstract*—Robot operating system 2 (ROS 2) has become the standard framework in modern robotics, but its steep learning curve and complex development environment present significant barriers to newcomers. This article presents an open-source web lab through which learners can start practicing ROS 2 programming on real robots right away with no development environment setup. Built upon our prior work with ROS 1, the web lab offers a realistic ROS 2 development experience by serving browser-based Linux desktop workstations connected to physical remote robots. We also demonstrate how the web lab can be leveraged to create a portable infrastructure for hosting ad hoc on-site robotics workshops that require zero setup from the participants.**

***Index Terms*—Docker, remote robot operation, robot operating system 2 (ROS 2), virtual labs, VNC.**

I. Introduction

ROBOT operating system 2 (ROS 2) has emerged as the standard software development framework for the contemporary robotics community, owing to its modular architecture and extensive ecosystem [1]. With the proliferation of robotics and automation across diverse industries, there is a growing need for specialists who can work with ROS 2. Notably, people perceive it as having an even steeper learning curve and greater development complexity than its predecessor ROS 1 [2]. Furthermore, newcomers to robotics and ROS can encounter a difficult barrier-to-entry in the form of a development environment setup [3].

While ROS 2 introduced multiplatform support—a feature not available in ROS 1—there are still many requisite installation steps a novice must face, which can deter them before they even get to writing their first ROS program. Delivering ROS 2 courses through online platforms where the development environment is preconfigured presents a scalable solution to this challenge. Feature-rich platforms, such as The Construct [4], enable remote learners to work on both simulated and real ROS robots in an almost realistic development environment with no local software setup required. This greatly simplifies the onboarding process without major tradeoffs; however, currently, there is a lack of open-source alternatives for such platforms.

To address the shortage of freely accessible tools for delivering comprehensive ROS 2 training remotely, we propose an open-source web lab that enables online learners to operate and program either physical or simulated ROS robots from a web browser with no specific operating system (OS) or local software installation prerequisites.

The contributions of this article are summarized as follows.

1) We present the technical solution for a web lab granting online learners access to an authentic ROS 2 Linux desktop development environment that offers an interactive programming experience of a remote robot.
2) We demonstrate the system's functionality with two different physical robots: 1) Robotont [5], [6], a ROS-compatible mobile robot platform, and 2) Universal Robots UR5 manipulator (see Video S1 in the Supplementary Material).
3) We show how the system can be adopted for use in on-premise robotics workshops, where participants can instantly gain access to a robotics development environment on their own devices, eliminating the need for organizers to provide them (see Video S2 in Table I).
4) We release the software under an open-source license and provide setup instructions along with installation scripts[1] that allow anyone to self-host the ready-made system and enable online access to their ROS-compatible robots.

II. Related Work

There is a considerable collection of solutions that have been put forth for ROS-based remote robotics labs, many of them, however, are either constrained to learning through simulation or feature an abstracted programming environment [7], [8], [9], [10], [11]. Here, we briefly examine a couple of notable attempts at delivering realistic ROS development experience over the Internet.

One approach taken by some remote labs is to connect online learners to remote ROS-based robots through a virtual private network (VPN) tunnel [12], [13], [14]. In [12], Turtlebot3

Received 10 May 2025; accepted 29 October 2025. Date of publication 5 November 2025; date of current version 2 January 2026. This work was supported in part by the European Union through the project "Increasing the knowledge intensity of Ida-Viru entrepreneurship." This article was recommended for publication by Associate Editor L. Penna Poubel upon evaluation of the reviewers' comments. *(Corresponding author: Dāvis Krūmiņš.)*

The authors are with the Institute of Technology, University of Tartu, 50090 Tartu, Estonia (e-mail: davis.kruumins@gmail.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/RAP.2025.3629341, provided by the authors.

Digital Object Identifier 10.1109/RAP.2025.3629341

[1] [Online]. Available: https://github.com/unitartu-remrob/remrob-setup

IEEE Robotics and Automation Practice (RA-P) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-P paper.

and Kinova Gen3 lite robot platforms are made available for hands-on robotics practice, while Bunse and Wieck [13] offered Turtlebot3 and PhantomX Mark III Hexapod. Both of these labs require learners to set up their own local development environment and thus are more suitable for users who already have some ROS experience.

A remote robotics lab that gives students the chance to work with industrial ROS robots is presented in [14]. The lab supports both ROS 1 and ROS 2 and is employed in a project-based learning competition, where students are able to work on a multirobot system consisting of a mobile rover and a UR5 robotic arm in a real-world testbed. To combat capacity issues, simulation environments are used exclusively for the first stages of the competition with only the best performing teams being granted access to physical hardware in the latter stages. Two different versions of the lab are described; the first version exposes ROS nodes over the Internet, similar to the approach in [12] and [13], while the more recently developed one utilizes a remote desktop approach. However, even the latest version does not fully eliminate the user-side setup, as it still requires users to establish a VPN connection. It also requires learners to navigate between separate systems for slot booking and robot observation—the latter being handled through a video conferencing application.

Robot Learning Lab (RLL) is a web lab for KUKA robotic arms to which learners or researchers can submit code and see its execution in the browser [15]. The provided web editor of RLL allows writing ROS programs with Python or Blockly Visual Programming Editor; to use their own preferred development tools, the learner must either install RLL software locally or use a prepared virtual machine. To date, RLL is exclusively dedicated to KUKA robotic manipulators, and the latest supported ROS distribution is Melodic [16].

OpenUAV is an open-source, ROS-centric web lab [17], which, similarly to RLL, runs the user code within containers, yet does not require any setup from the learner, as it provides a full ROS development environment via graphical desktop sharing over the web. OpenUAV has made the move to ROS 2 by adding support for ROS 2 Humble; however, it is still a cloud-exclusive testbed with no physical robots.

In our previous work, we introduced a web lab that offers authentic ROS 1 development experience with simulated and physical robots [18]. The lab was validated in a massive open online course on robotics in Estonia, but it lacked the ability to integrate with ROS 2 robots .

III. Requirements

We designed our web lab for remote ROS learning and development to satisfy the following requirements.

1) *Low barrier to entry:* Learner does not have to install any specific OS or specialized software, all that is required is a web browser.
2) *Realistic ROS development experience:* User interface is a standard Linux desktop environment with ROS software tools.

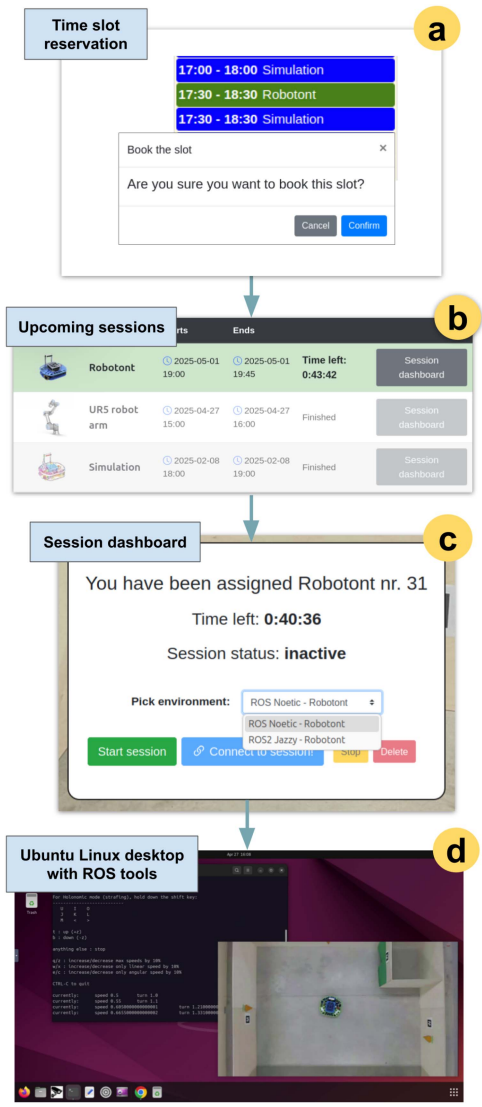

Fig. 1. Routine to start working with a physical remote ROS 2 robot: (a) reserving robot access, (b) activating the session, (c) choosing and booting the ROS 2 environment, and (d) connecting to in-browser Ubuntu 24.04 (Noble Numbat) desktop environment with live robot feed seen through a camera widget.

3) *Support for different ROS versions:* Both ROS 2 and ROS 1 environments are available. We retain compatibility with ROS 1 in case it is desired to use the system with robots that are not yet compatible with ROS 2.
4) *Varied learning modes:* Learner can choose between simulated or real robots.
5) *Extendibility:* Any ROS-supported robot can be easily added to the system.

IV. System Overview

A. Front-End Interface

The access to the remote lab is organized through time slot reservation. The lab administrators create time slots, which the learners can book on a first-come, first-served basis. Fig. 1 shows the steps that the learner routinely takes to start a ROS 2 development session with one of the remote lab's robots.

During an active session, the user is able to choose between different available ROS versions, as in Fig. 1(c). Upon booting the environment, it becomes possible to connect to a Linux desktop environment in the browser, which has the necessary tools to run ROS 2 programs on a remote robot seen through an overhead camera feed, as in Fig. 1(d).

A video demonstrating the described user flow can be found in the Supplementary Material (Video S1).

B. ROS Development Containers

A convenient way to manage the ROS sandbox environments of learners is to use container technologies. The main advantages are the following.

1) Containers allow hosting multiple simultaneous users on a single server.
2) Neither the learner nor the teacher has to fear accidental breaking of the system since the container sandbox can always be reset within seconds.
3) The sandbox environments can be highly configurable, and different ROS versions for different robots can be selected.
4) Containers can be made dynamic in respect to the connected user, e.g., loading a ROS workspace from their previous session.
5) The modularity of containers makes the web lab well positioned to support any future ROS distributions.

The container tool we opt for is Docker container platform. Docker images, a form of a system blueprint, are useful for managing different versions of ROS and robot-specific software packages. Furthermore, Docker allows enforcement of CPU and memory limits on the containers, useful for preventing resource abuse. Lastly, a valuable benefit of using Docker for containerization is the availability of *macvlan* network driver, which grants containers physical network IP addresses necessary for unhindered ROS peer discovery and message transmission.

Other container runtimes considered for sandboxing user sessions were Podman [19] and Sysbox [20], which offer better support for rootless containers that reduce the risk of a container breakout. However, with Podman, we discovered that rootless containers cannot be placed in the *macvlan* network. With Sysbox, the problem we encountered was its lack of GPU passthrough for containers. In contrast, for Docker, NVIDIA developers provide instructions for building CUDAGL images [21], which we were able to use for building hardware-accelerated ROS Docker containers. CUDAGL containers greatly improve the graphics performance of OpenGL-dependent software, such as Gazebo robotics simulator and sensor data visualization tool RViz, tools commonly utilized by ROS developers.

The main components of a single ROS Docker container[2] in our proposed system are as follows:

1) Ubuntu Linux (24.04 for ROS 2, 20.04 for ROS 1);
2) GNOME desktop environment;
3) ROS 2 Jazzy or ROS Noetic full desktop installation;
4) TigerVNC server [22];
5) VirtualGL [23].

TigerVNC transmits graphics of the headless container desktop over the network, while VirtualGL allows performing 3-D hardware rendering over VNC for OpenGL-based applications.

C. Architecture

To open the ROS development containers for remote connections, we propose the system architecture depicted in Fig. 2 (material adapted from [18]).

The end-user connects to a container through noVNC, as shown in Fig. 2(b), an open source web-based VNC client [24]. To organize access to the remote lab, a booking service was developed,[3] as shown in Fig. 2(g); if a learner has been authenticated and has an active session, then they are able to get assigned a physical robot container, as shown in Fig. 2(d), 2(e), and 2(i), or a simulation container, as shown in Fig. 2(j), based on the reserved session's type.

Container creation is orchestrated through our custom-built container control API,[4] as shown in Fig. 2(h), an intermediate layer to the Docker Engine API. The container control API is able to construct container configurations dynamically based on the incoming user's request and the available robot inventory.

Some of the dynamic container parameters the container control API makes use of include the following:

1) chosen ROS distribution (Jazzy for ROS 2, Noetic for ROS 1);
2) learner's ROS workspace;
3) assigned robot's ID;
4) webcam ID for the cell the robot is placed in.

Nginx web server configured as a reverse proxy binds the described components into a single system entrypoint, as shown in Fig. 2(a).

V. Use Case for On-Site ROS Workshops

In this section, we describe how the web lab, originally created with remote learning in mind, can also be adapted for on-premise robotics workshops where the heterogeneity of learners' devices presents a challenge in providing the appropriate ROS software development tools.

To ensure that everyone can participate, ROS workshop organizers either need to handle logistics of bringing devices with ROS preinstalled or distribute bootable Linux USB drives, which are increasingly hindered by modern system firmware (e.g., UEFI) security features and may not be compatible with certain devices, such as tablets or ChromeOS computers. To address this, we have developed a portable version of the system that eliminates the need for extensive preparation by letting participants work on ROS robots through the system-agnostic web application. In combination with portable robotic kits [25], it can facilitate ad hoc on-site robotics workshops or exhibits.

In the locally run version of the application, there is no robot access booking module; instead, the learner is directly

[2][Online]. Available: https://github.com/unitartu-remrob/remrob-docker

[3][Online]. Available: https://github.com/unitartu-remrob/remrob-webapp

[4][Online]. Available: https://github.com/unitartu-remrob/remrob-server

IEEE Robotics and Automation Practice (RA-P) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-P paper.

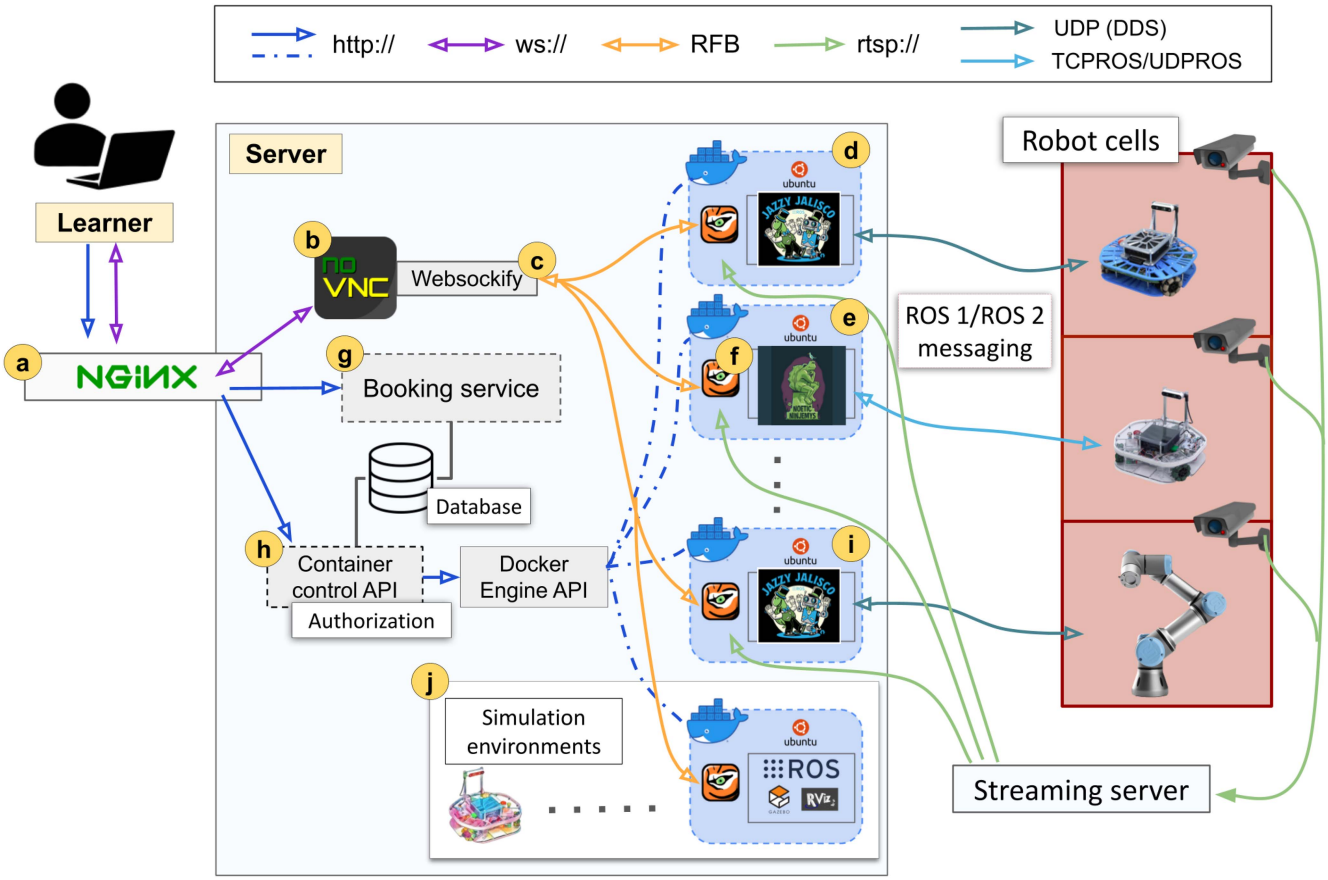

Fig. 2. System overview: (a) nginx reverse proxy server, (b) noVNC web application,(c) websockify bridge for translating TCP traffic of TigerVNC into WebSockets, (d) Ubuntu 24.04 Container with ROS 2 Jazzy networked to a ROS 2 mobile robot, (e) Ubuntu 20.04 Container with ROS Noetic networked to a ROS 1 mobile robot, (f) TigerVNC server, (g) booking service for reserving access to the remote lab, (h) container management API, (i) Ubuntu 24.04 container with ROS 2 Jazzy networked to a ROS 2 manipulator, and (j) simulation containers for robot digital twins.

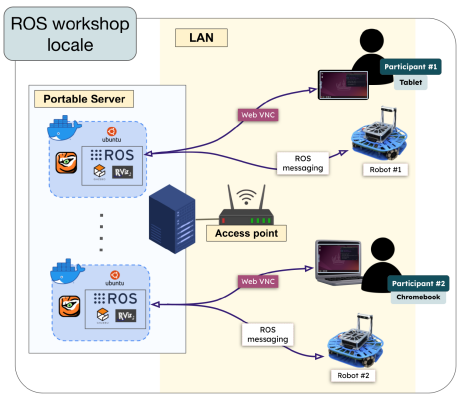

Fig. 3. Adopting the remote lab system for quickly deploying on-premise ROS workshops. Devices normally incompatible with ROS, e.g., tablets or Chromebooks, can access and program ROS robots via the system.

given the robot's credentials, which they can use to access its development environment. Upon entering the credentials, the learner is directed straight to the session dashboard, as shown in Fig. 1(c), where they are able to start a ROS 2 container.

Fig. 3 shows an example schematic of the web lab being deployed in a local area network (LAN) setting to support ROS workshop organization. A video demonstration of this use case with the Robotont robot [5], [6] is available in the Supplementary Material (Video S2).

VI. Replication

The software of the remote is released under MIT license. We have made available setup instructions along with an option for automated system installation via the Ansible automation engine. It is available online on GitHub.[5]

An option for lab building hardware-accelerated versions of the ROS containers is available, and we also provide instructions on how to configure the web lab's network for connecting any ROS-based robot to the system. See Supplementary Material S3 for more details.

VII. Future Work

The web lab has so far only been evaluated on a single server deployment [18]. Investigating a multiserver architecture could enable scaling the system to increase its capacity and performance.

[5][Online]. Available: https://github.com/unitartu-remrob/remrob-setup

IEEE Robotics and Automation Practice (RA-P) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-P paper.

TABLE I
DESCRIPTION OF SUPPLEMENTARY MATERIALS

	File name	Description
S1	remote-ros2-robot-web-lab-demo.mp4	Can also be viewed online.[6] This supplementary video gives an overview of the capabilities of the proposed remote web lab. It consists of three parts: first, the user flow as seen in Fig. 1 is shown, where by logging into the system through the website. [7] the user is able to teleoperate a remote Robotont robot by running a ROS 2 node. Secon, it proceeds to show an example of how through the same web lab a user is able to control a remote UR5 manipulator by setting a target pose goal with the MoveIt motion planner, thus demonstrating that the system is not limited to any specific single robot. Lastly, it shows the system's functionality as a potential simulation playground.
S2	use-case-for-onsite-ros-workshops-demo.mp4	Can also be viewed online.[8] This supplementary video demonstrates the system's use case for on-site ROS workshops. A portable server box is brought to the premises and connected to a power outlet. Robotont mobile robot is present and automatically connects to the server network as soon as it becomes active. A student arrives with a tablet device and connects to the portable server network by scanning a QR code on the robot. They proceed to type the server's IP address seen on the robot in the web browser's address bar, which leads them to the LAN version of the web lab's application. Upon entering the robot's credentials, they boot up a ROS 2 workstation connected with the robot. In the last part of the video, the student is seen teleoperating the Robotont mobile robot from their tablet device after starting a ROS 2 node from within the web application.
S3	ros2_robot_web_lab_source_code.zip	It is available online.[9] The supplementary code archive consists of five software modules: • remrob-setup • remrob-docker • remrob-server • remrob-webapp • remrob-spot-webapp The remote robot lab (S1) is based on *remrob-docker*, *remrob-server*, and *remrob-webapp*. Although each component can be set up individually (see the setup section in each module's respective README.md file), to streamline the process we recommend using the provided Ansible-based installation script found in *remrob-setup*. This module also includes instructions for adding robots to the server network and configuring hardware-accelerated ROS containers. The portable web lab (S2) software is based on *remrob-docker*, *remrob-server*, and *remrob-spot-webapp*. For details on how to build and start the application in LAN mode see the README.md file in *remrob-spot-webapp*. The code repositories are also available on GitHub: • https://github.com/unitartu-remrob/remrob-setup • https://github.com/unitartu-remrob/remrob-docker • https://github.com/unitartu-remrob/remrob-server • https://github.com/unitartu-remrob/remrob-webapp

VIII. CONCLUSION

This article serves as a technical demonstration of an open-source web lab for remote programming of ROS 2 robots. We consider it a well-suited platform for introducing ROS 2 to novice robotics students and enthusiasts as it requires no setup on the user's part, but retains fidelity of a professional development environment. In addition, we demonstrate a use case of the system for bootstrapping on-site ROS 2 workshops. The proposed web lab can host any ROS-based robot and paves the way for future large-scale ROS 2 educational experiments.

[6][Online]. Available: https://youtu.be/Tztqyd9i0Js
[7][Online]. Available: https://remrob.ut.ee
[8][Online]. Available: https://youtu.be/5NNo2MZAWfE
[9][Online]. Available: https://doi.org/10.5281/zenodo.15379490

REFERENCES

[1] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Sci. Robot.*, vol. 7, no. 66, 2022, Art. no. eabm6074.
[2] D. Portugal, R. P. Rocha, and J. P. Castilho, "Inquiring the robot operating system community on the state of adoption of the ROS 2 robotics middleware," *Int. J. Intell. Robot. Appl.*, vol. 9, pp. 454–479, 2025.
[3] D. Coleman, I. A. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: A moveit! case study," *J. Softw. Eng. Robot.*, vol. 5, no. 1, pp. 3–16, 2014.
[4] "The Construct Robotics Institute." (n.d.). [Online]. Available: https://www.theconstruct.ai/
[5] E. Mõtshärg et al., "Robotont 3–an accessible 3D-printable ROS-supported open-source mobile robot for education and research," *Front. Robot. AI*, vol. 11, 2024, Art. no. 1406645.

[6] R. Raudmäe et al., "ROBOTONT–open-source and ROS-supported omni-directional mobile robot for education and research," *HardwareX*, vol. 14, Jun. 2023, Art. no. e00436.

[7] W. A. M. Fernando, C. Jayawardena, and U. U. S. Rajapaksha, "Developing a user-friendly interface from robotic applications development," in *Proc. 2022 Int. Res. Conf. Smart Comput. Syst. Eng.*, 2022, pp. 196–204.

[8] D. Roldán-Álvarez, J. M. Cañas, D. Valladares, P. Arias-Perez, and S. Mahna, "Unibotics: Open ROS-based online framework for practical learning of robotics in higher education," *Multimedia Tools Appl.*, vol. 83, pp. 52841–52866, 2024.

[9] G. A. Casañ, E. Cervera, A. A. Moughlbay, J. Alemany, and P. Martinet, "ROS-based online robot programming for remote education and training," in *Proc. 2015 IEEE Int. Conf. Robot. Autom.*, 2015, pp. 6101–6106.

[10] B. Stefanuto, L. Piardi, A. O. Junior, M. Vallim, and P. Leitão, "Remote Lab of Robotic Manipulators through an Open Access ROS-based Platform," in *Proc. 2023 IEEE 21st Int. Conf. Ind. Informat.*, 2023, pp. 1–6.

[11] A. Kerem Erdoğmuş and U. Yayan, "Virtual robotic laboratory compatible mobile robots for education and research," in *Proc. 2021 Int. Conf. Innovations Intell. Syst. Appl.*, 2021, pp. 1–6.

[12] M. E. Cabrera and J. Raiti, "Physical robots for teaching mobility & manipulation using ROS in remote learning," in *Proc. 2024 ASEE Annu. Conf. & Expo.*, Portland, OR, USA, 2024, Art. no. 47844.

[13] C. Bunse and T. Wieck, "Experiences in developing and using a remote lab in teaching robotics," in *Proc. 2022 IEEE German Educ. Conf.*, 2022, pp. 1–6.

[14] A. Kumar, J. Jose, A. Jain, S. Kulkarni, and K. Arya, "Scalable and low-cost remote lab platforms: Teaching industrial robotics using open-source tools and understanding its social implications," in *Social Robotics*. O. Palinko et al., Eds., Singapore: Springer, 2025, pp. 202–215.

[15] W. Wiedmeyer, M. Mende, D. Hartmann, R. Bischoff, C. Ledermann, and T. Kroger, "Robotics education and research at scale: A remotely accessible robotics development platform," in *Proc. 2019 Int. Conf. Robot. Autom.*, May 2019, pp. 3679–3685.

[16] "The SDK for the KUKA Robot Learning Lab at KIT." (n.d.). [Online]. Available: https://github.com/KITrobotics/rll_sdk

[17] H. Anand et al., "OpenUAV cloud Testbed: A collaborative design studio for field robotics," in *Proc. IEEE 17th Int. Conf. Autom. Sci. Eng.*, 2021, pp. 724–731.

[18] D. Krūmiņš et al., "Open rmote web lab for learning robotics and ROS with physical and simulated robots in an authentic developer environment," *IEEE Trans. Learn. Technol.*, vol. 17, pp. 1313–1326, 2024.

[19] D. Walsh, *Podman in Action: Secure, Rootless Containers for Kubernetes, Microservices, and More*. New York, NY, USA: Simon and Schuster, 2023.

[20] "Sysbox: An open-source, next-generation "runc" that empowers rootless containers to run workloads such as Systemd, Docker, Kubernetes, just like VMs," (n.d.). [Online]. Available: https://github.com/nestybox/sysbox

[21] NVIDIA CUDA GL, "extends the CUDA images by adding support for OpenGL through libglvnd," (n.d.). [Online]. Available: https://gitlab.com/nvidia/container-images/cudagl

[22] TigerVNC, "High performance, multi-platform VNC client and server," (n.d.). [Online]. Available: https://github.com/TigerVNC/tigervnc

[23] "The virtualGL project, background," (n.d.). [Online]. Available: https://virtualgl.org/About/Background

[24] J. Martin, P. Ossman, and S. Mannehed, "noVNC - the open source VNC JavaScript client." (n.d.). [Online]. Available: https://novnc.com

[25] R. Raudmäe et al., "Open source and portable educational kits for enabling robotics education," in *Robotics in Education*, R. Balogh, D. Obdržálek, and M. Fislake, Eds. Berlin, Germany: Springer, 2024, pp. 275–282.